TÉCNICAS DE PROGRAMACIÓN

Martín Gustavo Salcedo Quiñones

ULADECH CATÓLICA
UNIVERSIDAD CATÓLICA LOS ÁNGELES CHIMBOTE

Utex

Martín Gustavo Salcedo Quiñones

El docente, Martín Gustavo Salcedo Quiñones egresado de la Escuela Profesional de Ingeniería de Computación y Sistemas, de la Universidad Privada Antenor Orrego, con título profesional de Ingeniero Informático y de Sistemas emitido por la Universidad Particular de Chiclayo, con maestría en Tecnologías de Información y Comunicaciones de la Universidad Católica Los Ángeles de Chimbote, con experiencia en un inicio de la profesión en proyectos informáticos y en el ámbito de la docencia por más de 20 años en instituciones de nivel superior.

Martín Gustavo Salcedo Quiñones

TÉCNICAS DE PROGRAMACIÓN

UNIVERSIDAD CATÓLICA LOS ÁNGELES
CHIMBOTE

TÉCNICAS DE PROGRAMACIÓN

Martín Gustavo Salcedo Quiñones

Editado por:

Universidad Católica Los Ángeles de Chimbote
Jr. Tumbes S/N, L8, Centro Comercial Financiero, Chimbote, Ancash - Perú
Telf: (51-043) 327846
www.uladech.edu.pe

Primera edición digital, enero 2021.

ISBN: 978-612-4308-32-1

Impreso en Perú / Printed in Perú

Diagramación digital:

Ediciones Carolina (Trujillo).

Libro digital disponible en:
http://repositorio.uladech.edu.pe/handle/123456789/19892

DEDICATORIA

A Dios Todopoderoso, a la madre Santísima Virgen María, a mi amada esposa, por su apoyo y compresión que me brinda cada día para alcanzar nuevos retos, tanto en lo profesional como en lo personal y a mi hijo con el amor de siempre Gabriel Salcedo Vejarano, a quien siempre protegeré y cuidaré para verlo como una persona de buenos valores y capaz de enfrentar al mundo.

Contenido

Capítulo II

HERENCIA, CLASES ABSTRACTAS, INTERFACES, POLIMORFISMO, PAQUETES Y EXCEPCIONES

Introducción

Estimado estudiante:

La asignatura Técnicas de Programación se ubica en el II ciclo según el plan de estudios de la Escuela Profesional de Ingeniería de Sistemas, facultad de Ingeniería de la Universidad Católica Los Ángeles de Chimbote. El curso está desarrollado para que el estudiante logre la competencia de elaborar aplicaciones a nivel de consola usando la programación orientado al objeto y como lenguaje de programación Java a través de la utilización del entorno de desarrollo Netbeans.

Este curso de Técnicas de Programación es de suma importancia en la línea de la programación ya que sienta las bases para el logro de desarrollar sistemas informáticos y por estrategia pedagógica se ha dividido en dos unidades de aprendizaje, las misma que contienen los temas necesarios para programar pensando en objetos.

La primera unidad, contiene el uso de los métodos de las principales clases de Java, la creación y uso de los métodos estáticos, recursividad de métodos, uso de arreglos en métodos, creación de clases y sus miembros (atributos y métodos) y posterior instancia de objetos, utilización de relaciones de clases y uso de modificadores de nivel de acceso a los miembros de una clase.

La segunda unidad, contiene el uso del operador o palabra reservada this y manejo de métodos constructores, herencia de clases, clases abstractas, interfaces, polimorfismo, paquetes y uso de excepciones y finalmente el desarrollo de una proyecto o aplicación que abarque lo mayoría de los temas mencionados anteriormente.

Ing. Martín Gustavo Salcedo Quiñones

Capítulo I

USO Y CREACIÓN DE MÉTODOS, ATRIBUTOS, RELACIONES ENTRE CLASES Y MODIFICADORES DE ACCESO

1. MÉTODOS DE CLASES DE JAVA

1.1. Estructura de una aplicación de Java sencilla

```
public class HolaULADECH {

    public static void main(String args[])
    {
        System.out.println("!Hola ULADECH! ");
    }

}
```

Hacer una aplicación sencilla usando el lenguaje de programación Java implica definir una clase y dentro de ella establecer un método principal llamado main. El compilador de Java busca a éste método y procede a ejecutar las instrucciones.

La clase que contenga al método principal main debe tener el nivel de acceso público. El nombre de la clase debe ser el mismo al nombre del archivo creado para la programación, este archivo tiene la extensión java. En el ejemplo anterior al colocar public class Hola ULADECH se está declarando la clase pública llamada HolaULADECH. Por lo tanto, el archivo de tener como nombre HolaULADECH.java. En el método main con la instrucción System.out.println ("! Hola ULADECH!") se logra imprimir el mensaje ¡Hola ULADECH!

Para compilar un programa de Java se realiza lo siguiente

C:\Program Files\Java\jdk1.8.0_131\bin>javac HolaULADECH.java <Intro>

Un programa en Java tiene ciertas características tales como:

- Cada sentencia o instrucción en un programa escrito en Java se escribe comúnmente en minúsculas
- El caracter de punto y coma (;) se utiliza para indicar la terminación de una instrucción con la posibilidad de hacer otra instrucción.
- Un grupo bloque de instrucciones es un conjunto de sentencias y/o declaraciones de Java que tienen al inicio y al final el uso de llaves {}

  Por ejemplo para comenzar a definir la clase se comienza con el uso de una llave abierta { y termina con una llave cerrada }. Así mismo, el método main inicia con un { y termina con una}. El uso de los bloques de instrucciones en un programa de Java es importante para señalar el grupo de instrucciones que se va a ejecutar.

1.2. La clase BufferedReader para entrada de datos

Todo programa tiene entrada y salida de datos. Para la entrada de datos se podría usar el método readLine y para la salida de datos se podría usar el método println. Para esto será necesario usar System.in para ingreso de datos y System.out para mostrar resultados o mensajes. Cuando se ingresa datos toma el formato de tipo String y según la necesidad se podrá hacer conversiones en caso se pretenda hacer uso del dato ingresado en un tipo de dato diferente a lo ingresado. A continuación, un ejemplo que permite imprimir la expresión “El resultado es “unido al contenido de la variable de memoria R”.

```
System.out.println(“El resultado es “+R);
```

La manera de realizar un ingreso de dato en una sola lines es definiendo una variable de memoria de tipo String, donde el método readLine lee el dato hasta el final de la línea. Luego de pulsar Enter el dato ingresado se almacena en la variable de memoria de tipo String. El método readLine pertenece a la clase BufferedReader y en el

momento de la creación o instancia de un objeto es necesario de InputStreamReader que a su vez se crea con System.in. Por ejemplo:

```
BufferedReader leer = new BufferedReader (new InputStreamReader(System.in));
String nombre;
System.out.println("Introduzca un nombre: ");
nombre = in.readLine();
```

Se puede apreciar en el ejemplo anterior que se crea un objeto denominado leer de tipo BufferedReader y se hace instanciando de InputStreamReader y de System.in lo que logra que el objeto leer tenga la capacidad de leer datos. Se declara una variable de memoria llamada nombre de tipo String, se imprime la expresión "Introduzca un nombre: "y luego se procede al ingreso del dato desde el teclado y una vez pulsado la tecla Enter se almacena el dato ingresado en la variable de memoria nombre.

1.3. La clase Math

La clase Math representa la librería matemática de Java. Contiene una lista grande de métodos estáticos para realizar una serie de cálculos que para ser usados no habrá necesidad de instanciar objetos a partir de esta clase pública llamada Math.

Math tiene una referencia a las dos constantes más utilizadas en matemática con una precisión de 15 decimales (suficiente para la mayoría de los mortales). Si ejecutamos:

```
System.out.println("e = " + Math.E);
System.out.println("pi = " + Math.PI);
```

veremos:

```
e = 2.718281828459045
pi = 3.141592653589793
```

Los métodos estáticos más conocidos y usados que tiene la clase Math son:

Método de la clase Math	Para qué sirve
Math. abs(x)	Para calcular el valor absoluto de x
Math. sin(double)	Para calcular el seno de un ángulo expresado en radianes
Math. cos(double)	Para calcular el coseno de un ángulo en radianes
Math. tan(double)	Para calcular la tangente de un ángulo en radianes
Math. sqrt(double)	Para calcular la raíz cuadrada de un número.
Math. pow(a,b)	Para calcular la potencia dado la base a y el exponente b
Math. round(x)	Para obtener el valor del número redondeado al entero
Math. random()	Para obtener un número aleatorio entre 0 y 1
Math. max(a,b)	Para obtener el número mayor de dos números
Math. min(a,b)	Para obtener el número menor de dos números

1.4. La clase Integer

Cada tipo de dato numérico tiene su propia clase. Así el tipo *int* tiene el objeto *Integer*. Se han codificado muchos métodos útiles dentro de la clase Integer, pero lo más usados son:

- String Integer.**toString**(int i,int base);
- String Integer.**toString**(int i);

  Por ejemplo:

  int valor=48;

  String cadena;

  cadena=Integer.toString(valor); //no colocarle base se asume que es base 10

- int I.**parseInt**(String s,int base);
- int I.**parseInt**(String s);

  Por ejemplo:

  String dato=”30”;

  int valor;

  valor=Integer.parseInt(dato); //no colocarle base se asume que es base 10

1.5. La clase Character

El trabajo con caracteres implica hacer uso de muchas funciones que lleven hacer comprobaciones y traslación de datos. Las funciones la podemos encontrar en la clase Character. Esta clase permite crear instancias de objetos al contrario de la clase Math donde sus métodos se pueden usar de manera directa.

Declaraciones

La primera sentencia creará una variable de tipo carácter y la segunda declarará un objeto Character:

```
char c;
Character C;
```

Comprobaciones booleanas

```
Character.isLowerCase(c) //devuelve verdadero si el character está en
                           minúscula
Character.isUpperCase(c)  //devuelve verdadero si el character está en
                           mayúscula
Character.isDigit(c)  // devuelve verdadero si el carácter es un dígito numérico
Character.isSpace(c) //devuelve verdadero si el character es un espacio y
                       falso todo lo contrario
```

A partir de la variable objeto Character C, no se podría hacer C.isUpperCase(), porque la variable C no ha sido instanciado con el predicado new. Instanciarlo permitirá hacer uso de los métodos de la clase Character

Traslaciones de caracteres

```
char c1 = Character.toLowerCase( c ); //lo pone en minúscula el contenido
                                        de c y lo almacena en c1
char c2 = Character.toUpperCase( c ); //lo pone en mayúscula el
                                        contenido de c y lo almacena en c2
```

Traslaciones de carácter/dígito

```
int i = Character.digit( c,base ); //toma el contenido de c y devuelve el
                                     número para ser almacenado en i
```

```
char c = Character.forDigit( i,base ); //toma el contenido de i y devuelve
                                        el caracter para la variable c
```

Métodos de la clase Character

```
C = new Character( 'A' ); //crea el objeto C asignando el carácter A a un
                           atributo
char c = C.charValue(); //devuelve el dato como character y lo almacena en
                         la variable c
String s = C.toString(); //devuelve el dato como cadena de caracteres y es
                          almacenado en s
```

1.6. La clase Float

Cada tipo de dato numérico tiene su propia Clase y por lo tanto se podría crear objetos. El tipo de dato primitivo float tiene la clase Float que a su vez contiene muchos métodos para diferentes usos

Declaraciones

La primera instrucción creará una variable de tipo float y la segunda declara un objeto Float:

```
float f;
Float F;
```

Conversiones de Clase/Cadena

```
String s = Float.toString( f ); //toma el contenido de f y lo convierte a String y
                                 lo almacena en s
f = Float.valueOf( "3.14" ); //toma el dato de cadena “3.14”, lo convierte a
                              float y lo almacena en f
```

Comprobaciones

```
boolean b = Float.isNaN( f );
boolean b = Float.isInfinite( f );
```

El método *isNaN()* comprueba si f es un *No-Número* . Un ejemplo de no-número es raiz cuadrada de -2. El método isInfinite() evalúa si un número específico es infinitamente grande en magnitud devolviendo verdadero.

1.7. La clase Double

Cada tipo de dato numérico tiene su propia Clase y por lo tanto se podría crear objetos. El tipo de dato primitivo double tiene la clase Double que a su vez contiene muchos métodos para diferentes usos.

Declaraciones

La primera instrucción creará una variable double y la segunda un objeto Double:

```
double d;
Double D;
```

Métodos más usados en Double

```
boolean D.equals(); //evalúa la igualdad de dos datos y devuelve verdadero si lo son
String D.toString(); //devuelve el dato como cadena de caracteres
int D.intValue(); //devuelve el dato como valor numérico entero (tipo int)
long D.longValue(); //devuelve el dato como valor numérico entero largo (tipo long)
float D.floatValue(); //devuelve el dato como valor numérico de tipo flotante
double D.doubleValue(); //devuelve el dato como valor numérico de tipo double
Double V.valueOf( String s ); //toma el contenido de s y devuelve el dato como double
```

1.8. Clase String

La clase String almacena un conjunto de caracteres, siendo estos caracteres tratados como una unidad. La cadena de caracteres puede contener letras, dígitos numéricos, caracteres especiales y otros. En Java una cadena de caracteres es un objeto instanciado de la clase String. La secuencia de caracteres se escribe entre comillas.

Un objeto creado a partir de la clase String, tiene como finalidad encapsular un dato tipo cadena de caracteres, haciendo que internamente se materialice en un array de char (1)

Por ejemplo:

"Computación", "Java es fácil", "Viva el Perú", "ULADECH 2020"

Los métodos más utilizados son:

cadena.**length()**: El método length devuelve el número total de caracteres de una cadena.

Ejemplo:

```
String nombre = "Martín";
int n;
n = cadena.length();
System.out.println("El nombre "+ nombre + "tiene "+ n + "caracteres");
```

Devolverá el mensaje *El nombre Martín tiene 6 caracteres*

cadena1.**equals**(cadena2): El método equals devuelve verdadero si las dos cadenas son iguales y false en caso contrario.

Ejemplo:

```
String nombre1 = "Maria";
String nombre2 = "Mariano";
if (nombre1.equals(nombre2))
        System.out.println("Los nombres son iguales");
else
        System.out.println("Los nombres no son iguales");
```

Devolverá el mensaje *Los nombres no son iguales*

cadena1.**equalsIgnoreCase**(cadena2): El método equalsIgnoreCase no toma en cuenta la diferencia entre letras mayúsculas y minúsculas de cada String al realizar la comparación.

Ejemplo:

```
String nombre1="Mariano";
String nombre2="MARIANO";
```

```
if(nombre1.equalsIgnoreCase(nombre2))
        System.out.println("Los nombres son iguales");
else
        System.out.println("Los nombres no son iguales");
```

Devolverá el mensaje *Los nombres son iguales*

cadena1.**compareTo**(cadena2) : El método compareTo retorna un numero entero que permite evaluar lo siguiente:

devuelve un valor > 0, si la cadena1 es mayor que la cadena2

devuelve un valor = 0, si la cadena1 es igual a la cadena2

devuelve un valor < 0, si lacadena1 es menor que la cadena2

Este método implica una comparación lexicográfica en función a los códigos que tiene los caracteres según la tabla ASCII, es decir, compara caracter por caracter.

Ejemplo:

```
String dato1="Mar";
String dato2="Mary";
if(dato1.compareTo(dato2)>0)
        System.out.println("El dato1 es mayor que el dato2")
else
        if(dato1.comparaTo(dato2)<0)
                System.out.println("El dato 1 es menor que el dato2");
        else
                System.out.println("El dato 1 es igual al dato2");
```

Devolverá el mensaje *El dato1 es menor que el dato2*

¿Por qué?

Compara los primeros caracteres "M" == "M" y sigue adelante

Luego compara los segundos caracteres "a" == "a" y sigue adelante

Posteriormente los terceros caracteres "r" == "r" y sigue adelante.

La variable dato1 ya no tiene más caracteres para comparar, en cambio la variable dato2 le queda aún el carácter "y" lo que hace a éste último mayor al dato1, es decir, el dato1 es menor al dato2.

cadena1.**compareToIgnoreCase**(cadena2): El método compareToIgnoreCase ignora la diferencia existente entre letras mayúsculas y minúsculas al momento de realizar la comparación de la cadena 1 con la cadena2.

Ejemplo:

```
String dato1="mariano";
String dato2="Mariana";
if(dato1.compareToIgnoreCase(dato2)>0)
        System.out.println("El dato1 es mayor que el dato2")
else
        if(dato1.comparaToIgnoreCase(dato2)<0)
                System.out.println("El dato 1 es menor que el dato2");
        else
                System.out.println("El dato 1 es igual al dato2");
```

Devolverá el mensaje *El dato1 es mayor que el dato2*

cadena.**charAt**(indice) : El método charAt retorna el carácter de la cadena según el valor del indice.

Ejemplo:

```
String nombre = "Felipe";
Char x;
X=nombre.charAt(3); // x toma el valor del carácter i.
```

Los índices de una cadena de caracteres inician en 0.

cadena.**toUpperCase**() : El método tiene la finalidad de convertir la cadena en letras mayúsculas

Ejemplo:

```
String nombre=”Fernando”;
String nom=nombre.toUpperCase();
System.out.println(nom);
```

Devolverá el mensaje *FERNANDO*

cadena.**toLowerCase()** : El método tiene la finalidad de convertir la cadena en letras minúsculas

Ejemplo:

```
String nombre=”FERNANDO”;
String nom=nombre.toLowerCase();
System.out.println(nom);
```

Devolverá el mensaje *fernando*

cadena.**trim()** : El método tiene la finalidad de eliminar los espacios al inicio y al final de la cadena de caracteres.

Ejemplo:

```
String dato1=”      Luis        ”;
String dato2=dato1.trim();
System.out.println(Sin uso de trim: ”+dato1);
System.out.println(Con uso de trim: ”+dato2);
```

Imprimirá los mensajes *Sin uso de trim: Luis*
Con uso de trim: Luis

cadena1.**startsWith**(cadena2): Evalúa si la cadena1 inicia con la cadena2 devolviendo el valor de verdadero.

Ejemplo:

```
String nombre1= “Maria”;
String nombre2= “Mariano”;
```

```
if(nombre2.starsWith(nombre1))
        System.out.println(nombre2+" comienza con "+nombre1);
else
        System.out.println(nombre2+" no comienza con "+nombre1);
```

Imprimirá el mensaje *Mariano comienza con Maria*

cadena1.**endsWith**(cadena2) : El método evalúa si la cadena1 termina con cadena2 y si es así devuelve verdadero.

Ejemplo:

```
String nombre1= "Computadora";
String nombre2= "dora";
if(nombre1.endsWith(nombre2))
        System.out.println(nombre1+" finaliza con "+nombre2);
else
        System.out.println(nombre1+" no finaliza con "+nombre2);
```

Imprimirá el mensaje *Computadora finaliza con dora*

cadena.**indexOf(**caracter) : El método retorna el índice de la primera ocurrencia del carácter.

Ejemplo:

```
String cadena="ULADECH es el mejor";
int posicion=cadena.indexOf("e");
System.out.println("La expresión "+cadena+" la letra e se encuentra
                    en la posición "+posicion);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la letra e se encuentra en la posición 8*

cadena.**indexOf**(carácter,posicion): El método retorna el índice de la primera ocurrencia del carácter, comenzando la búsqueda a partir de lo que se indica en posición.

Ejemplo:

```
String cadena=”ULADECH es el mejor”;
int indice=cadena.indexOf(“e”,9);
System.out.println(“La expresión “+cadena+” la letra e se encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la letra e se encuentra en la posición 11*

cadena1.**indexOf**(cadena2): El método retorna el índice de la primera ocurrencia de la cadena2 en cadena1.

Ejemplo:

```
String cadena=”ULADECH es el mejor”;
int indice=cadena.indexOf(“es”);
System.out.println(“La expresión “+cadena+” la expresión es se encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la expresión es se encuentra en la posición 8*

cadena1.**indexOf**(cadena2,posición): El método retorna el índice de la primera ocurrencia de la cadena2 en cadena1, iniciando la búsqueda a partir de posición cadena1.

Ejemplo:

```
String cadena=”ULADECH es el mejor”;
int indice=cadena.indexOf(“el”;10);
System.out.println(“La expresión “+cadena+” la expresión es se encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la expresión el se encuentra en la posición 11*

cadena.**lastIndexOf(**caracter) : El método retorna el índice de la última ocurrencia del carácter.

Ejemplo:

```
String cadena=”ULADECH es el mejor”;
int posicion=cadena.indexOf(“e”);
System.out.println(“La expresión “+cadena+” la última letra e se
                    encuentra en la posición “+posicion);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la última letra e se encuentra en la posición 15*

cadena.**lastIndexOf**(carácter,posicion) : El método retorna el índice de la última ocurrencia del caracter , iniciando la búsqueda a partir de lo que indica posición.

Ejemplo:

```
String cadena=”Viva el Peru”;
int indice=cadena.indexOf(“e”,5);
System.out.println(“La expresión “+cadena+” la última letra e se
                    encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión Viva el Peru la letra e se encuentra en la posición 9*

cadena1.**lastIndexOf**(cadena2): devuelve el índice de la última ocurrencia de cadena2 en cadena1.

Ejemplo:

```
String cadena=”ULADECH es el mejor”;
int indice=cadena.indexOf(“me”);
System.out.println(“La expresión “+cadena+” la última expresión me
                    se encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión ULADECH es el mejor la última expresión es se encuentra en la posición 14*

cadena1.**lastIndexOf**(cadena2, posición): El método retorna el índice de la última ocurrencia de la cadena2 en cadena1, iniciando la búsqueda a partir de lo indica posición.

Ejemplo:

```
String cadena=”ULADECH CATÓLICA”;
int indice=cadena.indexOf(“CA”;10);
System.out.println(“La expresión “+cadena+” la última expresión es
                    se encuentra en la posición “+indice);
```

Imprimirá el mensaje *La expresión ULADECH CATÓLICA la última expresión CA se encuentra en la posición 14*

cadena.**substring**(indice) : El método retorna una subcadena con los caracteres que se inician en indice e incluye todos los caracteres siguientes hasta la finalización de la cadena.

Ejemplo:

```
String dato=”COMPUTADORA”;
String extrae=dato.substring(dato,7);
System.out.println(extre);
```

Imprimirá el mensaje *DORA*

cadena.**substring**(indiceInicial, indiceFinal) : El método retorna una subcadena con los caracteres formados a partir de la posición que indica el indiceInicial incluyendo todos los caracteres hasta el IndiceFinal-1.

Ejemplo:

```
String dato=”ULADECH Católica”;
String extrae=dato.substring(3,7);
System.out.println(extrae);
```

Imprimirá el mensaje *DECH*

String.valueOf(): Convierte tipos de datos primitivos a cadenas. La clase String posee métodos sobrecargados llamdos valueOf para soportar cada uno de los tipos primitivos de Java.

Ejemplo:

String cad1 = String.valueOf(12);

String cad2 = String.valueOf(true);

String cad3 = String.valueOf('T');

String cad4 = String.valueOf(12.9867);

1.9. Programas resueltos en Java usando NetBeans

1) Hacer un programa para calcular el área de un triángulo dada la base y la altura

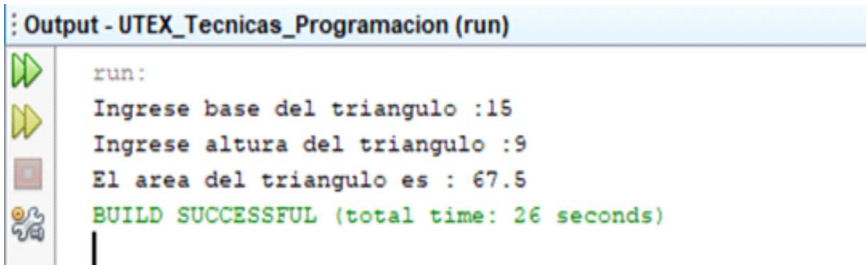
```
import java.io.*;

public class AreaTriangulo {
    public static void main(String arg[]) throws IOException
    {
        double base,altura,area;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Ingrese base del triangulo :");
        base=Double.parseDouble(in.readLine());
        System.out.print("Ingrese altura del triangulo :");
        altura=Double.parseDouble(in.readLine());
        area=base*altura/2.0;
        System.out.println("El area del triangulo es : "+area);
    }
}
```

***Interpretación de la programación*:**

Al usar import.java.io.*; importamos el paquete io (**i**nput **o**utput) que permitirá luego para el ingreso y salida de datos. Tenemos la clase AreaTriangulo que tienen al método estático llamado main que al colocar (String args[]) indica que no hay argumento alguno para el método y al colocar throws IOException son nuestra indicación al compilador de que estamos conscientes del hecho de que puede ocurrir una excepción (intercepción de error) de entrada/salida cuando el programa intente leer del teclado y que deliberadamente estamos haciendo caso omiso de tales excepciones de entrada/salida en nuestro programa. Luego se declara las variables base, altura y

area de tipo double. Posteriormente se declara la variable **in** del tipo **BufferedReader** y se crea el objeto **in** con el operador **new** usando el método constructor **BufferedReader**, la misma que usa el **InputStreamReader**, clase o interfaz que permite la lectura de entrada a través del teclado, es decir, la variable objeto in es la permitirá luego lograr ingresar datos. Luego con el método print usado a partir de System.out permite imprimir una expresión como por ejemplo "Ingrese la base del triángulo:". En la línea de la programación base= Double.parseDouble(in.readLine()); se busca el ingreso de un dato que se lee con el método readLine() a partir del objeto in previamente creado. Dado que el dato ingresado es de tipo String o cadena es necesario la conversión a un dato del tipo de la variable que almacenará dicho dato, por lo que es necesario usar un método de conversión como es el caso de parseDouble() que hace que el dato ingresado sea convertido al tipo de dato Double y luego asignado a la variable **base**. Lo mismo se hace con respecto a la variable **altura**. En la variable área se hace el cálculo del área del triángulo para luego imprimir el área del triángulo con la instrucción System.out.println("El area del triangulo es : "+area); donde se observa el uso del método println() que permite imprimir una expresión y luego la impresión continuará en la siguiente línea. Observa dentro del paréntesis el uso del operador + que se usa como concatenación (no como operador de suma), es decir, imprime la expresión "El área del triángulo es" con el contenido de la variable **área**.

Ejecución del programa:



2) Calcular el perímetro, el área y la diagonal de un rectángulo si se ingresan los lados.

```
import java.io.*;

public class Rectangulo {
    public static void main(String arg[]) throws IOException
    {
        double lado1,lado2,perimetro,area,diagonal;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Ingrese valor del primer lado :");
        lado1=Double.parseDouble(in.readLine());
        System.out.print("Ingrese valor del segundo lado :");
        lado2=Double.parseDouble(in.readLine());
        perimetro=2*(lado1+lado2);
        area=lado1*lado2;
        diagonal=Math.sqrt(Math.pow(lado1,2)+Math.pow(lado2,2));
        System.out.println("El perimetro es : "+perimetro);
        System.out.println("El area es : "+area);
        System.out.println("La diagonal es : "+diagonal);
    }
}
```

***Interpretación de la programación*:**

La novedad en la segunda aplicación es el uso del método pow() que permite calcular la potencia dado la base y el exponente, así por ejemplo Math.pow(lado1,2) calcula el cuadrado del contenido de la variable lado1. Con el método sqrt() se logra calcular la raíz cuadrada de un número, así por ejemplo Math.sqrt(4) calcula la raíz cuadrada de 4. Tanto el método pow y el método sqrt son métodos estáticos de la clase Math lo que permite usarlos directamente de la clase.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
    run:
    Ingrese valor del primer lado :12
    Ingrese valor del segundo lado :8
    El perimetro es : 40.0
    El area es : 96.0
    La diagonal es : 14.422205101855956
    BUILD SUCCESSFUL (total time: 5 seconds)
```

3) Se desea calcular el salario neto de un trabajador, permitiendo el ingreso del nombre, las horas trabajadas y precio por hora. Se debe tener en cuenta el pago de impuestos que es 10 por ciento sobre el salario bruto.

```
import java.io.*;

public class SalarioNeto {
    public static void main(String arg[]) throws IOException
    {
        String nombre;
        double sb, sn, ph, ht;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Ingrese nombre del trabajador :");
        nombre = in.readLine();
        System.out.print("Ingrese numero de horas trabajadas :");
        ht = Double.parseDouble(in.readLine());
        System.out.print("Ingrese precio de la hora :");
        ph = Double.parseDouble(in.readLine());
        sb = ph * ht;
        sn = sb - 0.10 * sb;
        System.out.println("El Salario neto del trabajador : " + nombre + " es : " + sn);
    }

}
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
    run:
    Ingrese nombre del trabajador :Fernando
    Ingrese numero de horas trabajadas :42
    Ingrese precio de la hora :26.5
    El Salario neto del trabajador : Fernando es : 1001.7
    BUILD SUCCESSFUL (total time: 23 seconds)
```

4) Hacer un programa para convertir metros a pies y pulgadas.
1 metro = 39.37 pulgadas 1 metro = 3.2 pies

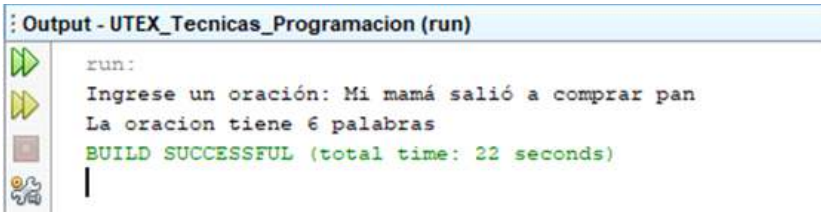
```
import java.io.*;

public class Conversion {
    public static void main(String arg[]) throws IOException
    {
        double metros, pies, pulgadas;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Ingrese valor en Metros :");
        metros = Double.parseDouble(in.readLine());
        pies = metros * 3.2;
        pulgadas = metros * 39.37;
        System.out.println("El valor en pies es : " + pies);
        System.out.println("El valor en pulgadas es : " + pulgadas);
    }
}
```

Ejecución del programa:

5) Ingresa una oración y visualiza la cantidad de palabras que hay en dicha oración.

```
import java.io.*;

public class ContandoPalabras {
    public static void main(String args[])throws IOException
    {
        String oracion;
        int i, palabras=0;
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Ingrese un oración: ");
        oracion=leer.readLine();
        for(i=0;i<oracion.length();i++)
        {
            if(oracion.substring(i,i+1).equals(" "))
                palabras++;
        }
        System.out.println("La oracion tiene "+(palabras+1)+ " palabras");
    }
}
```

Ejecución del programa:

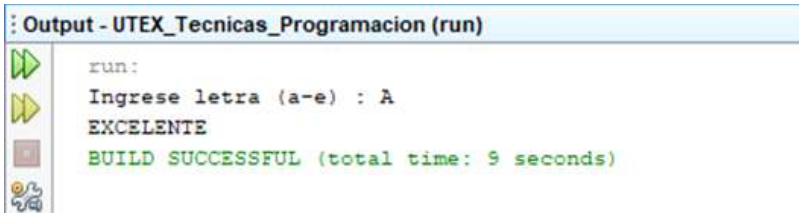


6) Ingresar una letra entre a y e y reportar el mensaje de acuerdo a:

a excelente
b bueno
c regular
d malo
e pésimo

```
import java.io.*;

public class ReportarRendimiento {
    public static void main(String arg[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String cadena;
        char letra;
        System.out.print("Ingrese letra (a-e) : ");
        cadena=br.readLine();
        letra=cadena.charAt(0);

        switch (letra) {
            case 'a': case 'A':
                System.out.println("EXCELENTE"); break;
            case 'b': case 'B':
                System.out.println("BUENO"); break;
            case 'c': case 'C':
                System.out.println("REGULAR"); break;
            case 'd': case 'D':
                System.out.println("MALO"); break;
            case 'e': case 'E':
                System.out.println("PESIMO"); break;
            default:
                System.out.println("letra equivocada");
        }
  }
}
```

Ejecución del programa:



2. CREACIÓN DE MÉTODOS ESTÁTICOS

2.1. ¿Qué son los métodos?

Los métodos en java permiten modularizar sus programas, es decir, permiten crear sub programas que cada una hagan algo específico. Todas las variables declaradas dentro de los métodos son variables locales, inclusive los parámetros que definen al método. Un método puede invocar la ejecución de otro método. La sintaxis para declarar un método es la siguiente:

```
tipo_de_valor_retorno  nombre_del_método ( lista_de_parámetros)
{
        Declaraciones e instrucciones
}
```

Donde:

tipo_de_valor_retorno: Es el tipo de valor que el método retornará una vez ejecutado las instrucciones elaboradas en el método. Si el método no retorna valor significa que dicho método deberá ser declarado como void.

nombre_del_método: Es cualquier expresión válida que permita identificar al método.

lista_de_parámetros: Es una lista de argumentos o parámetros que deberán ser separadas por comas donde se establecerán las

declaraciones de las variables que usará al método. Puede existir métodos que no tenga parámetro alguno.

El cuerpo del método es el conjunto de declaraciones e instrucciones que constituyen la lógica del método, es decir, lo que va a ejecutar. Durante la ejecución de una aplicación el programa encuentra una llamada a un método, se procede a la ejecución del método para luego retornar a la siguiente instrucción de donde se invocó al método. El método puede devolver algún valor esto puede llevar que dicho valor será recibido por una variable. Por ejemplo, si se desea calcular la suma de los primeros n números enteros. Se podría definir el siguiente método:

```
int sumaenteros(int n)
{
        int i, suma=0;
        for(i=1;i<=n;i++)
        {
                suma=suma+i;
        }
        return suma;
}
```

El tipo de retorno del método sumaenteros es de tipo entero teniendo como único parámetro la variable n. El cuerpo de instrucciones del método comienza declarando dos variables de memoria i y suma, esta última se inicializa con 0. La sentencia repetitiva for con la variable i genera los números enteros y con la variable suma va acumulando en cada interacción. Al final lo calculado en la variable de memoria suma es el valor de retorno del método. Supongamos que deseamos un método para mostrar el mensaje “ULADECH Católica es lo máximo”. Se podría definir el método de la siguiente manera:

```
void mostrarMensaje()
{
        System.out.println("ULADECH Católica es lo máximo");
}
```

En este caso el método mostrarMensaje no devuelve un valor, es por eso que el tipo de retorno a colocar es void. Lo único que ejecuta el método es mostrar un mensaje.

2.2. Invocación de métodos estáticos creados en una clase aparte

Si una clase tiene varios métodos estáticos pueden ser llamados o invocados de la siguiente manera: **NombreClase.nombreMetodo**, tal como se hace con los métodos de la clase Math, por ejemplo si uno desea calcular la raiz cuadrada se debe escribir Math.sqrt(x) y se desea calcular la potencia se debe escribir Math.pow(x,n). Es importante indicar que la clase que contiene los métodos estáticos puede estar dentro del mismo archivo con extensión de java o puede estar en un archivo de extensión de java aparte. Para un mejor manejo es mejor que pertenezcan las clases en el mismo paquete. A continuación, un ejemplo:

```
3  import java.io.*;
4
5  class Aritmetica{
6
7      public static int suma2numeros(int x, int y)
8      {
9          return x+y;
10     }
11
12     public static int resta2numeros(int x, int y)
13     {
14         return x-y;
15     }
16
17     public static int multiplica2numeros(int x, int y)
18     {
19         return x*y;
20     }
21
22     public static double divide2numeros(int x, int y)
23     {
24         return (double)x/(double)y;
25     }
26
27 }
28
```

```
29  public class CalculosAritmeticos {
30
31      public static void mensaje(String m)
32      {
33          System.out.print(m);
34      }
35
36      public static void main(String args[])throws IOException
37      {
38          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
39          int n1,n2;
40          mensaje("Ingrese el primer número: ");
41          n1=Integer.parseInt(leer.readLine());
42          mensaje("Ingrese el segundo número: ");
43          n2=Integer.parseInt(leer.readLine());
44          mensaje("La suma de los dos numeros es "+Aritmetica.suma2numeros(n1, n2)+"\n");
45          mensaje("La resta de los dos numeros es "+Aritmetica.resta2numeros(n1, n2)+"\n");
46          mensaje("La multiplicacion de los dos numeros es "+Aritmetica.multiplica2numeros(n1, n2)+"\n");
47          mensaje("La división de los dos numeros es "+Aritmetica.divide2numeros(n1, n2)+"\n");
48      }
49
50  }
```

De la figura anterior la clase Aritmetica contiene 4 métodos estáticos de las operaciones básicas de la aritmética. La clase Cálculos Aritméticos, en el método main se solicita el ingreso de dos números enteros y luego se imprime los resultados de las 4 operaciones aritméticas. Se puede apreciar que para hacer uso del método suma2numeros se debe anteponer la clase Aritmética dado que dicho método no pertenece a la clase Cálculos Aritméticos. El nombre de la clase que contiene al método main viene ser el nombre del archivo de extensión java.

2.3. Duración y alcance de un identificador

Un identificador o variable de memoria su **duración** está determinada a la existencia en la memoria del computador. En cambio, el **alcance** está determinado a la porción de un programa al que se puede hacer referencia el uso de dicha variable de memoria.

Un método puede tener variables de variables locales que pueden ser los parámetros que después del nombre del método y dentro de paréntesis y de aquellas variables que son declaradas o definidas dentro de la implementación del método. Las variables dentro de un método tienen duración automática dado que existen para el bloque donde se las definió y termina destruyéndose al término de la ejecución del bloque.

En la programación en Java se puede tener variables y métodos de duración estática que existen desde el momento en que la clase es invocada en el programa y se carga en memoria están disponibles para ser ejecutados hasta que la aplicación culmine en su ejecución. Las variables de duración estática ocupan un espacio de almacenamiento desde el momento que se cargan en memoria. En Java se hace uso de la palabra reservada static.

Los alcances de las variables de memoria o identificadores son de alcance de bloque de alcance de clase. Una variable declarada o definida afuera de todos los métodos tiene alcance de clase y puede ser utilizadas en cualquier método, en cambio las variables declaradas al principio de la implementación del método tienen alcance de bloque, lo mismo sucede con los parámetros del método.

Por ejemplo:

```
public class Matricula
{
        private String nombre;
        private static int pagoxcredito=50;

        public static double pago(int numerocreditos)
        {
                return pagoxcredito*numerocreditos
        }
}
```

Las variables de memoria nombre y pagoxcredito tienen alcance de clase, en cambio la variable numerocreditos tiene alcance de bloque.

2.4. Programas resueltos en Java usando NetBeans

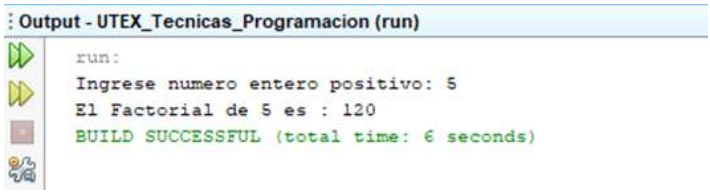
1) Hacer un programa para calcular el factorial de un número.

```
3  import java.io.*;
4
5  public class CalculoFactorial {
6      public static int factorial(int n)
7      {
8          int f=1,i;
9          for(i=1;i<=n;i++)
10                 f=f*i;
11         return f;
12     }
13
14     public static void main(String args[]) throws IOException
15     {
16         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
17         int n;
18         do {
19             System.out.print("Ingrese numero entero positivo: ");
20             n = Integer.parseInt(leer.readLine());
21         } while (n < 0);
22         System.out.println("El Factorial de "+n+" es : " + factorial(n));
23     }
24
25 }
```

***Interpretación de la programación*:**

Se tiene el método factorial que tiene un solo parámetro de tipo entero llamado **n** y que devuelve o retorna un valor de tipo entero. Este método declara dos variables de tipo entero i y f, éste último con un valor inicial de 1 para que el cálculo de sucesivas multiplicaciones no se perjudique. El método factorial es un método estático por lo que no será necesario crear o instanciar un objeto (recuerda que para instanciar un objeto se usa el operador new). En el método main se inicia con crear el objeto **leer** del tipo BufferedReader (para lectura de datos), luego se declara una variable n de tipo n (recuerda que esta variable no es el mismo del parámetro del método factorial ya que se declara en otro método). A través de la sentencia repetitiva **do** se busca validar el ingreso del valor para la variable **n** que debe ser un valor positivo, si es negativo volverá a solicitar el ingreso de otro valor para la variable **n**. Finalmente la instrucción System.out.println("el Factorial de "+n+" es: "+factorial(n)); imprimirá el factorial de n haciendo uso del método factorial.

Ejecución del programa:



2) Calcular el Máximo Común Divisor de dos números enteros.

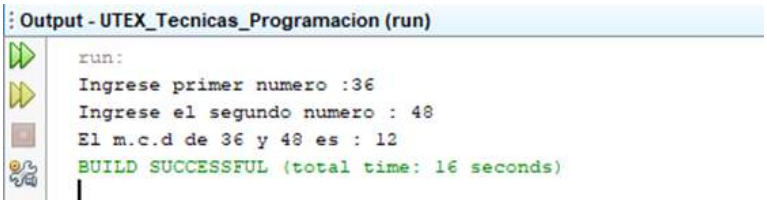
```
3  import java.io.*;
4
5  public class MaximoComunDivisor {
6      public static int mcd(int a, int b)
7      {
8          int d = 2, p = 1;
9          while (d <= a && d <= b)
10         {
11             if (a % d == 0 && b % d == 0)
12             {
13                 p = p * d;
14                 a = a / d;
15                 b = b / d;
16             } else
17             {
18                 d++;
19             }
20         }
21         return p;
22     }
23
24     public static void main(String args[]) throws IOException
25     {
26         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
27         int n1, n2, m;
28         do {
29             System.out.print("Ingrese primer numero :");
30             n1 = Integer.parseInt(leer.readLine());
31         } while (n1 < 0);
32
33         do {
34             System.out.print("Ingrese el segundo numero : ");
35             n2 = Integer.parseInt(leer.readLine());
36         } while (n2 < 0);
37         m = mcd(n1, n2);
38         System.out.println("El m.c.d de " + n1 + " y " + n2 + " es : " + m);
39     }
40
41 }
42
```

***Interpretación de la programación*:**

El método estático mcd posee dos parámetros a y b que servirá para almacenar los datos numéricos enteros, de los cuales se aplicará el

máximo común divisor. La variable de memoria d se inicializa en 2 para comenzar a sacar la mitad de los datos. La variable de memoria p almacenará el valor calculado cada vez que el divisor d es común para ambos datos se procederá a multiplicar con el valor que contenga la variable p. En la sentencia repetitiva la variable de memoria d se irá incrementándose en la medida que no se tenga un divisor común para ambos números enteros. En el método main luego de crear el objeto de lectura de datos leer, se procede a declarar las variables de memoria n1, n2 y m. Las variables de memoria n1 y n2 sirven para almacenar los datos ingresados, en esta ocasión existe una validación que consiste en asegurar que los números ingresados sean positivos y esto se logra con la sentencia repetitiva while. En la variable de memoria m almacena el valor de retorno del método estático mcd al indicar los dos números al método a través de n1 y n2. Finalmente se muestra el cálculo del máximo común divisor de los dos números enteros.

Ejecución del programa:



3) Elaborar un programa para reportar todos los divisores de un número entero N

```
import java.io.*;

public class DivisoresNumero {

    public static void mensaje(String m)
    {
        System.out.print(m);
    }

    public static void reporteDivisores(int n)
    {
        int divisor;
        mensaje("Los divisores del numero son :\n");
```

```
        for(divisor=1;divisor<=n;divisor++)
            if(n%divisor==0)
                mensaje(divisor+" ");
        mensaje("\n");
    }

    public static void main(String args[]) throws IOException
    {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        int num;
        do {
            mensaje("Ingrese numero :");
            num = Integer.parseInt(leer.readLine());
        } while (num <= 0);
        reporteDivisores(num);
    }
}
```

***Interpretación de la programación*:**

El método estático mensaje imprime una expresión en línea según el contenido de la variable de memoria m, como no retorna valor alguno el método se le antepone a su nombre la palabra void. El método estático reporteDivisores evalúa los posibles divisores desde del valor 1 hasta el valor del número del cual se esta sacando los divisores, para esto es necesario hacer uso de una sentencia repetitiva como for que evalúa cada posible divisor y si cumple que no hay residuo imprime el divisor seguido de un espacio para el posible siguiente divisor. Este método reporteDivisores como no retorna valor alguno se hace uso de la expresión void antes del nombre del método. Finalmente, en el método estático main, que significa principal, se alista la variable objeto leer para lectura de datos, se pide el ingreso del número, validando que sea un número entero positivo y luego se hace uso del método reporteDivisores dando como valor al parámetro lo que almacena la variable num.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Ingrese numero :32
Los divisores del numero son :
1 2 4 8 16 32
BUILD SUCCESSFUL (total time: 5 seconds)
```

4) Programa para ingresar un número y se reporte si es primo

```
import java.io.*;

public class NumeroPrimo {

    public static boolean esPrimo(int n)
    {
        int divisor = 1;
        do {
            divisor = divisor + 1;
        } while (n % divisor != 0 && divisor * divisor <= n);
        if (divisor * divisor > n) {
            return true;
        } else {
            return false;
        }
    }

    public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int num;
        do {
            System.out.print("Ingrese numero para ser evaluado:");
            num = Integer.parseInt(br.readLine());
        } while (num <= 0);
        if (esPrimo(num)) {
            System.out.println("El numero es primo");
        } else {
            System.out.println("El numero no es primo");
        }
    }
}
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Ingrese numero para ser evaluado:13
El numero es primo
BUILD SUCCESSFUL (total time: 2 seconds)
```

5) Crea una clase llamada Matematicas que contenga los métodos estáticos esPrimo para evaluar si un número es primo o no lo es, factorial para calcular el factorial de un número, mcd para calcular el máximo común divisor de dos números y reporteDivisores para listar los divisores de un número. Luego realiza operaciones usando los métodos antes mencionados.

```java
import java.io.*;

class Matematicas {

    public static boolean esPrimo(int n) {
        int d = 1;
        do {
            d = d + 1;
        } while (n % d != 0 && d * d <= n);
        if (d * d > n) {
            return true;
        } else {
            return false;
        }
    }

    public static int factorial(int n) {
        int f = 1, i;
        for (i = 1; i <= n; i++) {
            f = f * i;
        }
        return f;
    }

    public static int mcd(int a, int b) {
        int d = 2, p = 1;
        while (d <= a && d <= b) {
            if (a % d == 0 && b % d == 0) {
                p = p * d;
                a = a / d;
                b = b / d;
            } else {
                d++;
            }
        }
        return p;
    }

    public static void reporteDivisores(int n) {
        int divisor;
        System.out.println("Los divisores del numero son :");
        for (divisor = 1; divisor <= n; divisor++) {
            if (n % divisor == 0) {
                System.out.print(divisor + " ");
            }
        }
        System.out.println();
    }
}

public class UsandoMatematicas {

    public static void main(String args[]) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n1, n2;
        do {
            System.out.print("Ingrese numero :");
            n1 = Integer.parseInt(br.readLine());
        } while (n1 < 0);
        System.out.println("el Factorial es : " + Matematicas.factorial(n1));
        if (Matematicas.esPrimo(n1)) {
            System.out.println("El numero es primo");
        } else {
            System.out.println("El numero no es primo");
        }
```

```
68            System.out.println("");
69            Matematicas.reporteDivisores(n1);
70            do {
71                System.out.print("Ingrese un segundo numero :");
72                n2 = Integer.parseInt(br.readLine());
73            } while (n2 < 0);
74            System.out.println("El maximo comun dividor de "+n1+" y "+n2+" es "+Matematicas.mcd(n1,n2));
75        }
76    }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Ingrese numero :6
el Factorial es : 720
El numero no es primo

Los divisores del numero son :
1 2 3 6
Ingrese un segundo numero :10
El maximo comun dividor de 6 y 10 es 2
BUILD SUCCESSFUL (total time: 14 seconds)
```

3. RECURSIVIDAD Y SOBRECARGA DE MÉTODOS

3.1. ¿Qué es recursividad?

La recursividad en un método consiste cuando el mismo método se invoca a sí mismo después de alguna evaluación de alguna instrucción. Esta instrucción comúnmente es una sentencia selectiva que luego de evaluar culmina la ejecución del método o retorna un resultado que involucra una nueva ejecución del método. Esta forma de programar con métodos recursivos implica que un problema complejo sea dividido en varias partes hasta lograr la solución.

La recursividad del método termine sucede cuando se agota toda posibilidad de continuar, lo cual implica que en algún momento se debe de dejar de invocar al método recursivo para que éste deje de seguir ejecutándose. El uso de los métodos recursivos evita el uso de las sentencias repetitivas, estas últimas muy usadas en aquellos métodos que son iterativos.

En la recursividad de métodos se presenta la sentencia selectiva como la instrucción que invocará la ejecución del método recursivo o la terminación del mismo.

3.2. Recursividad vs Iteración

La recursividad y la iteración hacen de alguna estructura de control. Mientras la iteración utiliza una estructura de control repetitiva con la presencia de las llamadas sentencias de repetición

(for y while) en cambio la recursión es una estructura de selección dado que se evalúa una expresión lógica y se decide la repetición de la ejecución del método o la terminación del mismo.

Tanto la forma de programar con iteración como la recursión implican hacer repeticiones de instrucciones. El uso de la iteración emplea de forma explícita una estructura de repetición mientras que la recursión ejecuta la repetición a través de métodos. La iteración como la recursión en algún momento debe culminar de seguir ejecutándose; la iteración termina cuando ya no se puede cumplir con la condición para continuar con el bucle; la recursión termina cuando se deja de invocar al método recursivo.

Por ejemplo:

```
public static int sumaParesRecursivo(int n)
{
        if(n ==0)
                return 0;
        else
                return n*2+sumaParesRecursivo(n-1);
}
```

Este método sumaParesRecursivo es un método recursivo porque a través de una sentencia de selección if se evalúa el valor del parámetro n. Suponiendo que el valor de n es 4 entonces lo que se desea es sumar los cuatro primeros números pares que vendría ser 2, 4, 6, y 8. El uso del método sumaParesRecursivo donde n es 4 haría que en el if sea falso dando como retorno n*2+sumaParesRecursivo(n-1), que es lo mismo, 4*2+sumaParesRecursivo(3), entonces sería 8 + la ejecución de nuevo del método. En la segunda invocación del método if sigue siendo falso lo hace un retorno de 3*2+sumaParesRecursivo(3-1), es decir, 6+sumaParesRecursivo(2). El método se ejecuta por tercera vez y nuevamente la instrucción if es falso lo que ocasiona un retorno de 2*2+sumaParesRecursivo(2-1), es decir, 4+sumaParesRecursivos(1).

El método se ejecuta por cuarta vez y nuevamente la instrucción if es falso lo que permite un retorno de 1*2+sumaParesRecursivo(1-1), es decir, 2+sumaParesRecursivo(0). El método se ejecuta por quinta vez y es aquí donde la sentencia if recién se hace verdadero dando como retorno el valor de 0. Esto todo quiere decir que el método recursivo suma de la siguiente manera: 8+6+4+2+0, dando como resultado final 20.

```
public static int sumaParesIterativo(int n)
{
        int par=0, i, sumapares=0;
        for(i=1;i<=n;i++)
        {
                par=par+2;
                sumapares=sumapares+par;
        }
        return sumaPares;
}
```

Este método sumaParesIterativo es un método iterativo porque repite una o más instrucciones a través de una sentencia repetitiva. En el ejemplo anterior se puede apreciar que las variables de memoria par y sumapares se inicializan en cero. Dentro de la sentencia repetitiva for la primera instrucción par=par+2 permite la generación de los pares, cada par se genera en cada iteración, luego la variable sumapares acumula la suma de los pares que se van generando (sumapares=sumapares+par;). Sólo basta con la invocación del método una sola vez para que calcule la suma de los primeros n números pares.

3.3. Sobrecarga de métodos

Java permite definir varios métodos con el mismo nombre en tanto dichos métodos tengan diferentes juegos de parámetros (con

base en el número y el orden de los parámetros). Esto se denomina sobrecarga de métodos. Cuando se invoca un método sobrecargado, el compilador de Java selecciona el método adecuado examinando el número, los tipos y el orden de los argumentos en la llamada. La sobrecarga de métodos suele utilizarse para crear varios métodos con el mismo nombre que realizan tareas similares, pero sobre datos de diferentes tipos.

Por ejemplo:

```
import java.io.*;
public class sobrecargaMetodos{

    public static int cuadrado(int n)
    {
        return n*n;
    }

    public static float cuadrado(float n)
    {
        return n*n;
    }

    public static double cuadrado(double n)
    {
        return n*n;
    }
```

```
    public static void main(String args[]) throws IOException
    {
        int a=4;
        float b=8.2f;
        double c=14.5;
        System.out.println("El cuadrado de "+a+ " es :
                                        "+cuadrado(a));
        System.out.println("El cuadrado de "+b+" es :
                                        "+cuadrado(b));
        System.out.println("El cuadrado de "+c+" es :
                                        "+cuadrado(c));
    }
}
```

La clase sobrecargaMetodos tiene 4 métodos, de los cuales 3 de ellos tienen el mismo nombre cuadrado. El primer método cuadrado definido en la clase sobrecargaMetodos tiene como parámetro una variable de tipo entero y el valor de retorno es también de tipo de dato entero. Este método el compilador de Java lo usa cuando se desea calcular el cuadrado de un número entero. El segundo método cuadrado mencionado en la clase el parámetro es de tipo float y el valor de retorno es de tipo float. El compilador de Java usa este método cuando el cálculo tiene como número un valor de tipo flotante. El tercer método cuadrado tiene como parámetro una variable de tipo double y el valor de retorno también es de tipo double. El compilador de Java usa este método cuando se desea calcular el cuadrado de un número de tipo real (double).

En el método main (principal) se inicializa 3 variables de memoria a, b y c recibiendo el primero un valor de 4 (dato entero), el segundo el valor de 8.2f (dato flotante) y el tercero el valor de 14.5 (dato real). Finalmente se imprime el cuadrado de cada uno de los 3

valores teniendo en cuenta que la invocación del método cuadrado, el compilador de Java usa el método apropiado para el cálculo que este caso usa el primer método cuadrado para calcular el cuadrado de 4, el segundo método cuadrado para el cálculo del cuadrado de 8.2f y el tercer método cuadrado para el cálculo del cuadrado de 14.5

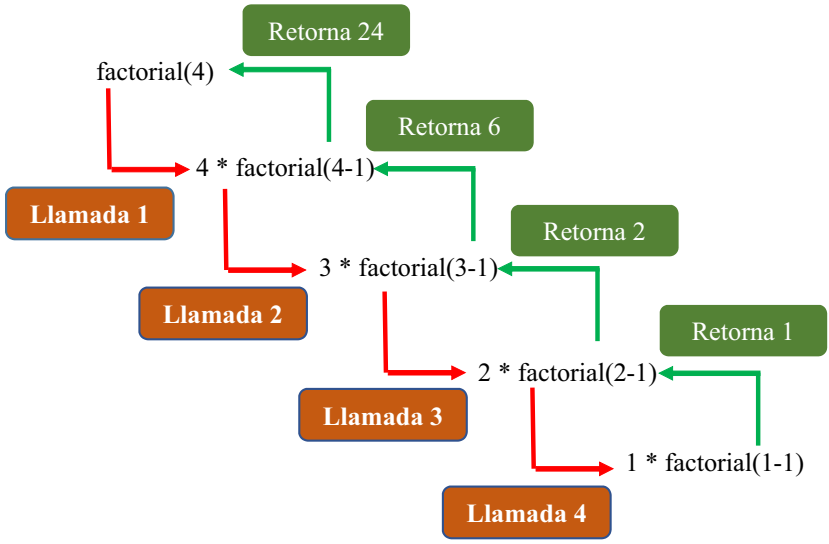
3.4. Programas resueltos en Java usando NetBeans

1) Elaborar una aplicación que permita calcular el factorial de un número.

```
3  import java.io.*;
4
5  public class RecursividadFactorial {
6
7      public static int factorial(int n)
8      {
9          if(n==0)
10             return 1;
11         else
12             return n*factorial(n-1);
13     }
14
15     public static void main(String args[]) throws IOException
16     {
17         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
18         int num;
19         do{
20             System.out.print("Ingrese numero :");
21             num=Integer.parseInt(br.readLine());
22         }while(num<=0);
23         System.out.println("El factorial es : "+factorial(num));
24     }
25 }
26
```

***Interpretación de la programación*:**

El método recursivo llamado factorial tiene un parámetro de tipo entero llamado n. Se evalúa su valor si es igual cero usando la sentencia IF, si es falso se retorna el valor de n multiplicado con lo que retornará el método factorial al darle el valor de n disminuido en 1.



En la figura se observa el método factorial calcula el factorial de 4. En la primera llamada retorna 4 * factorial(4-1), el factorial(4-1) en la segunda llamada del método retorna 3 * factorial(3-1), en la tercera llamada del método retorna 2 * factorial(2-1) y en la cuarta llamada del método retorna 1* factorial(1-1). El factorial de 0 retorna 1 y con la multiplicación con 1 devuelve 1; el factorial de (2-1) con la multiplicación de 2 retorna 2; el factorial de (3-1) con la multiplicación con 3 retorna el valor de 6 y el factorial de (4-1) con la multiplicación de 4 retorna el valor de 4.

En el método main se crea el objeto br de tipo BufferedReader (para lectura de datos), luego se declara la variable num de tipo entero. A través de la sentencia repetitiva se pide el ingreso del número para la variable num y si es un valor negativo se volverá a solicitar nuevamente el ingreso de dicho número. Finalmente, a través del método println se muestra el cálculo del método recursivo factorial teniendo como valor de entrada el contenido de la variable **num**.

Ejecución del programa:

2) Calcular la potencia de un número elevado a la n en forma recursiva. Considere que el número sea de tipo real y n sea un número entero positivo. Use recursividad.

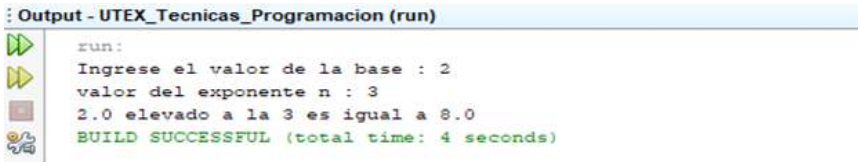
```
import java.io.*;

public class PotenciaNumero {

    public static void mensaje(String m)
    {
        System.out.print(m);
    }

    public static double potencia(double x, double n) {
        if (n == 0) {
            return 1;
        } else {
            return x * potencia(x, n - 1);
        }
    }

    public static void main(String args[]) throws IOException {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        int n;
        double num;
        mensaje("Ingrese el valor de la base : ");
        num = Double.parseDouble(leer.readLine());
        do {
            mensaje("valor del exponente n : ");
            n = Integer.parseInt(leer.readLine());
        } while (n <= 0);
        mensaje(num + " elevado a la " + n + " es igual a " + potencia(num, n)+"\n");
    }

}
```

***Interpretación de la programación*:**

El programa está compuesto por tres métodos estáticos: el primero llamado mensaje que es un método no recursivo, el segundo llamado potencia que es un método recursivo y el tercero llamado main (principal) que hace uso de los dos métodos anteriores. El método recursivo llamado método evalúa a través de una sentencia de selección if si el exponente es 0 y si no lo es entonces retorna el valor de la base x multiplicado por la ejecución nuevamente del método

llamado potencia donde se mantiene el valor de la base x y el exponente se disminuye en 1. El valor del exponente en algún momento llega a ser 0 y devuelve el valor de 1. En el método main se define la variable objeto de lectura llamado leer, se declara las variables de memoria n de tipo entero y num de tipo real. Se hace una validación para que el valor que reciba la variable de memoria n sea mayor de 0 y esto se logra a través de la sentencia repetitiva do while. Finalmente se imprime la potencia del numero elevado al valor de la variable de memoria n.

Ejecución del programa:



3) Elabora un programa que permita calcular el sueldo de un trabajador administrativo o de un docente de una institución educativa. Un administrativo tiene un sueldo base y se le descuenta por los días que ha faltado y por la cantidad de minutos de tardanza. Por cada día de falta se le descuenta el 3% de su sueldo base y 0.50 soles por cada minuto de tardanza. El empleado docente se le paga por cada hora trabajada y solo se le descuenta por un seguro que es 12% de sueldo. Calcula el sueldo neto de un administrativo o docente. Hacer uso de métodos sobrecargados

```
3  import java.io.*;
4
5  public class SueldoNeto {
6
7      public static void mensaje(String m)
8      {
9          System.out.print(m);
10     }
11
12     public static int leerDatoEntero()throws IOException
13     {
14         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15         return Integer.parseInt(leer.readLine());
16     }
17
```

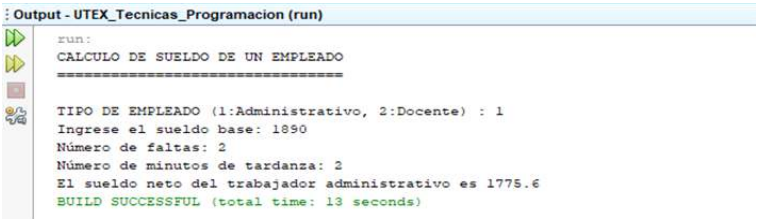
```
18      public static double leerDatoReal()throws IOException
19      {
20          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
21          return Double.parseDouble(leer.readLine());
22      }
23
24      public static double sueldoNeto(double sbase, int nrofalta, int mintardanzas)
25      {
26          return sbase-nrofalta*0.03*sbase-mintardanzas*0.5;
27      }
28
29      public static double sueldoNeto(double pagoxhora, int nrohoras)
30      {
31          return pagoxhora*nrohoras*0.88;
32      }
33
34      public static void main(String args[])throws IOException
35      {
36          int tipo, numhoras, nfaltas, mintar;
37          double sueldobas, montoxhora;
38          mensaje("CALCULO DE SUELDO DE UN EMPLEADO\n");
39          mensaje("================================\n");
40          mensaje("\n");
41          mensaje("TIPO DE EMPLEADO (1:Administrativo, 2:Docente) : ");
42          tipo=leerDatoEntero();
43          if(tipo==1)
44          {
45              mensaje("Ingrese el sueldo base: ");
46              sueldobas=leerDatoReal();
47              mensaje("Número de faltas: ");
48              nfaltas=leerDatoEntero();
49              mensaje("Número de minutos de tardanza: ");
50              mintar=leerDatoEntero();
51              mensaje("El sueldo neto del trabajador administrativo es "+
52                      sueldoNeto(sueldobas,nfaltas,mintar)+"\n");
53          }
54          else
55          {
56              mensaje("Ingrese el monto a pagar por hora: ");
57              montoxhora=leerDatoReal();
58              mensaje("Cantidad de horas trabajadas: ");
59              numhoras=leerDatoEntero();
60              mensaje("El sueldo neto del trabajador docente es "+
61                      sueldoNeto(montoxhora,numhoras)+"\n");
62          }
63      }
64
65  }
```

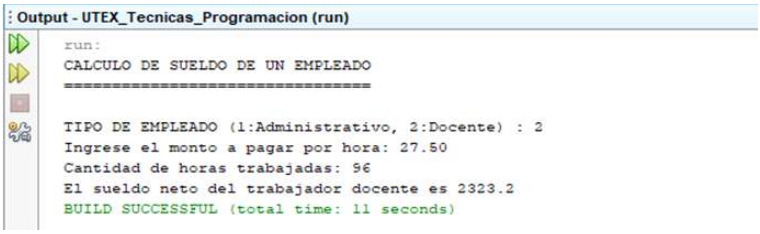
***Interpretación de la programación*:**

En este programa se crea dos métodos de lectura de datos, uno para leer un dato de tipo entero y otro método para leer dato de tipo real. Además, se tiene dos métodos con el mismo nombre llamados sueldoNeto (sobrecarga de métodos), el primero con tres parámetros para calcular el sueldo neto de un administrativo y el segundo con dos parámetros para calcular el sueldo neto de un docente. En el método main, se solicita primero indicar si el trabajador es de tipo

administrativo o docente, según lo indicado se procede a solicitar los demás datos. Finalmente se hace del método sobrecargado sueldoNeto para mostrar el resultado monto neto a recibir del trabajador.

Ejecución del programa:





4) Hacer un programa para que reporte los n términos de la serie de Fibonacci usando recursividad.

```
3  import java.io.*;
4
5  public class RecursividadFibonacci {
6
7      public static void mensaje(String m)
8      {
9          System.out.print(m);
10     }
11
12     public static int leerDatoEntero()throws IOException
13     {
14         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15         return Integer.parseInt(leer.readLine());
16     }
17
18     public static int fibonacci(int n)
19     {
21         if (n == 1) {
22             return 1;
23         } else if (n == 2) {
24             return 1;
25         } else {
26             return fibonacci(n - 1) + fibonacci(n - 2);
27         }
```

```
28
29      public static void main(String args[]) throws IOException
30      {
31          int n, i;
32          mensaje("**********************************\n");
33          mensaje("*        SERIE FIBONACCI         *\n");
34          mensaje("**********************************\n");
35          mensaje("\n");
36          do {
37              mensaje("Número de terminos de la serie : ");
38              n = leerDatoEntero();
39          } while (n <= 0);
40          for (i = 1; i <= n; i++) {
41              mensaje(fibonacci(i) + " ");
42          }
43          System.out.println("\n");
44      }
45
46  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
**********************************
*         SERIE FIBONACCI        *
**********************************

Número de terminos de la serie : 10
1 1 2 3 5 8 13 21 34 55

BUILD SUCCESSFUL (total time: 4 seconds)
```

5) Programa para calcular el máximo común divisor de dos números.

```
3   import java.io.*;
4
5   public class RecursividadMCD {
6
7       public static void mensaje(String m)
8       {
9           System.out.print(m);
10      }
11
12      public static int leerDatoEntero()throws IOException
13      {
14          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15          return Integer.parseInt(leer.readLine());
16      }
17
18      public static int mcd(int a,int b)
19      {
20          if (a % b == 0) {
21              return b;
22          } else {
23              return mcd(b, a % b);
24          }
25      }
```

```
26
27      public static void main(String args[]) throws IOException
28      {
29          int n1, n2;
30          mensaje("////////////////////////////\n");
31          mensaje("//  Maximo Común Divisor   //\n");
32          mensaje("////////////////////////////\n");
33          mensaje("\n");
34          do {
35              mensaje("Ingrese primer numero : ");
36              n1 = leerDatoEntero();
37          } while (n1 <= 0);
38          do {
39              mensaje("Ingrese segundo numero : ");
40              n2 = leerDatoEntero();
41          } while (n2 <= 0);
42          System.out.println("El mcd de " + n1 + " y " + n2 + " es : " + mcd(n1, n2));
43      }
44
45  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
////////////////////////////
//  Maximo Común Divisor   //
////////////////////////////

Ingrese primer numero : 20
Ingrese segundo numero : 30
El mcd de 20 y 30 es : 10
BUILD SUCCESSFUL (total time: 10 seconds)
```

6) Elabore un programa para reportar un numero al revés. Hacer uso de recursividad.

```
3   import java.io.*;
4
5   public class RecursividadNroInvertido {
6
7       public static void reves(int n)
8       {
9           mensaje(n % 10+"");
10          if (n / 10 != 0) {
11              reves(n / 10);
12          }
13      }
14
15      public static void mensaje(String m)
16      {
17          System.out.print(m);
18      }
19
20      public static int leerDatoEntero()throws IOException
21      {
22          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
23          return Integer.parseInt(leer.readLine());
24      }
25
```

```
26      public static void main(String args[]) throws IOException
27      {
28          int num;
29          mensaje("====================\n");
30          mensaje("  Número al revés   \n");
31          mensaje("====================\n");
32          mensaje("\n");
33          do {
34              mensaje("Ingrese numero :");
35              num = leerDatoEntero();
36          } while (num <= 0);
37          mensaje("Numero al reves :");
38          reves(num);
39          System.out.println();
40      }
41
42  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
====================
  Número al revés
====================

Ingrese numero :76453
Numero al reves :35467
BUILD SUCCESSFUL (total time: 7 seconds)
```

7) Desarrolle un programa para convertir un número de base 10 a base b (que b se encuentre entre 2 y 9). Hacer uso de recursividad.

```
3   import java.io.*;
4
5   public class RecursividadConvertirBase {
6
7       public static void mensaje(String m)
8       {
9           System.out.print(m);
10      }
11
12      public static int leerDatoEntero()throws IOException
13      {
14          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15          return Integer.parseInt(leer.readLine());
16      }
17
18      public static void conversionBase(int n, int b)
19      {
20          if (n < b) {
21              mensaje(n+"");
22          } else {
23              conversionBase(n / b, b);
24              mensaje(n % b+"");
25          }
26      }
27
```

```
28      public static void main(String args[]) throws IOException
29      {
30          int num, base;
31          mensaje("=============================================\n");
32          mensaje("Convesrsión de un número de base 10 a otra base\n");
33          mensaje("=============================================\n");
34          mensaje("\n");
35          do {
36              mensaje("Ingrese un numero de base 10 :");
37              num = leerDatoEntero();
38          } while (num <= 0);
39          do {
40              System.out.print("Base a la que quiere convertir (entre 2 a 9): ");
41              base = leerDatoEntero();
42          } while (base < 2 || base > 9);
43          mensaje("El numero " + num + " en base " + base + " es : ");
44          conversionBase(num, base);
45          mensaje("\n");
46      }
47
48  }
49
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
=============================================
Convesrsión de un número de base 10 a otra base
=============================================

Ingrese un numero de base 10 :6789
Base a la que quiere convertir (entre 2 a 9): 8
El numero 6789 en base 8 es : 15205
BUILD SUCCESSFUL (total time: 8 seconds)
```

4. ARREGLOS UNIDIMENSIONALES EN MÉTODOS

4.1. ¿Qué son arreglos?

Un arreglo es una estructura de datos capaz de almacenar un grupo de datos, es una variable de memoria compuesta de varios elementos esperando de ser llenados de datos del mismo tipo. Estos elementos dentro de la estructura poseen una posición denominada índice, vale decir que cada elemento del arreglo guarda un dato y tiene un índice que es un número correlativo que va desde 0. Para referirnos a un elemento del arreglo se debe indicar tanto el nombre como el índice.

Por ejemplo:

Un elemento del arreglo es numeros[1] que contiene el valor numérico de 38. El tamaño del arreglo numeros es de 5, por lo tanto, tiene 5 elementos que desde elemento numros[0] hasta el elemento numeros[4]. Los arreglos de Java comienzan siempre con el elemento con índice 0.

4.2. Declaración de un arreglo unidimensional o vector

En el lenguaje de programación de java existen 2 formas de declarar arreglos.

<tipo de dato> variable[];

ó

<tipo de dato> []variable;

Por ejemplo, si se desea declarar un arreglo para números enteros, se procede hacerlo de la siguiente manera:

int numeros[]; ó int []numeros;

4.3. Creación de un arreglo unidimensional o vector

Una vez declarado un arreglo, se tiene que crearlo. Para la creación se usa la palabra clave new seguida del tipo de dato y del tamaño del arreglo que debe estar dentro de los corchetes.

Por ejemplo:

```
numeros = new int[50];   // se está creando un arreglo de 50
                            elementos enteros
```

También se puede declarar y crear un arreglo en una sola instrucción.

Por ejemplo:

```
double promedios[] = new double[30];
```

Se está declarando y creando el arreglo promedios de tamaño 30 elementos para datos de tipo double.

4.4. Inicialización de un arreglo unidimensional o vector

En el lenguaje de programación Java se pueden inicializar arreglos al declararlos, es decir, se puede crear arreglos con valores o datos iniciales. Cuando se indica datos iniciales dentro de la declaración de un arreglo, Java procede internamente a hacer uso del

operador new definiendo el tamaño del arreglo de manera automática. Se debe hacer uso de llaves { } para listar los datos.

Ejemplo:

```
int numeros [] = { 1, 2, 3, 4, 5};

boolean arreglo[] = { true, false, true, true};

String meses[] = { "Enero", "Febrero", "Marzo", "Abril",
                   "Mayo", "Junio"};
```

4.5. ¿Cómo acceder a los datos almacenados en el arreglo?

Cada dato colocado en un arreglo se conoce como elemento del arreglo. Para acceder al contendido de un elemento de un arreglo especifique el nombre del arreglo con el índice del elemento entre corchetes [].

Ejemplo:

```
int numeros = { 12, 20, 80, 70, 90 };

for(int  indice = 0; indice < 5 ; indice++)
        System.out.println(numeros[indice]);
```

En este ejemplo el tamaño del arreglo es 5. En el lenguaje de programación Java el índice del primer elemento del arreglo es cero y la última posición es el tamaño del arreglo menso 1, es decir, un arreglo de tamaño 5 el índice va desde 0 hasta 4. Debido a esto, para iterar con todos los elementos del arreglo, el programa debe barrer con todos los elementos que se puede lograr cuando se use una sentencia repetitiva.

En el lenguaje de programación Java un arreglo es un objeto. Siendo un objeto puede tener métodos y propiedades. La propiedad length (longitud) contiene el tamaño del arreglo. La propiedad length es de solo lectura, dado que el tamaño del arreglo se define en el momento de la creación.

Por ejemplo: El siguiente código de programación muestra la forma de usar la propiedad length dentro de una sentencia for que itera con todos los elementos del arreglo.

```
int numeros[] = {30, 20, 50, 70, 40};

for(indice = 0; indice<numeros.length; indice++)
        System.out.println(numeros[indice]);
```

4.6. Referencias a arreglos unidimensionales

El lenguaje de programación de Java usa referencias para apuntar hacia otros arreglos.

Por ejemplo:

Las siguientes instrucciones utilizan la referencia *arreglo* para acceder a dos arreglos distintos.

```
import java.io.*;
public class ReferenciandoArreglos{
  public static void main(String args[])
  {
     int pares[] = {2, 4, 6, 8,10};
     int impares[] = {1, 3, 5, 7, 9, 11};
     int arreglo[];

     arreglo=pares;
     System.out.println("Primer Arreglo de Numeros Pares");

     for(int i=0;i<arreglo.length;i++)
           System.out.println(arreglo[i]); 1
```

```
        arreglo=impares;
        System.out.println("Segundo Arreglo de Números Impares");

        for(int i=0;i<arreglo.length;i++)
                System.out.println(arreglo[i]);
    }
}
```

En la parte A la instrucción arreglo=pares; se aplica lo que es referencia, es decir, el vector llamado arreglo se define con todos los datos que tiene el arreglo par, es por ello que cuando se usa la sentencia for, imprime los datos del arreglo pares. En la parte B la instrucción arreglo=impares; al ejecutarse deja de contener los valores del arreglo pares y ahora contiene los valores del arreglo impares. Con la sentencia for se busca imprimir el contenido del arreglo y esta vez será con los datos que tiene los arreglos impares.

4.7. Creación de métodos con uso de arreglos

Siendo los arreglos variables de memoria capaz de almacenar muchos datos, pueden ser parámetros o valor de retorno de los métodos.

Por ejemplo, si deseo elaborar un método que dado el valor de la variable de memoria n que indica la generación de los primeros n números pares y se desea guardarlos en un arreglo, este arreglo debe ser el valor de retorno del método.

```
public static int [] generacionPares(int n)
{
        int par=0, i;
        int pares[]=new int[n];
        for(i=0;i<n;i++)
        {
```

```
            par=par+2;
            pares[i]=par;
        }
        return pares;
}
```

En este método se procede a la creación del arreglo de pares de tamaño n. Con la sentencia for se genera los pares en cada iteración y se asigna en el elemento correspondiente del arreglo el par generado. El método retorna el arreglo llamado pares con el contenido de los pares generados.

Otro ejemplo con uso de arreglo en métodos puede ser encontrar la suma de los números contendidos en un arreglo numérico real.

```
public static double sumaNúmeros(double x[ ])
{
        int i, total=x.length;
        double suma=0.0;
        for(i=0;i<total;i++)
            suma=suma+x[i];
        return suma;
}
```

El método sumaNúmeros el parámetro x[] es un arreglo de tipo real, con la propiedad length se obtiene el tamaño y con la sentencia for se barre con todos los elementos del arreglo x sumando de forma acumulativa en la variable de memoria suma. Finalmente, lo acumulado en la variable suma será lo que retorne el método sumaNúmeros.

4.8. Programas resueltos en Java usando NetBeans

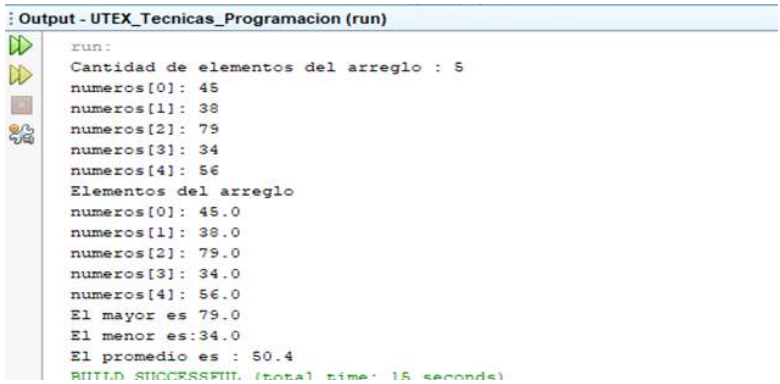
1) Elaborar un programa para ingresar n datos numéricos reales en un arreglo y los muestre en la pantalla, además reportar el mayor, el menor y el promedio.

```
import java.io.*;

public class EjercicioArreglo01 {

    public static void main(String arg[])throws IOException
    {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        double numeros[], mayor, menor, promedio, suma = 0;
        int n, i;

        do {
            System.out.print("Cantidad de elementos del arreglo : ");

            n = Integer.parseInt(leer.readLine());
        } while (n <= 0 || n > 100);

        numeros = new double[n];

        for (i = 0; i < n; i++) {
            System.out.print("numeros[" + i + "]: ");
            numeros[i] = Double.parseDouble(leer.readLine());
        }

        System.out.println("Elementos del arreglo");
        for (i = 0; i < n; i++) {
            System.out.println("numeros[" + i + "]: " + numeros[i]);
        }

        // Calculo del numero mayor y numero menor
        mayor = numeros[0];
        menor = numeros[0];
        for (i = 1; i < n; i++) {
            if (numeros[i] > mayor) {
                mayor = numeros[i];
            } else if (numeros[i] < menor) {
                menor = numeros[i];
            }
        }

        // Calculo de la suma de los elementos
        for (i = 0; i < n; i++) {
            suma = suma + numeros[i];
        }

        promedio = suma / n;
        System.out.println("El mayor es " + mayor);
        System.out.println("El menor es:" + menor);
        System.out.println("El promedio es : " + promedio);
    }

}
```

***Interpretación de la programación*:**

En el método main se inicia creando el objeto **leer** del tipo **BuffreredReader**, luego se declara variables de memoria de tipo double. Se aprecia la variable **numeros[]** que es una arreglo o vector que todavía no se define el tamaño o el número de elementos. Luego se declara dos variables enteras **n, i**. A través de la sentencia repetitiva se valida el valor de n dentro de un rango de 1 a 100, es decir, no se acepta un valor menor o igual a 0 ni un número mayor de 100. Posteriormente se crea el arreglo números con la instrucción números=new double[n]; ya que se define el tamaño del arreglo números con el valor de la variable **n**. A continuación, se usa la sentencia for que permite leer los datos para cada elemento del arreglo, la misma va desde un valor de 0 hasta i<n, es decir, si n es igual a 10 entonces la variable i irá desde 0 hasta 9. En cada interacción solicitará el ingreso de un número, la misma que se almacenará en el elemento correspondiente del arreglo. Cabe señalar que el objeto **leer** con su método ***readLine()*** lee el dato como cadena (String) entonces existe la necesidad de hacer la conversión, para ello se usa el método ***parseDouble*** de la clase **Double** que permite convertir el dato ingresado en valor numérico double. Luego se imprime los valores ingresados usando nuevamente la sentencia **for**. Para el cálculo del mayor y menor número ingresados en el arreglo vector se inicializa las variables con la instrucción mayor=números[0], menor=números[0]; donde la variable mayor y la variable menor recibe el valor del primer elemento del arreglo x. A continuación, con la sentencia for se recorre por cada elemento del arreglo números desde el segundo elemento hasta el último elemento del arreglo. En cada interacción se usa la sentencia IF para evaluar si el elemento es mayor a la variable **mayor**, si es verdadero se cambia el valor contenido en la variable mayor por el valor del elemento en evaluación, si es falso se cambia el valor del contenido en la variable menor. Posteriormente a través de otra sentencia for se procede a realizar la suma de todos los números contenidos en el arreglo y se va acumulando en la variable suma, que previamente se inicializó con el valor de cero. A continuación, se calcula el promedio y se muestra los resultados obtenidos.

Ejecución del programa:



```
Output - UTEX_Tecnicas_Programacion (run)
run:
Cantidad de elementos del arreglo : 5
numeros[0]: 45
numeros[1]: 38
numeros[2]: 79
numeros[3]: 34
numeros[4]: 56
Elementos del arreglo
numeros[0]: 45.0
numeros[1]: 38.0
numeros[2]: 79.0
numeros[3]: 34.0
numeros[4]: 56.0
El mayor es 79.0
El menor es:34.0
El promedio es : 50.4
BUILD SUCCESSFUL (total time: 15 seconds)
```

2) Del programa anterior aplicar la creación de métodos estáticos.

```
import java.io.*;

public class EjercicioArreglo02 {

    public static void mensaje(String m)
    {
        System.out.print(m);
    }

    public static int leerDatoEntero()throws IOException
    {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        return Integer.parseInt(leer.readLine());
    }

    public static double leerDatoReal()throws IOException
    {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        return Double.parseDouble(leer.readLine());
    }

    public static double[] leerDatosArreglo(double x[])throws IOException
    {
        int i;
        for (i = 0; i < x.length; i++) {
            mensaje("numeros[" + i + "]: ");
            x[i] = leerDatoReal();
        }
        return x;
    }

    public static void mostrarDatosArreglo(double x[])
    {
        int i;
        mensaje("Elementos del arreglo\n");
        for (i = 0; i < x.length; i++) {
            mensaje("numeros[" + i + "]: " + x[i]+"\n");
        }
    }
```

```
42
43      public static double mayorNumeroArreglo(double x[])
44      {
45          int i;
46          double mayor;
47          mayor = x[0];
48          for (i = 1; i < x.length; i++)
49              if (x[i] > mayor) {
50                  mayor = x[i];
51              }
52          return mayor;
53      }
54
55      public static double menorNumeroArreglo(double x[])
56      {
57          int i;
58          double menor;
59          menor = x[0];
60          for (i = 1; i < x.length; i++)
61              if (x[i] < menor) {
62                  menor = x[i];
63              }
64          return menor;
65      }
66
67      public static double sumaNumerosArreglo(double x[])
68      {
69          int i;
70          double suma=0;
71          for (i = 0; i < x.length; i++) {
72              suma = suma + x[i];
73          }
74          return suma;
75      }
76
77      public static void main(String arg[])throws IOException
78      {
79          double numeros[], mayor, menor, promedio, suma = 0;
80          int n, i;
81
82          do {
83              mensaje("Cantidad de elementos del arreglo : ");
84              n = leerDatoEntero();
85          } while (n <= 0 || n > 100);
86
87          numeros = new double[n];
88
89          numeros=leerDatosArreglo(numeros);
90
91          mostrarDatosArreglo(numeros);
92
93          promedio = sumaNumerosArreglo(numeros) / n;
94          mensaje("El mayor es " + mayorNumeroArreglo(numeros) + "\n");
95          mensaje("El menor es:" + menorNumeroArreglo(numeros) + "\n");
96          mensaje("El promedio es : " + promedio + "\n");
97      }
98
99  }
```

***Interpretación de la programación*:**

El programa tenía un solo método main y ahora tiene 8 métodos estáticos más. Se establece un método mensaje para imprimir cualquier expresión de cadena de caracteres, el método leerDatoEntero() sirve para la lectura de un dato de tipo entero, el método leerDatoReal() sirve para la lectura de un dato de tipo real, el método leerDatosArreglo() sirve para el llenado de un arreglo de tipo double, el método mostrarDatosArreglo() sirve para imprimir los datos almacenados en un arreglo de tipo double, el método mayorNumeroArreglo() permite calcular el número mayor de los números almacenados en el arreglo, el método menorNumeroArreglo() permite calcular el número menor de los números guardados en el arreglo y el método sumaNumerosArreglo() permite calcular la suma de todos los números almacenados en el arreglo de tipo double.

Ejecutando el programa usando estos métodos se obtiene los mismos resultados:

```
Output - UTEX_Tecnicas_Programacion (run)
    run:
    Cantidad de elementos del arreglo : 3
    numeros[0]: 45.90
    numeros[1]: 34.67
    numeros[2]: 20.09
    Elementos del arreglo
    numeros[0]: 45.9
    numeros[1]: 34.67
    numeros[2]: 20.09
    El mayor es 45.9
    El menor es:20.09
    El promedio es : 33.553333333333335
    BUILD SUCCESSFUL (total time: 14 seconds)
```

3) Programa para ingresar n valores reales en un arreglo y luego invierta los datos del arreglo.

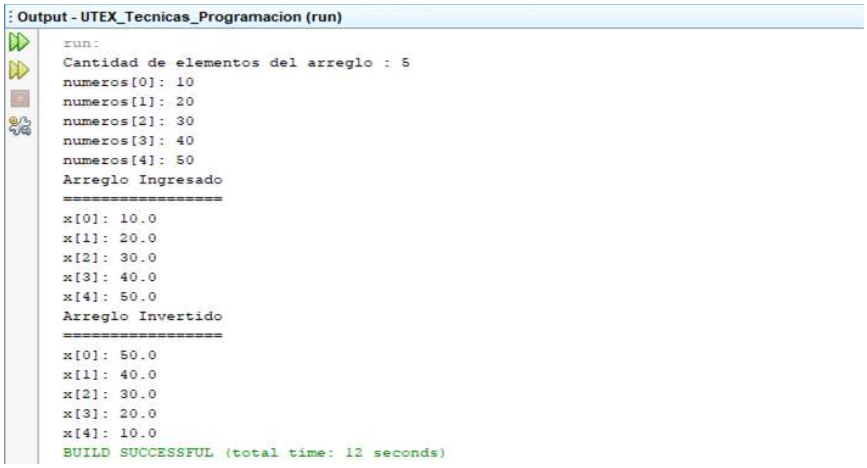
```
3  import java.io.*;
4
5  public class EjercicioArreglo03 {
6
7      public static void mensaje(String m)
8      {
9          System.out.print(m);
10     }
11
12     public static int leerDatoEntero()throws IOException
13     {
14         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15         return Integer.parseInt(leer.readLine());
16     }
```

```
17
18      public static double leerDatoReal()throws IOException
19      {
20          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
21          return Double.parseDouble(leer.readLine());
22      }
23
24      public static double[] leerDatosArreglo(double a[])throws IOException
25      {
26          int i;
27          for (i = 0; i < a.length; i++) {
28              mensaje("numeros[" + i + "]: ");
29              a[i] = leerDatoReal();
30          }
31          return a;
32      }
33
34      public static void mostrarDatosArreglo(double x[])
35      {
36          int i;
37          for (i = 0; i < x.length; i++) {
38              mensaje("x[" + i + "]: " + x[i]+"\n");
39          }
40      }
41
42      public static double[] arregloInvertido(double a[])
43      {
44          int i, j, n=a.length;
45          double temp;
46          for (i = 0, j = n - 1; i < n / 2; i++, j--) {
47              temp = a[i];
48              a[i] = a[j];
49              a[j] = temp;
50          }
51          return a;
52      }
53
54      public static void main(String arg[]) throws IOException {
55
56          double x[], temp;
57          int n, i, j;
58
59          do {
60              mensaje("Cantidad de elementos del arreglo : ");
61              n = leerDatoEntero();
62          } while (n <= 0 || n > 100);
63
64          x = new double[n];
65
66          x=leerDatosArreglo(x);
67
68          mensaje("Arreglo Ingresado\n");
69          mensaje("==================\n");
70          mostrarDatosArreglo(x);
71
72          x=arregloInvertido(x);
73
74          mensaje("Arreglo Invertido\n");
75          mensaje("==================\n");
76          mostrarDatosArreglo(x);
77
78      }
79
80  }
```

***Interpretación de la programación*:**

La novedad en esta programación es el método arregloInvertido() dado que busca invertir los valores del arreglo haciendo que el último valor ubicado en el último elemento del arreglo sea el primero, En el proceso de cambio de datos es necesario el uso de la variable temporal temp para salvar el dato a mover y luego colocar en la posición correspondiente. Los demás métodos se han usado de manera similar en el programa anterior.

Ejecución del programa:

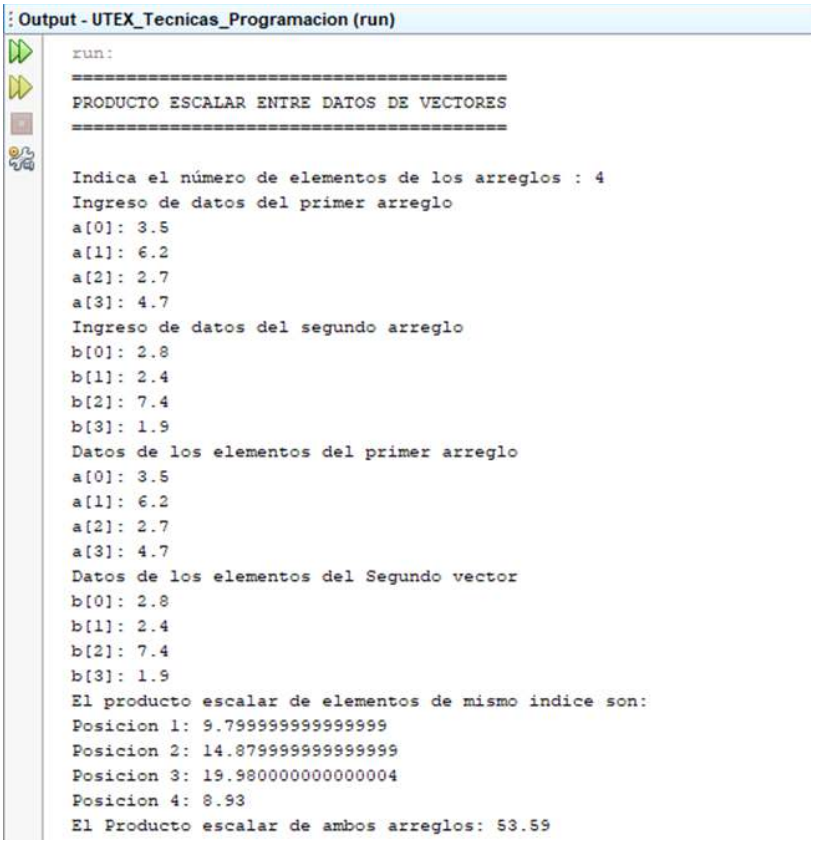


4) Elaborar un programa que permite el ingreso de datos numéricos reales en 2 vectores de n elementos cada uno y luego listar el producto escalar de ellos

```
3  import java.io.*;
4
5  public class EjercicioArreglo04 {
6
7      public static void mensaje(String m)
8      {
9          System.out.print(m);
10     }
11
12     public static int leerDatoEntero()throws IOException
13     {
14         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15         return Integer.parseInt(leer.readLine());
16     }
17
```

```
18      public static double leerDatoReal()throws IOException
19      {
20          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
21          return Double.parseDouble(leer.readLine());
22      }
23
24      public static double[] leerDatosArreglo(String nom,double x[])throws IOException
25      {
26          int i;
27          for (i = 0; i < x.length; i++) {
28              mensaje(nom+"[" + i + "]: ");
29              x[i] = leerDatoReal();
30          }
31          return x;
32      }
33
34      public static void mostrarDatosArreglo(String nom,double x[])
35      {
36          int i;
37          for (i = 0; i < x.length; i++) {
38              mensaje(nom+"[" + i + "]: " + x[i]+"\n");
39          }
40      }
41
42      public static void main(String arg[])throws IOException
43      {
44          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
45          double a[], b[], producto=0.0;
46          int n, i;
47
48          mensaje("=======================================\n");
49          mensaje("PRODUCTO ESCALAR ENTRE DATOS DE VECTORES\n");
50          mensaje("=======================================\n");
51          mensaje("\n");
52          do {
53              mensaje("Indica el número de elementos de los arreglos : ");
54              n = leerDatoEntero();
55          } while (n <= 0);
56
57          a = new double[n];
58          b = new double[n];
59
60          mensaje("Ingreso de datos del primer arreglo\n");
61          leerDatosArreglo("a",a);
62
63          mensaje("Ingreso de datos del segundo arreglo\n");
64          leerDatosArreglo("b",b);
65
66          mensaje("Datos de los elementos del primer arreglo\n");
67          mostrarDatosArreglo("a",a);
68
69          mensaje("Datos de los elementos del Segundo vector\n");
70          mostrarDatosArreglo("b",b);
71
72          mensaje("El producto escalar de elementos de mismo indice son:\n");
73          for (i = 0; i < n; i++) {
74              mensaje("Posicion "+(i+1)+": "+(a[i]*b[i])+"\n");
75              producto = producto + a[i] * b[i];
76          }
77          mensaje("El Producto escalar de ambos arreglos: " + producto + "\n");
78      }
79
80  }
```

Ejecución del programa:

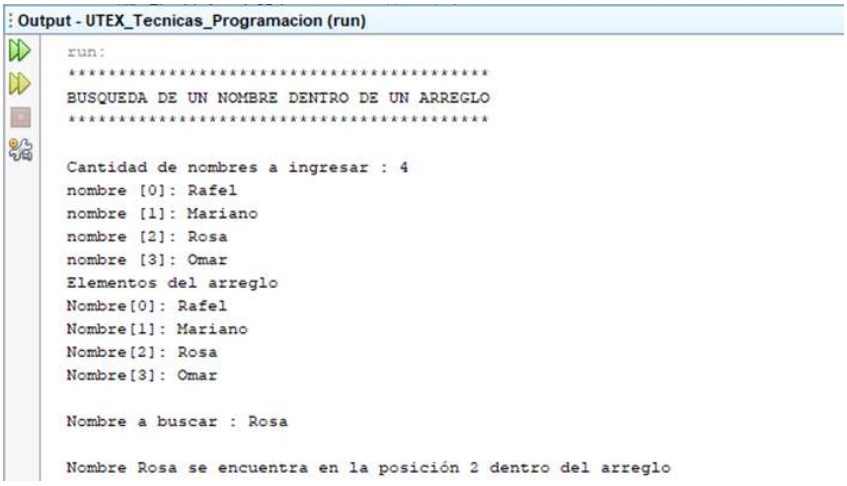


5) Elaborar un programa que permita el ingreso de n nombres en un arreglo y luego ingresar un nombre cualquiera y buscar si este se encuentra en el arreglo ingresado.

```
3  import java.io.*;
4
5  public class EjercicioArreglo05 {
6
7      public static void mensaje(String m)
8      {
9          System.out.print(m);
10     }
11
12     public static int leerDatoEntero()throws IOException
13     {
14         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
15         return Integer.parseInt(leer.readLine());
16     }
```

```
17
18      public static String leerDatoTexto()throws IOException
19      {
20          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
21          return leer.readLine();
22      }
23
24      public static String[] leerDatosArreglo(String a[])throws IOException
25      {
26          int i;
27          for (i = 0; i < a.length; i++) {
28              mensaje("nombre [" + i + "]: ");
29              a[i] = leerDatoTexto();
30          }
31          return a;
32      }
33
34      public static int posicionDatoBuscado(String nom, String vector[])
35      {
36          int posicion,i;
37          posicion = -1;
38          for (i = 0; i < vector.length; i++) {
39              if (vector[i].compareToIgnoreCase(nom) == 0) {
40                  posicion = i;
41                  break;
42              }
43          }
44          return posicion;
45      }
46
47      public static void main(String arg[])throws IOException
48      {
49          String nombres[], nombrebuscar;
50          int i, pos, n;
51
52          mensaje("*****************************************\n");
53          mensaje("BUSQUEDA DE UN NOMBRE DENTRO DE UN ARREGLO\n");
54          mensaje("*****************************************\n");
55          mensaje("\n");
56
57          do {
58              mensaje("Cantidad de nombres a ingresar : ");
59              n = leerDatoEntero();
60          } while (n <= 0);
61
62          nombres = new String[n];
63
64          nombres=leerDatosArreglo(nombres);
65
66          mensaje("Elementos del arreglo\n");
67          for (i = 0; i < n; i++) {
68              mensaje("Nombre[" + i + "]: " + nombres[i]+"\n");
69          }
70          mensaje("\n");
71          mensaje("Nombre a buscar : ");
72          nombrebuscar = leerDatoTexto();
73          mensaje("\n");
74          pos = posicionDatoBuscado(nombrebuscar,nombres);
75          if (pos != -1) {
76              mensaje("Nombre "+nombrebuscar+" se encuentra en la posición " + pos+" dentro del arreglo\n");
77          } else {
78              mensaje("Nombre "+nombrebuscar+" no se encuentra en el arreglo\n");
79          }
80      }
81
82  }
```

Ejecución del programa:



5. ESTRUCTURA DE UNA CLASE: ATRIBUTOS Y MÉTODOS

5.1. ¿Qué es una clase?

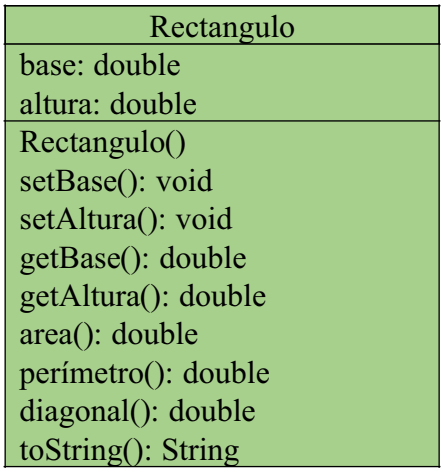
Una clase es un molde o una plantilla donde se establece los atributos (estado) y los métodos (comportamiento) comunes a todos los objetos que mantienen las mimas características. Las clases son la base a partir del cual permiten crear o instanciar objetos donde comúnmente los atributos reciben los datos o valores que permitirán procesar y generar resultados. Los objetos al contener los datos en los atributos permiten aplicar la programación orientado al objeto. Una clase posee un nombre es un conjunto de datos y de instrucciones (métodos) que actúa sobre los datos. Una clase también se define como un tipo abstracto de dato compuesto por atributos y métodos, donde los atributos hacen referencia a las características del concepto abstraído y los métodos hacen referencia al comportamiento de dicho concepto (2).

5.2. Estructura de una clase

Una clase contiene datos, estos pueden ser de tipos primitivos o referencias y los métodos. La sintaxis general para la declaración de una clase es:

```
modificadores class nombre_clase {
    declaraciones_de_atributos y métodos ;
}
```

Los modificadores o niveles de acceso son palabras clave que afecta el alcance de una clase. Por ejemplo, se crea la clase Rectángulo con los atributos base y altura y se desea calcular el área, perímetro y diagonal del Rectángulo. A continuación, el diagrama de la clase Rectángulo.



Definido los atributos y métodos se proceden a crear el código en Java para la clase Rectángulo.

```
import java.io.*;
class Rectángulo
{
        private double base;
        private double altura;

        public Rectángulo(double b, double h)
        {
             base = b;
             altura=h;
        }
```

```
public void setBase(double b)
{
    base=b;
}

public void setAltura(double h)
{
    altura=h;
}

public double getBase()
{
    return base;
}

public double getAltura()
{
    return altura;
}


public double área ()
{
    return base*altura;
}

public double perímetro()
{
    return 2*base+2*altura;
}
```

```
    public double diagonal()
    {
        return Math.sqrt(Math.pow(base,2)+Math.pow(altura,2));
    }

    public String toString()
    {
        return "base = "+base+"  "+altura;
    }
}
```

La clase Rectángulo tiene 2 atributos base y altura los cuales son privados esto quiere decir que estas 2 variables son visibles dentro de la clase Rectángulo. El primer método que se ha implementado o definido la programación es el ***constructor***, este método tiene como nombre igual a la clase, no retorna ningún valor y es de acceso público. Permite dar valores iniciales a los atributos del nuevo objeto. Es utilizado al momento de crear el objeto. Los atributos base y altura son de acceso privados, y si hay necesidad de cambiar los valores de los atributos se crean los métodos setBase(double b) y setAltura(double h) y si se desea obtener los valores de los atributos creamos los métodos getBase() y getAltura(). Además se han creado los métodos area(), perímetro() y diagonal() que permiten calcular el area, perímetro y diagonal del rectángulo usando en la implementación los atributos de la clase. En el método toString() se crea una cadena de texto con la información guardado en los atributos de la clase.

```
public class pruebaRectangulo{
  public static void main(String args[]) throws IOException
  {
      BufferedReader leer = new BufferedReader(new
                            InputStreamReader(System.in));
```

```
        double b, h;
        Rectangulo R;
        System.out.print("Ingrese el valor de la base : ");
        b=Double.parseDouble(br.readLine());
        System.out.print("Ingrese el valor de la altura : ");
        h=Double.parseDouble(br.readLine());
        R = new Rectangulo(b,h);
        System.out.println("Rectangulo : "+R);
        System.out.println("Area : "+R.area());
        System.out.println(“Perimetro : “+R.perimetro());
        System.out.println("Diagonal : "+R.diagonal());
    }
}
```

Dentro del método main de la clase Prueba Rectángulo se ha declarado dos variables de tipo primitivo double b y h y una variable R que es de tipo Rectángulo. Al colocar **Rectangulo R; s**e está declarando a R como una variable objeto de la Clase Rectángulo. La declaración no crea nuevos objetos. En la declaración (Rectángulo R) se declara una variable llamada R la cual será usada para referirnos a un objeto Rectángulo. La referencia está vacía. Una referencia vacía es conocida como referencia nula. Al colocar **R = new Rectángulo(3,4);** con el operador new creamos un objeto de la clase Rectángulo. El operador new instancia una clase asignando memoria para el nuevo objeto. El operador new requiere una llamada a un constructor de la clase a instanciar. El constructor inicializa el nuevo objeto. El operador new retorna una referencia al objeto creado. Una vez creado el objeto para poder llamar a sus métodos usamos lo siguiente objeto nombre de Método. Por ejemplo para calcular el área usamos R.area(), para calcular el perímetro R.perímetro() y para calcular la diagonal R.diagonal (). Al escribir System.out.println (“Rectángulo: “+R); en realidad se está llamando tácitamente al método toString de la clase R.

Si se desea modificar el atributo base del Objeto se debe usar el método setBase por ejemplo si después de crear el objeto queremos que base tenga el valor 10, se colocaría la siguiente instrucción: R.setBase(10); lo mismo se hace si se quiere modificar la altura. Si se desea saber el valor de algún atributo del objeto se usa los métodos get, por ejemplo, si quisiera imprimir el valor de la base del objeto R se tendría que escribir lo siguiente:

```
System.out.println("La base es: "+R.getBase());
```

5.3. Programas resueltos en Java usando NetBeans

1) Crea la clase Cilindro con los atributos radio y altura de tipo numérico real, que permita calcular el área y el volumen del cilindro.

   Los atributos y métodos de la clase Cilindro se describe a continuación:

Cilindro
radio: double altura: double
Cilindro() setRadio(): void setAltura(): void getRadio(): double getAltura(): double area(): double volumen(): double toString(): String

Definido los atributos y métodos se proceden a crear el código en Java para la clase Cilindro y para la clase ProbandoCilindro.

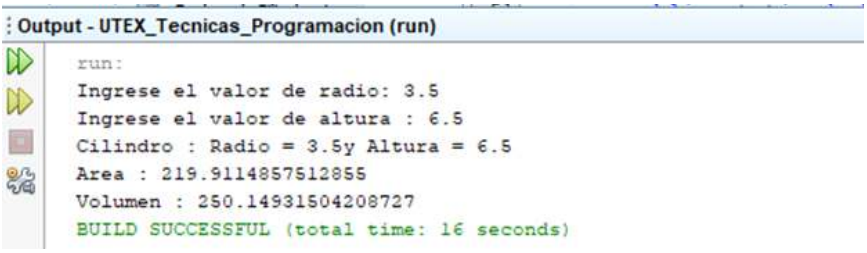
```java
package Capitulo_I.Sesion_05;

import java.io.*;

class Cilindro {

    private double radio;
    private double altura;

    public Cilindro(double r, double a) {
        radio = r;
        altura = a;
    }

    public void setRadio(double r) {
        radio = r;
    }

    public void setAltura(double a) {
        altura = a;
    }

    public double getRadio() {
        return radio;
    }

    public double getAltura() {
        return altura;
    }

    public double area() {
        return 2 * Math.PI * Math.pow(radio, 2) + 2 * Math.PI * radio * altura;
    }

    public double volumen() {
        return Math.PI * Math.pow(radio, 2) * altura;
    }

    public String toString() {
        return "Radio = " + radio + "y Altura = " + altura;
    }
}

public class ProbandoCilindro {

    public static void mensaje(String m)
    {
        System.out.print(m);
    }

    public static double leerDatoReal()throws IOException
    {
        BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
        return Double.parseDouble(leer.readLine());
    }

    public static void main(String args[]) throws IOException {
        double r, h;
        Cilindro C;
        mensaje("Ingrese el valor de radio: ");
        r = leerDatoReal();
        mensaje("Ingrese el valor de altura : ");
        h = leerDatoReal();
        C = new Cilindro(r, h);
        mensaje("Cilindro : " + C +"\n");
        mensaje("Area : " + C.area()+"\n");
        mensaje("Volumen : " + C.volumen()+"\n");
    }

}
```

***Interpretación de la programación*:**

Se crea clase Cilindro con los atributos radio y altura de tipo double. Luego se crea el método constructor Cilindro que permite establecer los valores de los dos atributos a través de los parámetros **r** y **a**. Luego se construye los métodos **setRadio()** y **setAltura()** que permite cambiar el valor del atributo radio y altura respectivamente. Con los métodos **getRadio()** y **getAltura()** se obtiene el contenido del atributo radio y altura respectivamente. Con los métodos **area()** y **volumen()** se calcula el área y el volumen del cilindro. El método **toString()** permite imprimir el radio y la altura del cilindro según el contenido de los atributos.

Con la clase **ProbandoCilindro**, se tiene el método mensaje para imprimir una cadena de texto y el método leerDatoReal para la lectura de datos de tipo real. En el método ***main***, se declara las variables de memoria **r** y **h** de tipo double y se declara la variable **C** del tipo de la clase **Cilindro**. Se ingresa los valores del radio y de la altura. Luego en la instrucción C = new Cilindro(r,h); se crea el objeto **C** (que previamente había sido declarado Cilindro C;) haciendo uso del *método constructor Cilindro* pasando los valores ingresados del radio y la altura a través de las variables de memoria **r** y **h**. A continuación, con la instrucción mensaje("Cilindro : "+C+"\n"); se imprime la expresión "Cilindro:" con lo programado en el método toString(), es decir, es lo mismo colocar ***C*** que colocar ***C.toString()***. Finalmente se imprime el área y el volumen del cilindro usando los métodos no estáticos **area()** y **volumen()** invocados desde el objeto C.

Ejecución del programa:



2) Crear una clase Producto que tenga los atributos nombre, stock y preciounitario, que permita calcular el inventario (stock * preciounitario). Desarrollar un programa que permita el ingreso de n productos para luego listar los datos con su respectivo inventario e indica el nombre del producto de mayor inventario. Hacer uso de arreglos.

```
package Capitulo_I.Sesion_05;

import java.io.*;

class Producto
{
    private String nombre;
    private int stock;
    private double preciounitario;

    public Producto(String nom, int s, double pu)
    {
        nombre=nom;
        stock=s;
        preciounitario=pu;
    }

    public void setNombre(String nom)
    {
        nombre=nom;
    }

    public void setStock(int s)
    {
        stock=s;
    }

    public void setPrecioUnitario(int pu)
    {
        preciounitario=pu;
    }

    public String getNombre()
    {
        return nombre;
    }

    public int getStock()
    {
        return stock;
    }

    public double getPrecioUnitario()
    {
        return preciounitario;
    }

    public double inventario()
    {
        return stock*preciounitario;
    }
}
```

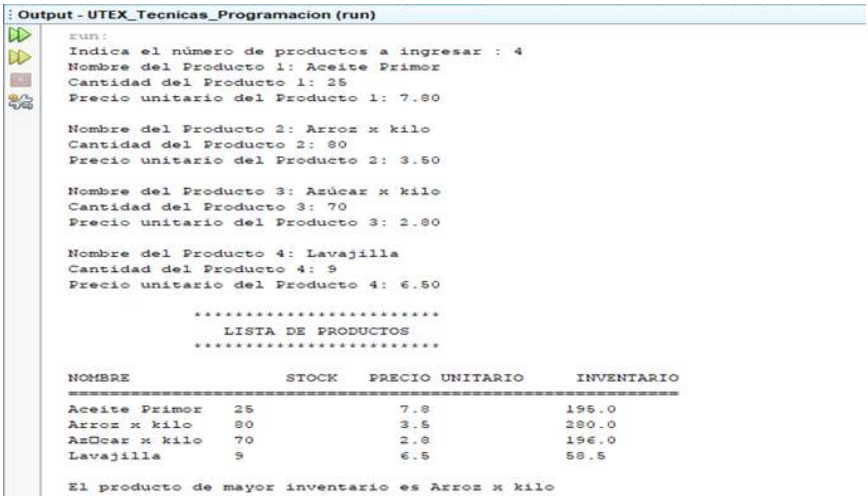
```
53
54  public class ProbandoProducto {
55
56      public static void mensaje(String m)
57      {
58          System.out.print(m);
59      }
60
61      public static int leerDatoEntero()throws IOException
62      {
63          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
64          return Integer.parseInt(leer.readLine());
65      }
66
67      public static String leerDatoTexto()throws IOException
68      {
69          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
70          return leer.readLine();
71      }
72
73      public static double leerDatoReal()throws IOException
74      {
75          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
76          return Double.parseDouble(leer.readLine());
77      }
78
79      public static void listarProductos(String n[],int c[],double p[], double inv[])
80      {
81          mensaje("            ************************\n");
82          mensaje("               LISTA DE PRODUCTOS   \n");
83          mensaje("            ************************\n");
84          mensaje("\n");
85          mensaje("NOMBRE               STOCK   PRECIO UNITARIO     INVENTARIO\n");
86          mensaje("=============================================================\n");
87          for(int i=0; i<n.length;i++)
88          {
89              mensaje(n[i]+"\t"+c[i]+"\t\t"+p[i]+"\t\t"+inv[i]+"\n");
90          }
91      }
92
93      public static String productoMayorInventario(String n[],double inv[])
94      {
95          double mayor=inv[0];
96          int i, pos=0;
97          for(i=1;i<n.length;i++)
98          {
99              if(inv[i]>mayor)
100             {
101                 mayor=inv[i];
102                 pos=i;
103             }
104         }
105         return n[pos];
106     }
107
108     public static void main(String args[])throws IOException
109     {
110         BufferedReader leer=new BufferedReader(new InputStreamReader(System.in));
111         int n, i;
112         String nombres[];
113         int cantidades[];
114         double precios[];
115         double inventarios[];
116
```

```
117             do {
118                 mensaje("Indica el número de productos a ingresar : ");
119                 n = leerDatoEntero();
120             } while (n <= 0);
121
122             nombres=new String[n];
123             cantidades=new int[n];
124             precios=new double[n];
125             inventarios=new double[n];
126
127             String nom;
128             int can;
129             double pre;
130
131             for(i=0;i<n;i++)
132             {
133                 mensaje("Nombre del Producto "+(i+1)+": ");
134                 nom=leerDatoTexto();
135                 mensaje("Cantidad del Producto "+(i+1)+": ");
136                 can=leerDatoEntero();
137                 mensaje("Precio unitario del Producto "+(i+1)+": ");
138                 pre=leerDatoReal();
139                 Producto P=new Producto(nom,can,pre);
140                 nombres[i]=P.getNombre();
141                 cantidades[i]=P.getStock();
142                 precios[i]=P.getPrecioUnitario();
143                 inventarios[i]=P.inventario();
144                 mensaje("\n");
145             }
146
147             listarProductos(nombres, cantidades,precios,inventarios);
148             mensaje("\n");
149             mensaje("El producto de mayor inventario es "+
150                     productoMayorInventario(nombres,inventarios)+"\n");
151         }
152
153     }
```

***Interpretación de la programación*:**

Se define la clase Producto con sus tres atributos nombre, stock y precio unitario. Luego se crea el método constructor con tres parámetros para definir los valores de los atributos cuando sea usado en un proceso de creación de objeto. También se crea los métodos de set y get para cada atributo y el método inventario para el cálculo del stock por el precio unitario. En la clase Probando Producto se crea varios métodos estáticos: el método mensaje que ayuda a imprimir texto, los métodos de lectura para dato de tipo entero y real, el método listarProductos está diseñado para imprimir los datos almacenados en los arreglos, el método producto Mayor Inventario que se encarga de obtener el producto de mayor inventario y por último se tiene al método main que solicitará la cantidad de productos a ingresar, se ingresa los datos y posteriormente se hace listado de lo ingresado y se muestra el nombre del producto de mayor inventario.

Ejecución del programa:



```
Output - UTEX_Tecnicas_Programacion (run)
run:
Indica el número de productos a ingresar : 4
Nombre del Producto 1: Aceite Primor
Cantidad del Producto 1: 25
Precio unitario del Producto 1: 7.80

Nombre del Producto 2: Arroz x kilo
Cantidad del Producto 2: 80
Precio unitario del Producto 2: 3.50

Nombre del Producto 3: Azúcar x kilo
Cantidad del Producto 3: 70
Precio unitario del Producto 3: 2.80

Nombre del Producto 4: Lavajilla
Cantidad del Producto 4: 9
Precio unitario del Producto 4: 6.50

            ************************
               LISTA DE PRODUCTOS
            ************************

NOMBRE                STOCK    PRECIO UNITARIO      INVENTARIO
=================================================================
Aceite Primor    25                7.8               195.0
Arroz x kilo     80                3.5               280.0
Az□car x kilo    70                2.8               196.0
Lavajilla        9                 6.5               58.5

El producto de mayor inventario es Arroz x kilo
```

3) Crea la clase Alumno con los atributos nombre, nota1 y nota2 y con métodos que permita calcular el promedio y la condición académica de aprobado o desaprobado.

```
3  import java.io.*;
4
5  class Alumno {
6
7      private String nombre;
8      private double nota1;
9      private double nota2;
10
11     public Alumno(String nom, double n1, double n2) {
12         nombre = nom;
13         nota1 = n1;
14         nota2 = n2;
15     }
16
17     public void setNombre(String nom) {
18         nombre = nom;
19     }
20
21     public void setNota1(double n1) {
22         nota1 = n1;
23     }
24
25     public void setNota2(double n2) {
26         nota2 = n2;
27     }
28
29     public String getNombre() {
30         return nombre;
31     }
```

```
32
33      public double getNota1() {
34          return nota1;
35      }
36
37      public double getNota2() {
38          return nota2;
39      }
40
41      public double promedio() {
42          return (nota1 + nota2) / 2;
43      }
44
45      public String condicion() {
46          if (promedio() >= 10.5) {
47              return "aprobado";
48          } else {
49              return "desaprobado";
50          }
51      }
52
53      public String toString() {
54          return "nombre : " + nombre + " nota1 = " + nota1 + " nota2 = " + nota2;
55      }
56  }
57
58  public class ProbandoAlumno {
59
60      public static void mensaje(String m)
61      {
62          System.out.print(m);
63      }
64
65      public static String leerDatoTexto()throws IOException
66      {
67          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
68          return leer.readLine();
69      }
70
71      public static double leerDatoReal()throws IOException
72      {
73          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
74          return Double.parseDouble(leer.readLine());
75      }
76
77      public static void main(String args[]) throws IOException {
78          String nom;
79          double n1, n2;
80          Alumno A;
81          mensaje("Ingrese nombre : ");
82          nom = leerDatoTexto();
83          mensaje("Ingrese nota1 : ");
84          n1 = leerDatoReal();
85          mensaje("Ingrese nota2 : ");
86          n2 = leerDatoReal();
87          A = new Alumno(nom, n1, n2);
88          mensaje("Alumno : " + A + "\n");
89          mensaje("Promedio : " + A.promedio()+"\n");
90          mensaje("Condicion : " + A.condicion()+"\n");
91      }
92
93  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Ingrese nombre : Manuel Vasquez
Ingrese nota1 : 14.5
Ingrese nota2 : 12.0
Alumno : nombre : Manuel Vasquez nota1 = 14.5 nota2 = 12.0
Promedio : 13.25
Condicion : aprobado
BUILD SUCCESSFUL (total time: 31 seconds)
```

4) Crea la clase Trabajador con los atributos nombre, precio Hora y horas Trabajadas y con métodos que calcule salario bruto, los impuestos (12.8% del Salario Bruto) y el salario Neto, que viene ser el salario bruto menos los impuestos.

```
3  import java.io.*;
4
5  class Trabajador {
6
7      private String nombre;
8      private double horasTrabajadas;
9      private double precioHora;
10
11     public Trabajador(String n, double ht, double ph) {
12         nombre = n;
13         horasTrabajadas = ht;
14         precioHora = ph;
15     }
16
17     public void setNombre(String n) {
18         nombre = n;
19     }
20
21     public void setHorasTrabajadas(double ht) {
22         horasTrabajadas = ht;
23     }
24
25     public void setPrecioHora(double ph) {
26         precioHora = ph;
27     }
28
29     public String getNombre() {
30         return nombre;
31     }
32
33     public double getHorasTrabajadas() {
34         return horasTrabajadas;
35     }
36
37     public double getPrecioHora() {
38         return precioHora;
39     }
40
41     public double salarioBruto() {
42         return precioHora * horasTrabajadas;
43     }
```

```
44
45      public double impuestos() {
46          return 0.128 * salarioBruto();
47      }
48
49      public double salarioNeto() {
50          return salarioBruto() - impuestos();
51      }
52
54      public String toString() {
55          return "Nombre del trabajador: " + nombre + "\nHoras Trabajadas : " +
56                  horasTrabajadas + "\nPrecio Hora : " + precioHora;
57      }
58  }
59
60  public class ProbandoTrabajador {
61
62      public static double leerDatoReal()throws IOException
63      {
64          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
65          return Double.parseDouble(leer.readLine());
66      }
67
68      public static String leerDatoTexto()throws IOException
69      {
70          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
71          return leer.readLine();
72      }
73
74      public static void imprime(String m)
75      {
76          System.out.print(m);
77      }
78
79      public static void main(String args[]) throws IOException {
80          String nom;
81          double ph, ht;
82          Trabajador T;
83          imprime("Ingrese nombre del trabajador : ");
84          nom = leerDatoTexto();
85          imprime("Ingrese numero de horas Trabajadas : ");
86          ht = leerDatoReal();
87          imprime("Ingrese precio de la Hora : ");
88          ph = leerDatoReal();
89          T = new Trabajador(nom, ht, ph);
90          imprime("============\n");
91          imprime(" RESULTADOS  \n");
92          imprime("============\n");
93          imprime( T + "\n");
94          imprime("Salario Bruto : " + T.salarioBruto() + "\n");
95          imprime("Impuestos : " + T.impuestos() + "\n");
96          imprime("Salario Neto  : " + T.salarioNeto() + "\n");
97      }
98
99  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Ingrese nombre del trabajador : Rafael Salcedo
Ingrese numero de horas Trabajadas : 60
Ingrese precio de la Hora : 27.50
=============
 RESULTADOS
=============
Nombre del trabajador: Rafael Salcedo
Horas Trabajadas : 60.0
Precio Hora : 27.5
Salario Bruto : 1650.0
Impuestos : 211.20000000000002
Salario Neto  : 1438.8
BUILD SUCCESSFUL (total time: 17 seconds)
```

5) Crea la clase Móvil que contenga los atributos velocidad Inicial, aceleración y tiempo. Usando métodos calcula la distancia recorrida por un móvil.

```
3  import java.io.*;
4
5  class Movil {
6
7      private double velocidadinicial;
8      private double aceleracion;
9      private double tiempo;
10
11     public Movil(double vi, double a, double t) {
12         velocidadinicial = vi;
13         aceleracion = a;
14         tiempo = t;
15     }
16
17     public void setVelocidadInicial(double vi) {
18         velocidadinicial = vi;
19     }
20
21     public void setAceleracion(double a) {
22         aceleracion = a;
23     }
24
25     public void setTiempo(double t) {
26         tiempo = t;
27     }
28
29     public double getVelocidadInicial() {
30         return velocidadinicial;
31     }
32
33     public double getAceleracion() {
34         return aceleracion;
35     }
36
37     public double getTiempo() {
38         return tiempo;
39     }
```

```
40
41      public String toString() {
42          return "Velocidad Inicial = " + velocidadinicial + " m/s Aceleracion = " +
43                  aceleracion + " m/s2 Tiempo = " + tiempo + " s";
44      }
45
46      public double distanciaRecorrida() {
47          return velocidadinicial * tiempo + aceleracion * Math.pow(tiempo, 2)/2.0;
48      }
49  }
50
51  public class UsandoMovil {
52
53      public static void imprime(String m)
54      {
55          System.out.print(m);
56      }
57
58      public static double leerDatoReal()throws IOException
59      {
60          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
61          return Double.parseDouble(leer.readLine());
62      }
63
64      public static void main(String args[]) throws IOException {
65          double vi, a, t;
66          imprime("=============================================\n");
67          imprime("MOVIMIENTO RECTILINEO UNIFORMEMENTE VARIADO\n");
68          imprime("=============================================\n");
69          imprime("\n");
70          imprime("Ingrese velocidad Inicial (m/s) : ");
71          vi = leerDatoReal();
72          imprime("Ingrese aceleracion (m/s2) : ");
73          a = leerDatoReal();
74          imprime("Ingrese tiempo (s): ");
75          t = leerDatoReal();
76          Movil M = new Movil(vi, a, t);
77          imprime("Movil con " + M +"\n");
78          imprime("Distancia recorrida : " + M.distanciaRecorrida()+" m\n");
79      }
80
81  }
```

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
=============================================
MOVIMIENTO RECTILINEO UNIFORMEMENTE VARIADO
=============================================

Ingrese velocidad Inicial (m/s) : 10
Ingrese aceleracion (m/s2) : 8
Ingrese tiempo (s): 12
Movil con Velocidad Inicial = 10.0 m/s Aceleracion = 8.0 m/s2 Tiempo = 12.0 s
Distancia recorrida : 696.0 m
BUILD SUCCESSFUL (total time: 19 seconds)
```

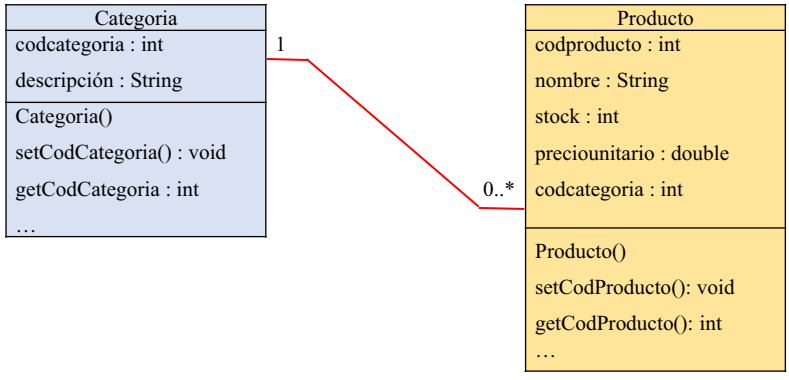
6. RELACIONES ENTRE CLASES

6.1. Uso de clases relacionadas

Un grupo de objetos aislados tiene poca capacidad para resolver un problema real dado que muchas veces hacer consultas de datos se hace a partir de la información que guarda varios objetos. En una aplicación o programa los objetos intercambian información y tienen alguna relación entre sus datos. A nivel de diseño de clases se distingue 5 tipos de relaciones básicas entre clases de objetos: dependencia, asociación, agregación, composición y herencia, siendo la asociación la relación más importante y del cual se usa más. La relación de tipo asociación es la relación entre dos clases a partir de algún atributo en común que los ligue y mantenga una comunicación de datos al momento de ser utilizados a nivel de objetos. La relación de clases se establece las cardinalidades en los extremos.

6.2. Diagramando clases relacionadas

Se tiene el siguiente diagrama de clases:



Estas dos clases Categoría y Producto existe una relación cuya cardinalidad consiste en que cada categoría puede estar en ninguna o en muchos productos. Por ejemplo, en una bodega se ofrece productos comestibles y productos de limpieza. Si se desea crear objetos de las clases Categoría y Producto y se desea presentar a nivel gráfico podría ser así:

Objeto: C1 (creado a partir de la clase Categoría)

1	Bebidas

Objeto: C2 (creado a partir de la clase Categoría)

2	Lácteos

Objeto: C3 (creado a partir de la clase Categoría)

3	Condimentos

Objeto: P1 (creado a partir de la clase Producto)

1	Leche evaporada Gloria	50	3.40	2

Objeto: P2 (creado a partir de la clase Producto)

2	Cocacola 2 litros	30	5.60	1

Objeto: P3 (creado a partir de la clase Producto)

3	Yogurt Laive 1 litro	40	5.50	2

Con lo graficado se observa la presencia de 3 objetos de la clase Categoría y 3 objetos de la clase de Producto. Se tiene las categorías Bebidas, Lácteos y Condimentos con los códigos de categoría 1, 2 y 3 respectivamente. Los productos Leche evaporada Gloria y Yogurt Laive 1 litro es de la categoría Lácteos (código de categoría para Lácteos es 2) y el producto Cocacola 2 litros es de la categoría

Bebidas (código de categoría para Bebidas es 1). No hay por el momento ningún producto de la categoría condimentos. Justamente esto es lo que indica la cardinalidad 1 ------- 0…* se interpreta que una categoría puede tener 0 productos como es el caso de la categoría condimentos o puede tener 1 o más productos como es el caso de las categorías Bebidas y Lácteos (* representa 1 o más). Cabe señalar que el primer atributo de la clase Categoría y el último atributo de la clase Producto es la que establece la relación entre las clases.

Implementando las clases en programación de Java sería:

```
class Categoría

        {
    private int codcategoria;
    private String descripción;

    public Categoria(int codcat, String des)
    {
            codcategoria = codcat;
            descripción = des;
    }

    public void setCodCategoria(int codcat)
    {
    codcategoria =codcat
    }

    public int getCodCategoria()
    {
    return codcategoria;
    }
```

```
        .
        .
        .
       }

class Producto
        {
        private int codproducto;
        private String nombre;
        private int stock;
        private double preciounitario;
        private int codcategoria;

        public Producto(int codpro, String nom, int s, double pu,
                        int codcat)
        {
                codproducto = codpro;
                nombre = nom;
                stock = s;
                preciounitario = pu;
                codcategoria = codcat;
        }

        public void setCodProducto(int codpro)
        {
                codproducto = codpro;
        }

        public int getCodProducto()
        {
```

```
        return codproducto;
    }
    .
    .
    .
    }
```

Para implementar estas dos clases y poder manejar los datos expresados anteriormente n forma gráfica (3 objetos de la clase Categoría y 3 objetos de la clase de Producto) se debe programar lo siguiente:

```
public class ProbandoCategoriaProducto
{
    public static void main(String args[])throws IOException
    {
        BufferedReader leer=new BufferedReader(new
                        InputStreamReader(System.in));
        Categoria C1, C2, C3;
        Producto P1, P2, P3;
        int codcat, codpro, s;
        String des, nom;
        double pu;
        System.out.print("Codigo de Categoria: ");
        codcat = Integer.parseInt(leer.readLine());
        System.out.print("Descripción de la Categoria: ");
        des=leer.readLine();
        C1=new Categoria(codcat, des);
        System.out.print("Codigo de Categoria: ");
        codcat = Integer.parseInt(leer.readLine());
        System.out.print("Descripción de la Categoria: ");
```

```
des=leer.readLine();
C2=new Categoria(codcat, des);
System.out.print("Codigo de Categoria: ");
codcat = Integer.parseInt(leer.readLine());
System.out.print("Descripción de la Categoria: ");
des=leer.readLine();
C3=new Categoria(codcat, des);
System.out.print("Codigo de Producto: ");
codpro = Integer.parseInt(leer.readLine());
System.out.print("Nombre del Producto: ");
nom=leer.readLine();
System.out.print("Stock del Producto: ");
s = Integer.parseInt(leer.readLine());
System.out.print("Precio del Producto: ");
pu = Double.parseDouble(leer.readLine());
System.out.print("Codigo de Categoria: ");
codcat = Integer.parseInt(leer.readLine());
P1=new Producto(codpro, nom, s, pu, codcat);
System.out.print("Codigo de Producto: ");
codpro = Integer.parseInt(leer.readLine());
System.out.print("Nombre del Producto: ");
nom=leer.readLine();
System.out.print("Stock del Producto: ");
s = Integer.parseInt(leer.readLine());
System.out.print("Precio del Producto: ");
pu = Double.parseDouble(leer.readLine());
System.out.print("Codigo de Categoria: ");
codcat = Integer.parseInt(leer.readLine());
P2=new Producto(codpro, nom, s, pu, codcat);
System.out.print("Codigo de Producto: ");
```

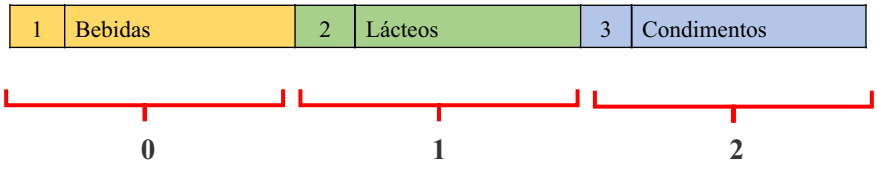
```
codpro = Integer.parseInt(leer.readLine());
System.out.print("Nombre del Producto: ");
nom=leer.readLine();
System.out.print("Stock del Producto: ");
s = Integer.parseInt(leer.readLine());
System.out.print("Precio del Producto: ");
pu = Double.parseDouble(leer.readLine());
System.out.print("Codigo de Categoria: ");
codcat = Integer.parseInt(leer.readLine());
P3=new Producto(codpro, nom, s, pu, codcat);
}
}
```

Resulta trabajoso ir creando variables de objetos en la programación según la necesidad lo que hace que el código de la programación se extienda y se haga a la medida, es por ello la necesidad de manejar los objetos dentro de arreglos.

6.3. Arreglo de objetos

La clase una vez definida y programada está lista para la creación de objetos. Un arreglo es una estructura de datos que a través de sus elementos guarda datos. Se pretende hacer que cada elemento del arreglo sea un objeto.

Arreglo: categorías



En Java se declara y se crea el arreglo:

Categoría categorías [] = new Categoría[3];

El arreglo categorías sus elementos tienen la estructura de la clase Categoría. Esta clase tiene dos atributos cod categoría y descripción, esto hace que cada elemento puede almacenar dos datos debido a los dos atributos. Una manera de almacenar los datos en el arreglo es:

Categoría C = new Categoría(1,"Bebidas");

Categorías [0]=C;

Se crea un objeto de la clase Categorías haciendo uso del método constructor dando como valores iniciales a los atributos 1 y Bebidas. El objeto C al tener los atributos con valores al momento de hacer mención de un elemento del arreglo se lo logra almacenar dicho objeto C en el elemento categorías [0].

6.4. Programas resueltos en Java usando NetBeans

1) Crear un programa que permita el ingreso de varias categorías y varios productos haciendo uso de las clases Categoría (atributos: codcategoría, descripción) y Producto (atributos: codproducto, nombre, stock, precio unitario, codcategoría). Luego muestre un listado de los productos y el producto de mayor stock.

```
3  import java.io.*;
4
5  class Categoria
6  {
7      private int codcategoria;
8      private String descripcion;
9
10     public Categoria(int codcat, String des)
11     {
12         codcategoria = codcat;
13         descripcion = des;
14     }
15
16     public void setCodCategoria(int codcat)
17     {
18         codcategoria=codcat;
19     }
20
21     public void setDescripcion(String des)
22     {
23         descripcion=des;
24     }
25
26     public int getCodCategoria()
27     {
28         return codcategoria;
29     }
30
```

```
31          public String getDescripcion()
32          {
33              return descripcion;
34          }
35
36      }
37
38      class Articulo
39      {
40          private int codproducto;
41          private String nombre;
42          private int stock;
43          private double preciounitario;
44          private int codcategoria;
45
46          public Articulo(int codpro, String nom, int s, double pu, int codcat)
47          {
48              codproducto=codpro;
49              nombre=nom;
50              stock=s;
51              preciounitario=pu;
52              codcategoria=codcat;
53          }
54
55          public void setCodProducto(int codpro)
56          {
57              codproducto=codpro;
58          }
59
60          public void setNombre(String nom)
61          {
62              nombre=nom;
63          }
64
65          public void setStock(int s)
66          {
67              stock=s;
68          }
69
70          public void setPrecioUnitario(double pu)
71          {
72              preciounitario=pu;
73          }
74
75          public void setCodCategoria(int codcat)
76          {
77              codcategoria=codcat;
78          }
79
80          public int getCodProducto()
81          {
82              return codproducto;
83          }
84
85          public String getNombre()
86          {
87              return nombre;
88          }
89
90          public int getStock()
91          {
92              return stock;
93          }
```

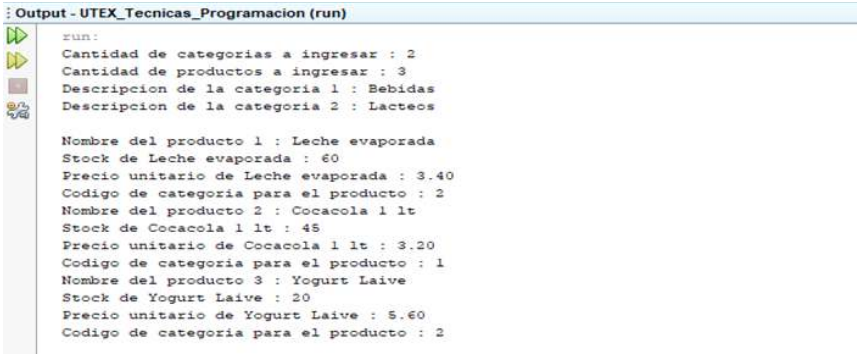
```
94
95          public double getPrecioUnitario()
96          {
97              return preciounitario;
98          }
99
100         public int getCodCategoria()
101         {
102             return codcategoria;
103         }
104     }
105
106     public class UsandoCategoriaProducto {
107
108         public static void imprime(String m)
109         {
110             System.out.print(m);
111         }
112
113         public static int leerDatoEntero()throws IOException
114         {
115             BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
116             return Integer.parseInt(leer.readLine());
117         }
118
119         public static String leerDatoTexto()throws IOException
120         {
121             BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
122             return leer.readLine();
123         }
124
125         public static double leerDatoReal()throws IOException
126         {
127             BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
128             return Double.parseDouble(leer.readLine());
129         }
130
131         public static String buscaCategoria(int c, Categoria cat[])
132         {
133             int i;
134             for(i=0;i<cat.length;i++)
135             {
136                 if(cat[i].getCodCategoria()==c)
137                 {
138                     return cat[i].getDescripcion();
139                 }
140             }
141             return "";
142         }
143
144         public static String productoMayorStock(Producto p[])
145         {
146             int i,pos=0, mayor=p[0].getStock();
147             for(i=1;i<p.length;i++)
148             {
149                 if(p[i].getStock()>mayor)
150                 {
151                     mayor=p[i].getStock();
152                     pos=i;
153                 }
154             }
155             return p[pos].getNombre();
156         }
157
```

```
158         public static void main(String args[])throws IOException
159         {
160             int n, i, s, c;
161             String des,nom;
162             double pu;
163 
164             do {
165                 System.out.print("Cantidad de categorias a ingresar : ");
166                 n = leerDatoEntero();
167             } while (n <= 0 || n > 10);
168 
169             Categoria categorias[] = new Categoria[n];
170 
171             do {
172                 System.out.print("Cantidad de productos a ingresar : ");
173                 n = leerDatoEntero();
174             } while (n <= 0 || n > 20);
175 
176             Producto productos[] = new Producto[n];
177 
178             for(i=0;i<categorias.length;i++)
179             {
180                 imprime("Descripcion de la categoria "+(i+1)+" : ");
181                 des=leerDatoTexto();
182                 Categoria C=new Categoria((i+1),des);
183                 categorias[i]=C;
184             }
185             imprime("\n");
186             for(i=0;i<productos.length;i++)
187             {
188                 imprime("Nombre del producto "+(i+1)+" : ");
189                 nom=leerDatoTexto();
190                 imprime("Stock de "+nom+" : ");
191                 s=leerDatoEntero();
192                 imprime("Precio unitario de "+nom+" : ");
193                 pu=leerDatoReal();
194                 imprime("Codigo de categoria para el producto : ");
195                 c=leerDatoEntero();
196                 Producto P=new Producto((i+1),nom,s,pu,c);
197                 productos[i]=P;
198             }
199             imprime("\n");
200             imprime("=====================================\n");
201             imprime("      LISTADO DE LOS PRODUCTOS       \n");
202             imprime("=====================================\n");
203             imprime("\n");
204             imprime("Codigo\tProducto\t\tStock\tPrecio\tCategoria\n");
205             for(i=0;i<productos.length;i++)
206             {
207                 imprime(productos[i].getCodProducto()+"\t"+productos[i].getNombre()+"\t\t"+
208                         productos[i].getStock()+"\t\t"+productos[i].getPrecioUnitario()+
209                         "\t"+buscaCategoria(productos[i].getCodCategoria(),categorias)+
210                         "\n");
211             }
212             imprime("\n");
213             imprime("El producto de mayor stock es "+productoMayorStock(productos)+"\n");
214 
215         }
216 
217     }
```

***Interpretación de la programación*:**

La programación se inicia con la creación de las clases Categoría y Producto con sus atributos y métodos. En la clase Usando Categoría Producto se define varios métodos, uno denominado imprime que sirve para imprimir una cadena de texto, otro denominado ler DatoEntero para la lectura de un valor numérico entero, otro denominado ler Dato Real para la lectura de un valor numérico real, otro denominado leerDatoTexto para la lectura de un dato de tipo cadena de texto, otro llamado buscaCategoria que consiste en buscar la descripción de la categoría dado el valor del código de la categoría y otro método llamado productoMayorStock que retorna el nombre del producto de mayor stock. En estos dos últimos métodos se hace uso de los métodos get como por ejemplo getCategoria() para obtener el código de la categoría del elemento especificado del arreglo. En el método main, después de declarar variables de memoria se pide el ingreso de la cantidad de categoría a almacenar como también la cantidad de productos a ingresar, se procede al ingreso de los datos en ambos arreglos con la ayuda de la sentencia repetitiva for donde además se crea objetos de la clase Categoría y de la Clase Producto que sirven para el llenado de cada elemento del arreglo. Luego se hace un listado de los productos mostrando en lugar del código de categoría la descripción de la categoría. Finalmente se muestra el producto de mayor stock.

Ejecución del programa:



```
=======================================
      LISTADO DE LOS PRODUCTOS
=======================================

Codigo  Producto                Stock   Precio  Categoria
1       Leche evaporada         60              3.4     Lacteos
2       Cocacola 1 lt           45              3.2     Bebidas
3       Yogurt Laive            20              5.6     Lacteos

El producto de mayor stock es Leche evaporada
BUILD SUCCESSFUL (total time: 1 minute 56 seconds)
```

2) Crear la clase Facultad con los atricutos codigofac y nombre. También crear la clase EscuelaProfesional con los atributos codigoesc, nombre, facultad. Lugo procede a ingresar los datos de dos escuelas profesionales.

```
3  import java.io.*;
4
5  class Facultad
6  {
7      String codigofac;
8      String nombre;
9
10     public Facultad(String cod, String nom)
11     {
12         codigofac=cod;
13         nombre=nom;
14     }
15 }
16
17 class EscuelaProfesional
18 {
19     String codigoesc;
20     String nombre;
21     Facultad facultad;
22
23     public EscuelaProfesional(String cod, String nom, Facultad f)
24     {
25         codigoesc=cod;
26         nombre=nom;
27         facultad=f;
28     }
29
30         public String toString()
31     {
32         return "La escuela profesional "+nombre+" pertenece a la facultad de "+
33                 facultad.nombre;
34     }
35 }
36
37 public class RelacionFacultadEscuela {
38
39     public static void imprime(String m)
40     {
41         System.out.print(m);
42     }
43
```

```
44      public static String leerDatoTexto()throws IOException
45      {
46          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
47          return leer.readLine();
48      }
49
50      public static void main(String[] args)throws IOException
51      {
52          String codesc, nomesc, codfac, nomfac;
53          int i;
54          imprime("=======================================\n");
55          imprime("DATOS DE LA PRIMERA ESCUELA PROFESIONAL\n");
56          imprime("=======================================\n");
57          imprime("Ingrese el codigo de la escuela profesional : ");
58          codesc=leerDatoTexto();
59          imprime("Ingrese el nombre de la escuela profesional : ");
60          nomesc=leerDatoTexto();
61          imprime("Ingrese el codigo de facultad : ");
62          codfac=leerDatoTexto();
63          imprime("Ingrese el nombre de facultad : ");
64          nomfac=leerDatoTexto();
65          Facultad F1=new Facultad(codfac,nomfac);
66          EscuelaProfesional EP1=new EscuelaProfesional(codesc,nomesc,F1);
67          imprime("\n");
68          imprime("=======================================\n");
69          imprime("DATOS DE LA SEGUNDA ESCUELA PROFESIONAL\n");
70          imprime("=======================================\n");
71          imprime("Ingrese el codigo de la escuela profesional : ");
72          codesc=leerDatoTexto();
73          imprime("Ingrese el nombre de la escuela profesional : ");
74          nomesc=leerDatoTexto();
75          imprime("Ingrese el codigo de facultad : ");
76          codfac=leerDatoTexto();
77          imprime("Ingrese el nombre de facultad : ");
78          nomfac=leerDatoTexto();
79          Facultad F2=new Facultad(codfac,nomfac);
80          EscuelaProfesional EP2=new EscuelaProfesional(codesc,nomesc,F2);
81          imprime("\n");
82          imprime(EP1+"\n");
83          imprime(EP2+"\n");
84      }
85
86  }
```

***Interpretación de la programación*:**

La aplicación se inicia con la creación de la clase Facultad y la clase EscuelaProfesional. La novedad en esta programación está en el tercer atributo de la clase EscuelaProfesional es de tipo clase Facultad, esta otra forma de relacionar una clase con otra. En el método constructor EscuelaProfesional, el tercer parámetro que es f es de tipo Facultad. El método toString establece una cadena de texto que indica el nombre de la escuela profesional y a qué facultad pertenece. En la clase RelacionFacultadEscuela se establece los métodos imprime y leerDatoTexto. En el método main se solicita el ingreso de datos de dos escuelas profesionales indicando a qué facultad pertenece y termina con la impresión de los datos ingresados.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
=======================================
DATOS DE LA PRIMERA ESCUELA PROFESIONAL
=======================================
Ingrese el codigo de la escuela profesional : 001
Ingrese el nombre de la escuela profesional : Ingenieria de Sistemas
Ingrese el codigo de facultad : 01
Ingrese el nombre de facultad : Ingenieria

=======================================
DATOS DE LA SEGUNDA ESCUELA PROFESIONAL
=======================================
Ingrese el codigo de la escuela profesional : 0002
Ingrese el nombre de la escuela profesional : Odontologia
Ingrese el codigo de facultad : 02
Ingrese el nombre de facultad : Medicina

La escuela profesional Ingenieria de Sistemas pertenece a la facultad de Ingenieria
La escuela profesional Odontologia pertenece a la facultad de Medicina
```

3) Crear una aplicación que permita registrar datos de artefactos a partir del diseño de dos clases: Marca (atributos: codmarca, nombre) y Artefacto (atributos: codartefacto, descripción, precio, codmarca). Luego listar los artefactos y posteriormente dado el nombre de la marca listar los artefactos de la marca indicada.

Crear el archivo LecturaDatos.java que contenga:

```
3  import java.io.*;
4
5  public class LecturaDatos {
6
7      public static int leerDatoEntero()throws IOException
8      {
9          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
10         return Integer.parseInt(leer.readLine());
11     }
12
13     public static String leerDatoTexto()throws IOException
14     {
15         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
16         return leer.readLine();
17     }
18
19     public static double leerDatoReal()throws IOException
20     {
21         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
22         return Double.parseDouble(leer.readLine());
23     }
24
25 }
```

Crear el archivo Marca.java que contenga:

```
public class Marca {
    private int codmarca;
    private String nombre;

    public Marca(int cod, String n)
    {
        codmarca=cod;
        nombre=n;
    }

    public void setCodMarca(int cod)
    {
        codmarca=cod;
    }

    public void setNombre(String n)
    {
        nombre=n;
    }

    public int getCodMarca()
    {
        return codmarca;
    }

    public String getNombre()
    {
        return nombre;
    }
}
```

Crear el archivo Artefacto.java que contenga

```
public class Artefacto {

    private int codartefacto;
    private String descripcion;
    private double precio;
    private int codmarca;

    public Artefacto(int coda, String d, double p, int codm)
    {
        codartefacto=coda;
        descripcion=d;
        precio=p;
        codmarca=codm;
    }

    public void setCodProducto(int coda)
    {
        codartefacto=coda;
    }

    public void setDescripcion(String d)
    {
        descripcion=d;
    }

    public void setPrecio(double p)
    {
        precio=p;
    }
```

```
32
33      public void setCodMarca(int codm)
34      {
35          codmarca=codm;
36      }
37
38      public int getCodArtefacto()
39      {
40          return codartefacto;
41      }
42
43      public String getDescripcion()
44      {
45          return descripcion;
46      }
47
48      public double getPrecio()
49      {
50          return precio;
51      }
52
53      public int getCodMarca()
54      {
55          return codmarca;
56      }
57
58  }
```

Crear el archivo TiendaArtefactos que contenga:

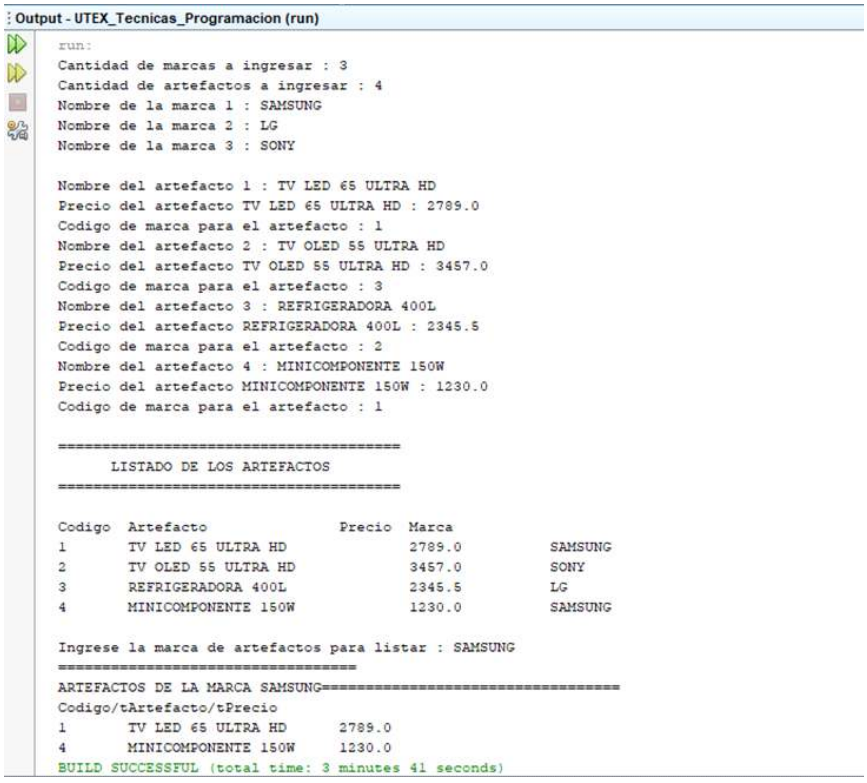
```
3   import java.io.*;
4
5   public class TiendaAretefactos {
6
7       public static void imprime(String m)
8       {
9           System.out.print(m);
10      }
11
12      public static String buscaMarca(int c, Marca mar[])
13      {
14          int i;
15          for(i=0;i<mar.length;i++)
16          {
17              if(mar[i].getCodMarca()==c)
18              {
19                  return mar[i].getNombre();
20              }
21          }
22          return "";
23      }
24
25      public static void listarArtefactos(String des, Artefacto art[], Marca mar[])
26      {
27          int i, cod=0;
28          imprime("================================\n");
29          imprime("ARTEFACTOS DE LA MARCA "+des);
30          imprime("================================\n");
31          for(i=0;i<mar.length;i++)
32          {
33              if(mar[i].getNombre().equalsIgnoreCase(des))
34              {
35                  cod=mar[i].getCodMarca();
36                  break;
37              }
38          }
```

```
39              imprime("Codigo/tArtefacto/tPrecio\n");
40              for(i=0;i<art.length;i++)
41              {
42                  if(cod==art[i].getCodMarca())
43                      imprime(art[i].getCodArtefacto()+"\t"+art[i].getDescripcion()+
44                          "\t"+art[i].getPrecio()+"\n");
45              }
46
47          }
48
49          public static void main(String args[])throws IOException
50          {
51              int i, c, n;
52              String nom, m;
53              double p;
54
55                  do {
56                  System.out.print("Cantidad de marcas a ingresar : ");
57                  n = LecturaDatos.leerDatoEntero();
58              } while (n <= 0 || n > 10);
59
60              Marca marcas[] = new Marca[n];
61
62              do {
63                  System.out.print("Cantidad de artefactos a ingresar : ");
64                  n = LecturaDatos.leerDatoEntero();
65              } while (n <= 0 || n > 20);
66
67              Artefacto artefactos[] = new Artefacto[n];
68
69              for(i=0;i<marcas.length;i++)
70              {
71                  imprime("Nombre de la marca "+(i+1)+" : ");
72                  nom=LecturaDatos.leerDatoTexto();
73                  Marca C=new Marca((i+1),nom);
74                  marcas[i]=C;
75              }
76              imprime("\n");
77              for(i=0;i<artefactos.length;i++)
78              {
79                  imprime("Nombre del artefacto "+(i+1)+" : ");
80                  nom=LecturaDatos.leerDatoTexto();
81                  imprime("Precio del artefacto "+nom+" : ");
82                  p=LecturaDatos.leerDatoReal();
83                  imprime("Codigo de marca para el artefacto : ");
84                  c=LecturaDatos.leerDatoEntero();
85                  Artefacto A=new Artefacto((i+1),nom,p,c);
86                  artefactos[i]=A;
87              }
88
89              imprime("\n");
90              imprime("=====================================\n");
91              imprime("      LISTADO DE LOS ARTEFACTOS       \n");
92              imprime("=====================================\n");
93              imprime("\n");
94              imprime("Codigo\tArtefacto\t\tPrecio\tMarca\n");
95              for(i=0;i<artefactos.length;i++)
96              {
97                  imprime(artefactos[i].getCodArtefacto()+"\t"+artefactos[i].getDescripcion()+
98                          "\t\t"+artefactos[i].getPrecio()+"\t\t"+
99                          buscaMarca(artefactos[i].getCodMarca(),marcas)+"\n");
100             }
101             imprime("\n");
102             imprime("Ingrese la marca de artefactos para listar : ");
103             m=LecturaDatos.leerDatoTexto();
104             listarArtefactos(m, artefactos, marcas);
105         }
106     }
```

***Interpretación de la programación*:**

Esta es la primera aplicación donde el código de la programación es distribuido en varios archivos de extensión java, es decir, en varias clases. Se crea una clase LecturaDatos para programar métodos que permiten la lectura de datos desde el teclado. Luego, se crea las clases Marca y Artefacto con los atributos indicados en el enunciado y finalmente se tiene la clase TiendaArtefactos que solicita el ingreso de una cantidad de marcas y de artefactos para luego listar los artefactos y posteriormente, se pide el ingreso del nombre de una marca para luego mostrar los artefactos de la marca indicado.

Ejecución del programa:



```
Output - UTEX_Tecnicas_Programacion (run)
run:
Cantidad de marcas a ingresar : 3
Cantidad de artefactos a ingresar : 4
Nombre de la marca 1 : SAMSUNG
Nombre de la marca 2 : LG
Nombre de la marca 3 : SONY

Nombre del artefacto 1 : TV LED 65 ULTRA HD
Precio del artefacto TV LED 65 ULTRA HD : 2789.0
Codigo de marca para el artefacto : 1
Nombre del artefacto 2 : TV OLED 55 ULTRA HD
Precio del artefacto TV OLED 55 ULTRA HD : 3457.0
Codigo de marca para el artefacto : 3
Nombre del artefacto 3 : REFRIGERADORA 400L
Precio del artefacto REFRIGERADORA 400L : 2345.5
Codigo de marca para el artefacto : 2
Nombre del artefacto 4 : MINICOMPONENTE 150W
Precio del artefacto MINICOMPONENTE 150W : 1230.0
Codigo de marca para el artefacto : 1

=====================================
      LISTADO DE LOS ARTEFACTOS
=====================================

Codigo  Artefacto              Precio  Marca
1       TV LED 65 ULTRA HD             2789.0          SAMSUNG
2       TV OLED 55 ULTRA HD            3457.0          SONY
3       REFRIGERADORA 400L             2345.5          LG
4       MINICOMPONENTE 150W            1230.0          SAMSUNG

Ingrese la marca de artefactos para listar : SAMSUNG
==================================
ARTEFACTOS DE LA MARCA SAMSUNG==================================
Codigo/tArtefacto/tPrecio
1       TV LED 65 ULTRA HD      2789.0
4       MINICOMPONENTE 150W     1230.0
BUILD SUCCESSFUL (total time: 3 minutes 41 seconds)
```

7. MODIFICADORES DE ACCESO A LOS MIEMBROS DE UNA CLASE

7.1. ¿Qué son modificadores de acceso?

Los modificadores de acceso son palabras reservadas del lenguaje de programación Java que define la forma de cómo acceder a las clases y a los componentes de las mismas, es decir, determinan desde qué clases se puede acceder a un determinado miembro de la clase o de la clase misma.

7.2. Tipos de modificadores

En Java se tiene 4 tipos: public, private, protected y default, este último no tiene ninguna palabra clave asociada. Por lo tanto, si no especificamos ningún modificador de acceso se utiliza el nivel de acceso por defecto, que consiste en que la clase o miembro de la clase puede ser accedido sólo desde las clases que pertenezcan al paquete.

7.2.1. Modificador public

El modificador de acceso public permite acceder a cualquier clase o miembro de una clase desde cualquier clase. Las clases, los atributos y métodos que hagan uso de public serán accesibles de cualquier lugar de la aplicación programada.

Por ejemplo:

```
package p01;
public class Matematicas
        i{
        public int suma(int x,int y)
        {
                return x+y;
        }

        public int resta(int x, int y)
        {
                return x-y;
        }
        }
```

En el programa anterior que define la clase pública Matematicas con la palabra reservada *public* y los métodos suma y resta también tienen un nivel de acceso *public*. Esta clase Matematicas se encuentra en el paquete denominado p01.

```
package p02;

import java.io.*;
import paquete01.Matematicas;

public class Operaciones {

  public static void mensaje(String m)
  {
    System.out.print(m);
  }
```

```
public static int leerDatoEntero()throws IOException
{
    BufferedReader leer = new BufferedReader(new
                                InputStreamReader(System.in));
    return Integer.parseInt(leer.readLine());
}

public static void main(String args[])throws IOException
{
    int n1, n2;
    mensaje("Ingrese el primer número : ");
    n1=leerDatoEntero();
    mensaje("Ingrese el segundo número : ");
    n2=leerDatoEntero();
    Matematicas M=new Matemáticas();
    mensaje("La suma de los números es : "+M.suma(n1, n2)+"\n");
    mensaje("La resta de los números es : "+M.resta(n1, n2)+"\n");
}
}
```

La clase pública Operaciones se encuentra dentro del paquete denominado paquete02, al momento de hacer uso de instrucción: *import p01.Matematicas;* esto permite hacer uso de la clase Matematicas que se encuentra en el paquete p01 dado que esta clase es pública. En la instrucción *Matematicas M=new Matematicas();* se puede crear un objeto M de la clase Matematica que está en el otro paquete llamado p01.

7.2.2. Modificador private

El modificador private tiene bastante restricción y sólo permite a los miembros de la clase que se pueda usar dentro de la clase donde se define, es decir, se puede acceder desde la clase donde fueron declarados o creados. Cualquier otra clase que esté dentro de un paquete no podrá acceder a los miembros o elementos privados de otra clase del mismo paquete. Cabe señalar que las clases y las interfases no se pueden dar un nivel de acceso de tipo private.

Por ejemplo:

```
package p01;
class A
{
        private void imprime(String m)
        {
                System.out.println(m);
        }
}

public class B
{
        public static void main(String args[])
        {
                A objeto=new A();
                //tratando de acceder al metodo privado imprime que está en A
                Objeto.imprime("Esto es fácil");
        }
}
```

Al momento de ejecutar el programa presenta un error en la instrucción objeto.imprime(“ Esto es fácil”); debido a que el método imprime es de acceso privado en la clase A y solo puede ser usado dentro de la clase A y no puede compartir el método imprime a la clase B y a ninguna clase distinta de A. Para que ejecute correctamente el programa anterior el método imprime debe ser public.

7.2.3. Modificador protected

El modificador protected consiste en que los atributos y métodos declarados dentro de la clase son accesible dentro del mismo paquete y desde otra clase que haya sido extendido o heredada de la clase que contiene a los elementos de acceso protected, es decir, que una subclase puede estar en otro paquete y tiene la posibilidad de hacer uso de atributos y métodos de nivel de acceso protected de un clase que esté en otro paquete. Por ejemplo:

```
package p01;
public class A
{
        protected void imprime(String m)
        {
                System.out.println(m);
        }
}

package p02;
import p01.*;
public class B extends A
{
```

```
        public static void main(String args[])
        {
                B objeto=new B();
                objeto.imprime("Haciendo uso del modificador
                                protected");
        }
}
```

Se crea un método imprime en la clase A de nivel de acceso protected. Tener en cuenta que la clase A está en el paquete p01. En el paquete p02 se importa la clase A del paquete p01 con la instrucción import p01.*; la clase B es extendido de A a través del uso de la palabra reservada extend, esto quiere decir que la clase B recibe de A el método imprime y lo hace como suyo. Es por esto que luego de crear el objeto a partir de la clase B se puede escribir la instrucción objeto.imprime("Haciendo uso del modificador protected");

7.2.4. Modificador default

El modificador default es cuando no especificas ninguna expresión al momento de crear la clase o los miembros de una clase. La característica más importante es que pueden ser accedidos dentro un mismo paquete. Por ejemplo:

```
package p01;
class A
{
        void imprime(String m)
        {
                System.out.println(m);
        }
}
```

```
package p02;
public class B
{
        public static void main(String args[])
        {
        A objeto=new A();
        objeto.imprime("Esto no funciona");
        }
}
```

La clase A tiene nivel de acceso default dado que no se especifica expresión alguna antes de la palabra class y se encuentra en el paquete p01. Se pretende en el paquete p02 hacer uso del método imprime que también tiene de nivel acceso default. El programa al momento de ser ejecutado no podrá hacer por el simple hecho de que se quiere usar cuando se tiene programado en dos paquetes distintos.

Los distintos modificadores de acceso quedan resumidos en la siguiente tabla:

Nivel de Acceso	Clase	Paquete	Subclase	Todos
Public	✓	✓	✓	✓
protected	✓	✓	✓	x
default	✓	✓	x	x
private	✓	x	x	x

7.3. Palabras reservadas para uso de variables

7.3.1. Static

La palabra reserva static no es para crear datos contantes sino para hacer que los miembros de una clase sólo pertenezcan a la clase y no ser usadas en las instancias de la clase. Esto quiere decir que si los miembros de una clase son todos static no es necesario crear un objeto para poder acceder a los atributos y métodos de la clase. El método main que busca el compilador de Java es de tipo static y no se necesita crear un objeto para su ejecución. Por ejemplo, crea una clase A en un archivo A.java de la siguiente manera:

```
package p01;
public class A
{
        static int cuenta=0;
        public A()
        {
                cuenta=cuenta+1;
        }
}
```

Se puede apreciar que el atributo cuenta es de tipo entero y es static con un valor inicial de 0. En el método constructor A() la variable o atributo cuenta se incrementa en 1, esto quiere decir que cada vez que se cree un objeto a partir de la clase A el valor del atributo se incrementará en 1. Luego procede a crear la clase UsandoA de la siguiente manera:

```
package p01;
public class B
{
```

```
        public static void main(String args[])
        {
                A objeto1 = new A();
                A objeto2 = new A();
                System.out.println("Se ha creado
                                "+objeto1.cuenta+" objetos");
        }
}
```

En la clase B, en el método main se crea dos objetos y si se desea imprimir el contenido del atributo cuenta devolverá el valor de 2 dado que se han creado 2 objetos. Da lo mismo colocar en la programación objeto1.cuenta que poner objeto2.cuenta como también A.cuenta

7.3.2. Final

Una variable o atributo, método o clase no se podrá modificar su contenido. Si un atributo o variable se le coloca la palabra reservada final no se puede asignarle otro valor. Si una clase se le pone la palabra reservada final no podrá extender la clase, es decir, no podrá dar herencia a una nueva clase. Si a un método se le crea con la palabra reservada final no se podrá sobrescribir sobre él. Por ejemplo, lo que se programa a continuación no funcionaría.

```
public class UsandoFinal
{
        public static void main(String args[])
        {
                final String mensaje = "Perú es una linda tierra";
                mensaje = "Trujillo es la ciudad de la eterna
                                primavera";
        }
}
```

Este programa no funcionaría porque se pretende cambiar el contenido del mensaje. Lo que viene a continuación si funcionaría.

```
public class UsandoFinal
{
        public static void main(String args[])
        {
                String nuevomensaje;
                final String mensaje = "Perú es una linda tierra";
                nuevomensaje = mensaje + " y de gente muy linda";
        }
}
```

7.4. Programas resueltos en Java usando NetBeans

1) Crear una clase Trabajador que haga uso del modificador static y luego crea dos objetos de la clase Trabajador.

Se usa la programación hecha en la clase LecturaDatos que se encuentra en el paquete Capitulo_I.Sesion_06

```
1  package Capitulo_I.Sesion_06;
2
3  import java.io.*;
4
5  public class LecturaDatos {
6
7      public static int leerDatoEntero()throws IOException
8      {
9          BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
10         return Integer.parseInt(leer.readLine());
11     }
12
13     public static String leerDatoTexto()throws IOException
14     {
15         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
16         return leer.readLine();
17     }
18
19     public static double leerDatoReal()throws IOException
20     {
21         BufferedReader leer = new BufferedReader(new InputStreamReader(System.in));
22         return Double.parseDouble(leer.readLine());
23     }
24
25 }
```

En el archivo Trabajador.java se programa lo siguiente:

```java
package Capitulo_I.Sesion_07;

public class Trabajador {

    private static int contador;
    private String nombres;
    private String apellidos;

    public Trabajador(String nom, String ape) {
        contador++;
        nombres = nom;
        apellidos = ape;
    }

    public void finalize() {
        --contador;
    }

    public void setNombres(String nom) {
        nombres = nom;
    }

    public void setApellidos(String ape) {
        apellidos = ape;
    }

    public static void setContador(int cont) {
        contador = cont;
    }

    public String getNombres() {
        return nombres;
    }

    public String getApellidos() {
        return apellidos;
    }

    public static int getContador() {
        return contador;
    }

    public String toString() {
        return "Trabajador(a) : "+apellidos + " " + nombres;
    }

}
```

En el archivo ProbandoTrabajador se programa lo siguiente:

```java
package Capitulo_I.Sesion_07;

import java.io.*;
import Capitulo_I.Sesion_06.LecturaDatos;

public class ProbandoTrabajador {

    public static void imprime(String m)
    {
        System.out.print(m);
    }

```

```
13      public static void main(String args[]) throws IOException
14      {
15          String nom, ape;
16          imprime("Cantidad de objetos creados : "+Trabajador.getContador()+"\n");
17          imprime("\n");
18          imprime("=====================\n");
19          imprime("  PRIMER TRABAJADOR  \n");
20          imprime("=====================\n");
21          imprime("Nombres : ");
22          nom=LecturaDatos.leerDatoTexto();
23          imprime("Apellidos : ");
24          ape=LecturaDatos.leerDatoTexto();
25          Trabajador T1 = new Trabajador(nom, ape);
26          imprime(T1+"\n");
            imprime("Cantidad de objetos creados : " + T1.getContador()+"\n");
28          imprime("\n");
29          imprime("=====================\n");
30          imprime(" SEGUNDO TRABAJADOR  \n");
31          imprime("=====================\n");
32          imprime("Nombres : ");
33          nom=LecturaDatos.leerDatoTexto();
34          imprime("Apellidos : ");
35          ape=LecturaDatos.leerDatoTexto();
36          Trabajador T2 = new Trabajador(nom, ape);
37          imprime(T2+"\n");
            imprime("Cantidad de objetos creados : " + T2.getContador()+"\n");
39      }
40
41  }
```

***Interpretación de la programación*:**

En la clase Trabajador se hace uso de un atributo private static y un método public static. El atributo contador se inicializa con un valor igual cero. Este atributo va contando el número de Objetos de la Clase Trabajador en la medida en que se van creando los objetos, es decir, se va incrementando en el constructor cada vez que se crea un objeto. Para saber en un momento dado cuántos objetos de la clase Trabajador se ha creado basta con ejecutar el método estático getContador() que devuelve el valor de la variable contador. Un método definido con static no puede acceder a miembros de una clase que no usa la palabra reservada static. La aplicación hace uso de 3 clases: la clase LecturaDatos que tiene los métodos para leer desde el teclado, la clase Trabajador donde se declara el atributo contador como entero y como estáticos y los atributos nombres y apellidos, aparte de los métodos set y get y el método toString. La tercera clase es ProbandoTrabajador que en la línea 4 de la programación se importa el paquete que contiene a la clase LecturaDatos para ser usado en el método main. En la medida que se crea un objeto a partir de la clase Trabajador la variable entera y estática llamada contador se incrementa en 1.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Cantidad de objetos creados : 0

=====================
  PRIMER TRABAJADOR
=====================
Nombres : Manuel Miguel
Apellidos : Sandoval Vela
Trabajador(a) : Sandoval Vela Manuel Miguel
Cantidad de objetos creados : 1

=====================
 SEGUNDO TRABAJADOR
=====================
Nombres : Luisa Fernanda
Apellidos : Zavaleta Suarez
Trabajador(a) : Zavaleta Suarez Luisa Fernanda
Cantidad de objetos creados : 2
BUILD SUCCESSFUL (total time: 43 seconds)
```

2) Crea una clase llamada Articulo con los atributos: codarticulo, descripcion, cantidad y preciounitario. Crea un programa que permita el ingreso de n artículos donde el código de articulo se genere a partir de una variable estática. Además se muestre el listado de los artículos ordenados por de mayor a menor con respecto al precio unitario.

La programación para la clase Articulo es como sigue.

```
1  package Capitulo_I.Sesion_07;
2
3  public class Articulo {
4
5      private static int cuenta=0;
6      private int codarticulo;
7      private String descripcion;
8      private int cantidad;
9      private double preciounitario;
10
11     public Articulo(String des, int cant, double pu)
12     {
13         cuenta++;
14         codarticulo=cuenta;
15         descripcion=des;
16         cantidad=cant;
17         preciounitario=pu;
18     }
19
20     public void setCodArticulo(int c)
21     {
22         codarticulo=c;
23     }
24
25     public void setDescripcion(String des)
26     {
27         descripcion=des;
28     }
29
```

```
30      public void setCantidad(int cant)
31      {
32          cantidad=cant;
33      }
34
35      public void setPrecioUnitario(double pu)
36      {
37          preciounitario=pu;
38      }
39
40      public int getCodArticulo()
41      {
42          return codarticulo;
43      }
44
45      public String getDescripcion()
46      {
47          return descripcion;
48      }
49
50      public int getCantidad()
51      {
52          return cantidad;
53      }
54
55      public double getPrecioUnitario()
56      {
57          return preciounitario;
58      }
59  }
```

La programación de la clase UsandoArtículo es como sigue

```
1   package Capitulo_I.Sesion_07;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   public class UsandoArticulo {
7
8       public static void imprime(String m)
9       {
10          System.out.print(m);
11      }
12
13      public static void leerDatosArticulos(Articulo x[])throws IOException
14      {
15          int i,cant;
16          String des;
17          double pu;
18          for(i=0;i<x.length;i++)
19          {
20              imprime("Descripcion del articulo "+(i+1)+" : ");
21              des=LecturaDatos.leerDatoTexto();
22              imprime("Cantidad del articulo "+(i+1)+" : ");
23              cant=LecturaDatos.leerDatoEntero();
24              imprime("Precio unitario del articulo "+(i+1)+" : ");
25              pu=LecturaDatos.leerDatoReal();
26              Articulo A=new Articulo(des,cant,pu);
27              x[i]=A;
```

```
28              imprime("\n");
29          }
30      }
31
32      public static void listarArtculosOrdenadosPrecio(Articulo x[])
33      {
34          int i, j, auxI;
35          String auxS;
36          double auxD;
37          for(i=2;i<=x.length;i++)
38              for(j=0;j<=x.length-i;j++)
39              {
40                  if(x[j].getPrecioUnitario()<x[j+1].getPrecioUnitario())
41                  {
42                      auxS=x[j].getDescripcion();
43                      x[j].setDescripcion(x[j+1].getDescripcion());
44                      x[j+1].setDescripcion(auxS);
45                      auxI=x[j].getCantidad();
46                      x[j].setCantidad(x[j+1].getCantidad());
47                      x[j+1].setCantidad(auxI);
48                      auxD=x[j].getPrecioUnitario();
49                      x[j].setPrecioUnitario(x[j+1].getPrecioUnitario());
50                      x[j+1].setPrecioUnitario(auxD);
51                  }
52              }
53          imprime("=======================================\n");
54          imprime("LISTADO DE ARTICULOS ORDENADOS POR PRECIO\n");
55          imprime("=======================================\n");
56          imprime("\n");
57          imprime("Codigo\tDescripcion\t\tCantidad\tPrecio\n");
58          for(i=0;i<x.length;i++)
59          {
60              imprime(x[i].getCodArticulo()+"\t"+x[i].getDescripcion()+"\t\t"+
61                      x[i].getCantidad()+"\t\t"+x[i].getPrecioUnitario()+"\n");
62          }
63      }
64
65      public static void main(String atgs[])throws IOException
66      {
67          int n;
68          do {
69              System.out.print("Cantidad de articulos a ingresar : ");
70              n = LecturaDatos.leerDatoEntero();
71          } while (n <= 0 || n > 100);
72
73          Articulo articulos[]=new Articulo[n];
74
75          leerDatosArticulos(articulos);
76
77          listarArtculosOrdenadosPrecio(articulos);
78
79      }
80  }
```

***Interpretación de la programación*:**

Se crea la clase Articulo con los atributos indicados en el enunciado. Fue necesario crear una variable estática entera llamada cuenta para ir generando los nuevos códigos de los artículos en la medida que se iba creando más artículos. En la programación de la clase UsandoArtículo en la línea 4 se importa el paquete que contiene la clase LecturaDatos.

Se vio la necesidad de crear un método para el llenado del arreglo de n artículos y otro método para la ordenación de mayor a menor con respecto al precio unitario para luego listar los datos de todos los artículos. Para la ordenación de los datos se hace del método de la burbuja.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
Cantidad de articulos a ingresar : 3
Descripcion del articulo 1 : Aceite Primor 1L
Cantidad del articulo 1 : 40
Precio unitario del articulo 1 : 7.80

Descripcion del articulo 2 : Yogurt Gloria 1L
Cantidad del articulo 2 : 30
Precio unitario del articulo 2 : 5.80

Descripcion del articulo 3 : Cocacola 1 litro
Cantidad del articulo 3 : 25
Precio unitario del articulo 3 : 3.30

========================================
LISTADO DE ARTICULOS ORDENADOS POR PRECIO
========================================

Codigo  Descripcion             Cantidad        Precio
1       Aceite Primor 1L                40              7.8
2       Yogurt Gloria 1L                30              5.8
3       Cocacola 1 litro                25              3.3
BUILD SUCCESSFUL (total time: 1 minute 19 seconds)
```

3) Crea una clase para manejar algunas figuras geométricas y calcula el área y el perímetro del cuadrado, rectángulo y del triángulo. Hacer uso de métodos estáticos y métodos no estáticos.

Respuesta con métodos estáticos:

```
1   package Capitulo_I.Sesion_07;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   class FigurasGeometricas_E
7   {
8       public static double areaCuadrado(int lado1) {
9           return lado1 * lado1;
10      }
11
12      public static double areaRectangulo(int lado1, int lado2) {
13          return lado1 * lado2;
14      }
15
```

```
16      public static double areaTriangulo(int lado1, int lado2, int lado3) {
17          double p, r;
18          p = (lado1 + lado2 + lado3) / 2.0;
19          r = p * (p - lado1) * (p - lado2) * (p - lado3);
20          if (r > 0) {
21              return Math.sqrt(r);
22          } else {
23              return 0.0;
24          }
25      }
26
27      public static int perimetroCuadrado(int lado1) {
28          return 4 * lado1;
29      }
30
31      public static int perimetroRectangulo(int lado1, int lado2) {
32          return 2 * lado1 + 2 * lado2;
33      }
34
35      public static int perimetroTriangulo(int lado1, int lado2, int lado3) {
36          return lado1 + lado2 + lado3;
37      }
38  }
39
40  public class CalculoAreaPerimetro_Estaticos {
41      public static void imprime(String m)
42      {
43          System.out.print(m);
44      }
45
46      public static void main(String args[])throws IOException
47      {
48          int opcion, l1, l2, l3;
49          imprime("CALCULO DEL AREA Y PERIMETRO\n");
50          imprime("Ingrese 1: cuadrado, 2:rectangulo, 3:triangulo -> ");
51          opcion = LecturaDatos.leerDatoEntero();
52          if (opcion == 1) {
53              imprime("Ingrese el valor del lado del cuadrado: ");
54              l1 = LecturaDatos.leerDatoEntero();
55              imprime("El area del cuadrado es: " +
56                      FigurasGeometricas_E.areaCuadrado(l1)+"\n");
57              imprime("El perimetro del cuadrado es: " +
58                      FigurasGeometricas_E.perimetroCuadrado(l1)+"\n");
59          }
60          if (opcion == 2) {
61              imprime("Ingrese el valor del lado 1 del rectangulo: ");
62              l1 = LecturaDatos.leerDatoEntero();
63              imprime("Ingrese el valor del lado 2 del rectangulo: ");
64              l2 = LecturaDatos.leerDatoEntero();
65              imprime("El area del rectangulo es: " +
66                      FigurasGeometricas_E.areaRectangulo(l1, l2)+"\n");
67              imprime("El perimetro del rectangulo es: " +
68                      FigurasGeometricas_E.perimetroRectangulo(l1, l2)+"\n");
69          }
70          if (opcion == 3) {
71              imprime("Ingrese el valor del lado 1 del triangulo: ");
72              l1 = LecturaDatos.leerDatoEntero();
73              System.out.print("Ingrese el valor del lado 2 del triangulo: ");
```

```
74                  l2 = LecturaDatos.leerDatoEntero();
75                  System.out.print("Ingrese el valor del lado 3 del triangulo: ");
76                  l3 = LecturaDatos.leerDatoEntero();
77                  imprime("El area del triangulo es: " +
78                          FigurasGeometricas_E.areaTriangulo(l1, l2, l3)+"\n");
79                  imprime("El perimetro del rectangulo es: " +
80                          FigurasGeometricas_E.perimetroTriangulo(l1, l2, l3)+"\n");
81              }
82          }
83      }
```

Ejecución del programa con métodos estáticos:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
CALCULO DEL AREA Y PERIMETRO
Ingrese 1: cuadrado, 2:rectangulo, 3:triangulo -> 2
Ingrese el valor del lado 1 del rectangulo: 20
Ingrese el valor del lado 2 del rectangulo: 12
El area del rectangulo es: 240.0
El perimetro del rectangulo es: 64
BUILD SUCCESSFUL (total time: 9 seconds)
```

Respuesta con métodos no estáticos:

```
1   package Capitulo_I.Sesion_07;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   class FigurasGeometricas_NE
7   {
8       public double areaCuadrado(int lado1)
9       {
10          return lado1*lado1;
11      }
12
13      public double areaRectangulo(int lado1, int lado2)
14      {
15          return lado1*lado2;
16      }
17
18      public double areaTriangulo(int lado1, int lado2, int lado3)
19      {
20          double p, r;
21          p=(lado1+lado2+lado3)/2.0;
22          r=p*(p-lado1)*(p-lado2)*(p-lado3);
23          if (r>0)
24              return Math.sqrt(r);
25          else
26              return 0.0;
27      }
28
29      public int perimetroCuadrado(int lado1)
30      {
31          return 4*lado1;
32      }
33
34      public int perimetroRectangulo(int lado1, int lado2)
35      {
36          return 2*lado1+2*lado2;
37      }
```

```
38
39      public int perimetroTriangulo(int lado1, int lado2, int lado3)
40      {
41          return lado1+lado2+lado3;
42      }
43  }
44
45
46  public class CalculoAreaPerimetro_No_Estaticos {
47      public static void mensaje(String m)
48      {
49          System.out.print(m);
50      }
51
52      public static void main(String args[])throws IOException
53      {
54          int opcion, l1,l2,l3;
55          FigurasGeometricas_NE F=new FigurasGeometricas_NE();
56          mensaje("CALCULO DEL AREA Y PERIMETRO\n");
57          mensaje("Ingrese 1: cuadrado, 2:rectangulo, 3:triangulo -> ");
58          opcion=LecturaDatos.leerDatoEntero();
59          if (opcion==1)
60          {
61              mensaje("Ingrese el valor del lado del cuadrado: ");
62              l1=LecturaDatos.leerDatoEntero();
63              mensaje("El area del cuadrado es: "+F.areaCuadrado(l1)+"\n");
64              mensaje("El perimetro del cuadrado es: "+F.perimetroCuadrado(l1)+"\n");
65          }
66          if (opcion==2)
67          {
68              mensaje("Ingrese el valor del lado 1 del rectangulo: ");
69              l1=LecturaDatos.leerDatoEntero();
70              mensaje("Ingrese el valor del lado 2 del rectangulo: ");
71              l2=LecturaDatos.leerDatoEntero();
72              mensaje("El area del rectangulo es: "+F.areaRectangulo(l1,l2)+"\n");
73              mensaje("El perimetro del rectangulo es: "+
74                      F.perimetroRectangulo(l1,l2)+"\n");
75          }
76          if (opcion==3)
77          {
78              mensaje("Ingrese el valor del lado 1 del triangulo: ");
79              l1=LecturaDatos.leerDatoEntero();
80              System.out.print("Ingrese el valor del lado 2 del triangulo: ");
81              l2=LecturaDatos.leerDatoEntero();
82              System.out.print("Ingrese el valor del lado 3 del triangulo: ");
83              l3=LecturaDatos.leerDatoEntero();
84              mensaje("El area del triangulo es: "+F.areaTriangulo(l1,l2,l3)+"\n");
85              mensaje("El perimetro del rectangulo es: "+
86                      F.perimetroTriangulo(l1,l2,l3)+"\n");
87          }
88      }
89  }
```

Ejecución del programa con métodos no estáticos

```
Output - UTEX_Tecnicas_Programacion (run)
run:
CALCULO DEL AREA Y PERIMETRO
Ingrese 1: cuadrado, 2:rectangulo, 3:triangulo -> 1
Ingrese el valor del lado del cuadrado: 12
El area del cuadrado es: 144.0
El perimetro del cuadrado es: 48
BUILD SUCCESSFUL (total time: 8 seconds)
```

AUTOEVALUACIÓN

a) Responde a las siguientes preguntas:

1. Clase que pertenece al paquete java.io y permite hacer lectura de datos desde el teclado: _____
2. Método que permite calcular la potencia dado la base y el exponente: _____
3. Método que devuelve el número de caracteres o la longitud de caracteres de una cadena de texto: _____
4. Método que permite evaluar si una cadena 1 es igual a una cadena 2 sin importar las mayúsculas o minúsculas: _____
5. Los métodos que no retornan valor usan la instrucción return (verdadero o falso): _____
6. Una clase tiene como miembros atributos y métodos (verdadero o falso): _____
7. Para crear o instanciar un objeto de la clase Rectángulo se debe escribir: _____
8. Los elementos o miembros que sólo pueden ser accedidos dentro una clase, pertenecen al nivel de acceso _____
9. Modificador que permite que un miembro pertenezca a la clase y no a un objeto es: _____
10. Las relaciones entre clases permiten conocer las asociaciones existentes entre ellos (verdadero o falso): _____

b) Desarrolla las siguientes aplicaciones:

1. Construye una aplicación que permita el ingreso de una cantidad de segundos y muestre su equivalencia en días, horas, minutos y segundos. Posteriormente construye una aplicación que permita ingresar una cantidad de soles y muestre su equivalencia en billetes de 200 soles, de 100 soles, 50 soles, 20 soles, 10 soles y en monedas de 5 soles, 2 soles y de 1 sol.

2. Construye una aplicación que permita el ingreso de una cantidad de segundos y muestre su equivalencia en días, horas, minutos y segundos. Esta vez lo haces creando un método.

3. Construye una aplicación que permita sumar los digitos de un numero entero usando recursividad de métodos.

4. Construye una aplicación que permita el ingreso de 5 números en un vector A y 5 números en un vector B. Luego coloque todos los números tanto del vector A y del vector B en un vector C. Finalmente muestre la suma y el promedio de todos los números.

5. Construye una aplicación que permita crear la clase Trapecio con los atributos con base mayor, base menor y la altura. Luego calcula el perímetro y el área del trapecio.

6. Construye una aplicación que permita crear la clase Marca con los atributos codigomar y nombremar. También crear la clase Bus con los atributos codigobus, nroasientos, nroejes y nropisos. Luego proceder a ingresar datos en Marca y en Bus y posteriormente visualizar los datos de los buses ingresados. Hacer uso de arreglos.

Capítulo II

HERENCIA, CLASES ABSTRACTAS, INTERFACES, POLIMORFISMO, PAQUETES Y EXCEPCIONES

8. MÉTODOS CONSTRUCTORES Y LA REFERENCIA THIS

8.1. ¿Qué son métodos constructores?

Toda clase tiene al menos un método constructor declarado de manera explícita o implícita. Es utilizado para la creación de los objetos y comúnmente establece los valores iniciales de los atributos al momento de instanciar el objeto. Los métodos constructores pueden tener parámetros y si una clase no especifica un método constructor el compilador de Java le adiciona uno de acceso público sin parámetros y cuya implementación no contiene instrucción de programación alguna.

8.2. Características de los métodos constructores

Los métodos constructores tienen las siguientes características:

a) Un método constructor debe ser declarado con nivel de acceso público, en otros casos que amerite se podría cambiar el nivel de acceso.

b) Un método constructor siempre tiene el mismo nombre de la clase a la que pertenece.

c) Un método constructor no retorna valor alguno y no necesita hacer uso de la palabra reservada void.

d) Un método constructor no puede ser heredado.

Por ejemplo:

```
public class Operaciones
{
        private int número1;
        private int número2;

        public Operaciones()
        {
                número1=0;
                número2=0;
        }
}
```

En la clase Operaciones se establece dos atributos numero1 y numero2 de tipo entero. En el método constructor se inicializa con el valor de 0 a los dos atributos.

8.3. Sobrecarga de métodos constructores

En una clase pude existir más de un método constructor, con o sin parámetros. Si hay más de un método constructor con parámetros deberán diferenciarse en cantidad de parámetros o en la declaración de tipo de dichos parámetros. Por ejemplo:

```
public class Operaciones
{
        private int número1;
        private int número2;

        public Operaciones()
        {
                número1=0;
                número2=0;
        }
```

```
        public Operaciones(int x, int y)
        {
        número1=x;
        número2=y;
        }
}
```

En este programa existe en la clase Operaciones dos métodos constructores uno sin parámetros que inicializa a 0 a los atributos y el otro con dos parámetros que establece un valor inicial según los valores de los parámetros.

8.4. Uso de la referencia this

Cuando se pretende hacer uso de los métodos puede surgir la necesidad de dar valores iniciales a los atributos perteneciente a la clase. Por ejemplo:

```
public class Marca
{
        private int codmarca;
        private String nombre;

        public Marca(int codmarca, String nombre)
        {
                this.codmarca=codmarca;
                this.nombre=nombre;
        }
        …
        …
}
```

En el programa anterior, dentro del constructor Marca se desea asignar valores a los atributos de la clase, sin embargo, dentro del constructor se define como parámetros de entrada los mismos nombres como se han definido los atributos. Para tener acceso a los atributos de la clase Marca se utiliza la referencia this. Cuando se usa this se está haciendo referencia al atributo de la clase y la omisión implica que se está usando el parámetro.

8.5. Programas resueltos en Java usando NetBeans

1) Se desea crear una clase llamada Camisa que haga uso de un método constructor y la referencia this y pueda ser usado en otra clase para realizar un pedido, el mismo que puede ser pagado al contado o al crédito. Al contado hay un descuento del 10% y al crédito un incremento del 5% del precio total pagadero en 2 cuotas.

```
1   package Capitulo_II.Sesion_08;
2
3   public class Camisa {
4
5       private String descripcion;
6       private String talla;
7       private int stock;
8       private double precio;
9
10      public Camisa(String descripcion, String talla, int stock, double precio) {
11          this.descripcion = descripcion;
12          this.talla = talla;
13          this.stock = stock;
14          this.precio = precio;
15      }
16
17      public void setDescripcion(String descripcion) {
18          this.descripcion = descripcion;
19      }
20
21      public void setTalla(String talla) {
22          this.talla = talla;
23      }
24
25      public void setStock(int stock) {
26          this.stock = stock;
27      }
28
29      public void setPrecio(double precio) {
30          this.precio = precio;
31      }
32
33      public String getDescripcion() {
34          return descripcion;
35      }
```

```
36
37      public String getTalla() {
38          return talla;
39      }
40
41      public int getStock() {
42          return stock;
43      }
44
45      public double getPrecio() {
46          return precio;
47      }
48
49      @Override
50      public String toString() {
51          return "La camisa "+descripcion+ " de talla "+talla+" cuesta "+precio;
52      }
53
54      public double preciodescontado()
55      {
56          return precio*0.9;
57      }
58
59      public double precioincrementado()
60      {
61          return precio*1.05;
62      }
63
64  }
```

En la clase PedidoCamisas se programa lo siguiente.

```
1   package Capitulo_II.Sesion_08;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   public class PedidoCamisas {
7
8       public static void imprime(String m)
9       {
10          System.out.print(m);
11      }
12
13      public static void main (String args[])throws IOException
14      {
15          int s, cod, cant,fpago;
16          String des,t;
17          double p;
18          imprime("=========================================\n");
19          imprime("          PEDIDOS DE CAMISAS             \n");
20          imprime("=========================================\n");
21          imprime("Descripcion de la camisa : ");
22          des=LecturaDatos.leerDatoTexto();
23          imprime("Talla de la camisa : ");
24          t=LecturaDatos.leerDatoTexto();
25          imprime("Stock de la camisa : ");
26          s=LecturaDatos.leerDatoEntero();
27          imprime("Precio unitario de la camisa : ");
28          p=LecturaDatos.leerDatoReal();
29          imprime("\n");
30          Camisa C=new Camisa(des, t, s, p);
31          imprime(C+"\n");
32          imprime("\n");
```

```
33          do
34          {
35              imprime("Cantidad a pedir : ");
36              cant=LecturaDatos.leerDatoEntero();
37          }while(cant>s);
38          imprime("Forma de pago (1: Contado, 2: Credito) : ");
39          fpago=LecturaDatos.leerDatoEntero();
40          imprime("\n");
41          if(fpago==1)
42              imprime("El precio total del pedido es "+
43                      (C.preciodescontado()*cant)+"\n");
44          else
45          {
46              imprime("El precio total del pedido es "+
47                      (C.precioincrementado()*cant)+"\n");
48              imprime("El cliente debera pagar dos cuotas de "+
49                      (C.precioincrementado()*cant)/2+"\n");
50          }
51      }
52
53  }
```

Interpretación de la programación

La clase camisa tiene cuatro atributos referido a la descripción, talla, stock y precio. El método constructor Camisa hace usa de referencia this. Se crea los métodos set y get y se adiciona los métodos toString, preciodescontado y precioaumntado para luego ser usado en la clase PedidoCamisas. En esta última clase hace uso de la clase LecturaDatos que se encuentra en el paquete Capítulo I. Sesión_06. En el método main se ingresa los datos de una camisa y luego se solicita la cantidad de pedido y la forma de pago para finalmente mostrar al final el precio total del pedido y en caso la forma de pago es al crédito se visualiza el monto de la cuota.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
==========================================
           PEDIDOS DE CAMISAS
==========================================
Descripcion de la camisa : Algodon de manga larga
Talla de la camisa : Large
Stock de la camisa : 80
Precio unitario de la camisa : 42.50

La camisa Algodon de manga larga de talla Large cuesta 42.5

Cantidad a pedir : 50
Forma de pago (1: Contado, 2: Credito) : 2

El precio total del pedido es 2231.25
El cliente debera pagar dos cuotas de 1115.625
```

2) Crea una aplicación que permita calcular el área y el perimetro de una cuadrado, rectángulo y del triángulo. Hacer uso de métodos constructores sobrecargados y de la referencia this.

```
1   package Capitulo_II.Sesion_08;
2
3   public class Figura {
4
5       private double lado1;
6       private double lado2;
7       private double lado3;
8
9       public Figura(double lado1) {
10          this.lado1 = lado1;
11      }
12
13      public Figura(double lado1, double lado2) {
14          this.lado1 = lado1;
15          this.lado2 = lado2;
16      }
17
18      public Figura(double lado1, double lado2, double lado3) {
19          this.lado1 = lado1;
20          this.lado2 = lado2;
21          this.lado3 = lado3;
22      }
23
24      public void setLado1(double lado1) {
25          this.lado1 = lado1;
26      }
27
28      public void setLado2(double lado2) {
29          this.lado2 = lado2;
30      }
31
32      public void setLado3(double lado3) {
33          this.lado3 = lado3;
34      }
35
36      public double getLado1() {
37          return lado1;
38      }
39
40      public double getLado2() {
41          return lado2;
42      }
43
44      public double getLado3() {
45          return lado3;
46      }
47
48      public double areaCuadrado()
49      {
50          return lado1*lado1;
51      }
52
53      public double areaRectangulo()
54      {
55          return lado1*lado2;
56      }
57
```

```
58      public double areaTriangulo()
59      {
60          double p,r;
61          p=(lado1+lado2+lado3)/2;
62          r=p*(p-lado1)*(p-lado2)*(p-lado3);
63          if(r>0)
64              return Math.sqrt(r);
65          else
66              return 0.0;
67      }
68
69      public double perimetroCuadrado()
70      {
71          return 4*lado1;
72      }
73
74      public double perimetroRectangulo()
75      {
76          return 2*lado1+2*lado2;
77      }
78
79      public double perimetroTriangulo()
80      {
81          return lado1+lado2+lado3;
82      }
83  }
```

En la clase UsandoFigura se programa lo siguiente:

```
1   package Capitulo_II.Sesion_08;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   public class UsandoFigura {
7       public static void imprime(String m)
8       {
9           System.out.print(m);
10      }
11
12      public static void main(String args[])throws IOException
13      {
14          int f;
15          double l1, l2, l3;
16          imprime("=========================\n");
17          imprime("   FIGURAS GEOMETRICAS    \n");
18          imprime("=========================\n");
19          imprime("Indique la figura geometrica (1 Cuadrado, 2 Rectangulo, 3 Triangulo) : ");
20          f=LecturaDatos.leerDatoEntero();
21          if(f==1)
22          {
23              imprime("Indique el valor del lado del cuadrado : ");
24              l1=LecturaDatos.leerDatoReal();
25              Figura F=new Figura(l1);
26              imprime("El area del cuadrado es "+F.areaCuadrado()+"\n");
27              imprime("El perimetro del cuadrado es "+F.perimetroCuadrado()+"\n");
28          }
29          if(f==2)
30          {
31              imprime("Indique el valor del lado 1 del rectangulo : ");
32              l1=LecturaDatos.leerDatoReal();
33              imprime("Indique el valor del lado 2 del rectangulo : ");
34              l2=LecturaDatos.leerDatoReal();
35              Figura F=new Figura(l1,l2);
36              imprime("El area del rectangulo es "+F.areaRectangulo()+"\n");
37              imprime("El perimetro del rectangulo es "+F.perimetroRectangulo()+"\n");
38          }
```

```
39              if(f==3)
40              {
41                  imprime("Indique el valor del lado 1 del triangulo : ");
42                  l1=LecturaDatos.leerDatoReal();
43                  imprime("Indique el valor del lado 2 del triangulo : ");
44                  l2=LecturaDatos.leerDatoReal();
45                  imprime("Indique el valor del lado 3 del triangulo : ");
46                  l3=LecturaDatos.leerDatoReal();
47                  Figura F=new Figura(l1,l2,l3);
48                  if(F.areaTriangulo()!=0.0)
49                  {
50                      imprime("El area del triangulo es "+F.areaTriangulo()+"\n");
51                      imprime("El perimetro del triangulo es "+F.perimetroTriangulo()+"\n");
52                  }
53                  else
54                  {
55                      imprime("Los datos ingresados no corresponde a un triangulo\n");
56                  }
57              }
58          }
59      }
```

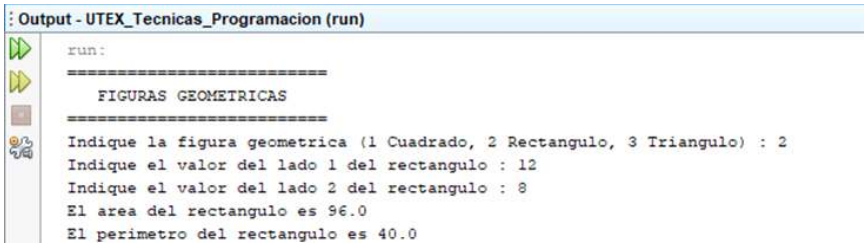
Interpretación de la programación

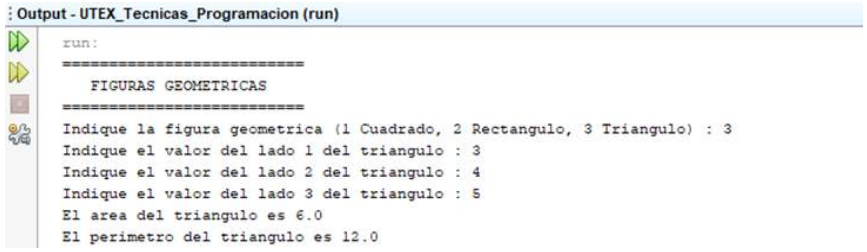
En la clase Figura se crea tres métodos constructores, el primero sólo usa uno de los tres atributos y está diseñado para la figura del cuadrado. El segundo método constructor usa dos atributos y está creado para la figura del rectángulo. El tercer método constructor usa tres atributos y está diseñado para la figura del triángulo. Se crea tres métodos para el cálculo del área y tres métodos para el cálculo de los perímetros. También está los métodos set y get de cada uno de los atributos.

En la clase UsandoFigura se importa nuevamente la clase LecturaDatos que se encuentra en el paquete Capítulo_I.Sesión_06. En el método main se procede a solicitar la figura geométrica a usar para los cálculos y según la figura seleccionada se solicita el ingreso de los valores de los lados. Finalmente se muestra el cálculo del área y del perímetro de la figura. En el caso del triángulo se hace una validación en caso que los datos ingresados de los tres lados no lleguen a formar un triángulo.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
==========================
   FIGURAS GEOMETRICAS
==========================
Indique la figura geometrica (1 Cuadrado, 2 Rectangulo, 3 Triangulo) : 1
Indique el valor del lado del cuadrado : 12.5
El area del cuadrado es 156.25
El perimetro del cuadrado es 50.0
```


Output - UTEX_Tecnicas_Programacion (run)
run:
==========
FIGURAS GEOMETRICAS
==========
Indique la figura geometrica (1 Cuadrado, 2 Rectangulo, 3 Triangulo) : 2
Indique el valor del lado 1 del rectangulo : 12
Indique el valor del lado 2 del rectangulo : 8
El area del rectangulo es 96.0
El perimetro del rectangulo es 40.0


Output - UTEX_Tecnicas_Programacion (run)
run:
==========
FIGURAS GEOMETRICAS
==========
Indique la figura geometrica (1 Cuadrado, 2 Rectangulo, 3 Triangulo) : 3
Indique el valor del lado 1 del triangulo : 3
Indique el valor del lado 2 del triangulo : 4
Indique el valor del lado 3 del triangulo : 5
El area del triangulo es 6.0
El perimetro del triangulo es 12.0

9. HERENCIA DE CLASES

9.1. ¿Qué es herencia de clases?

La herencia de clases es la extensión de una clase, es decir, una nueva clase hereda los atributos y métodos de una clase superior denominada comúnmente superclase. La nueva clase creada se le denomina subclase. El lenguaje de programación no reconoce la herencia múltiple, es decir, no es posible que una clase herede de varias clases, lo que si permite es el uso de interfaces para lograr una herencia múltiple. Una subclase, llamada también clase hija, generalmente tiene sus propios atributos y métodos, por lo que una subclase maneja más atributos y métodos de una superclase. Una subclase representa un grupo pequeño de objetos dado que los atributos y métodos son más específicos. Cuando se crea un objeto de la subclase se tiene los atributos y métodos de la superclase más los propios atributos y métodos de la subclase.

La herencia es la base implica la reutilización del código. Cuando una clase se extiende de una clase padre, esta hereda todos los miembros y métodos de su antecesor. también es posible redefinir (override) los miembros para adaptarlos a la nueva clase o ampliarlos. en general, todas las subclases no solo adoptan las variables y métodos de las superclases, sino que los modifican. (3)

Por ejemplo:

```
class Empleado
{
```

```
        protected String apellidos;
        protected String nombres;

        public Empleado(String ape, String nom)
        {
        apellidos=ape;
        nombres=nom;
        }
        …
        …
}

class Administrativo extends Empleado
{
        private double sueldobasico;


        public Administrativo(String ape, String nom, double sb)
        {
                super(ape,nom);
                sueldobasico=sb;
        }
}
```

En el programa anterior se tiene la clase Empleado que tiene dos atributos: apellidos y nombres y el método constructor define los valores iniciales de los atributos según los parámetros ape y nom. Esta clase puede tener otros métodos más. En la clase Administrativo es la subclase siendo la clase Empleado la superclase. En Java se usa la palabra reservada extends para indicar el uso de herencia. En esta clase Administrativo se indica un solo atributo denominado

sueldobasico y en el método constructor el número de parámetros considera tres parámetros para dar valores a los dos atributos heredados de la clase Empleado, es por ello que en el método constructor de la clase Administrativo se hace en la primera instrucción del método super para invocar la ejecución del método constructor de la superclase por lo que es necesario indicar los parámetros ape y nom.

Con las clases se puede crear una jerarquía, una clase Universitario puede heredar de la clase Estudiante y este a su vez heredar de la clase Persona. La clase Universitario hereda los atributos de la clase Estudiante y de las que heredó de Persona. En la medida que se vayan creando más subclases en una jerarquía estas se vuelven cada vez más especializadas.

9.2. Modificador de acceso protected

Los miembros de una clase que hacen uso del modificador de acceso protected están accesibles para los métodos de la superclase, para los métodos de la subclase y para los métodos de aquellas clases que pertenezcan al mismo paquete.

9.3. Objetos de la superclase y objetos de subclase

Se puede crear objetos de la superclase y objetos de la subclase. Los objetos de la subclase tienen definitivamente las características de la superclase, así por ejemplo si se crea una clase llamado Cuadrado que tiene un solo atributo que vendría ser lado, se puede construir un método para definir el área del cuadrado. Luego, se puede crear una clase llamada PrismaCuadrangular que herede de la clase Cuadrado y haciendo uso del método de área del cuadrado se puede construir el área del prisma cuadrangular.

```
import java.io.*;
class Cuadrado
{
        protected double lado;
```

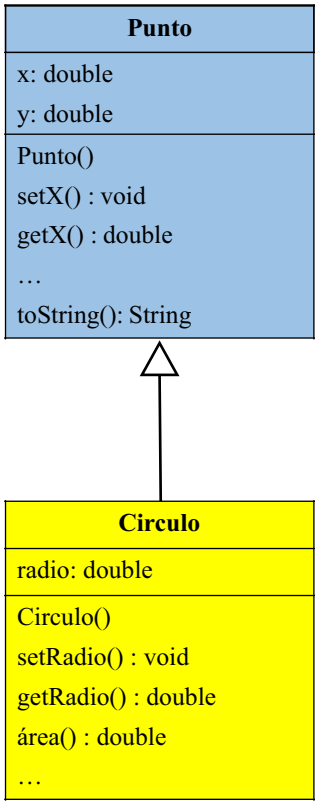
```
	public Cuadrado(double lado)
	{
		this.lado=lado;
	}
	public void setLado(double la)
	{
		lado=la;
	}
	public double getLado()
	{
		return lado;
	}
	public double área()
	{
		return lado*lado;
	}
}

class PrismaCuadrangular extends Cuadrado
{
	protected double altura;

	public PrismaCuadrangular(double la, double h)
	{
		super(la, h);
		altura=h;
	}
	public void setAltura(double h)
	{
		altura=h;
```

```
        }
        public double getAltura()
        {
                return altura;
        }
        public double areaPrisma()
        {
                return super.área()*2+altura*lado*4;
        }
}

public class UsandoPrismaCuadrangular
{
      public static main(String args[])throws IOException
      {
            Cuadrado C=new Cuadrado(5);
            System.out.println("El área del cuadrado es "+C.area());
            PrismaCuadrangular P=new PrismaRectangular(5, 7);
            System.out.println("El área de la base del prisma es
                                    "+P.area());
            System.out.println("El área total del prisma es
                                    "+P.areaPrisma());
      }
}
```

En el método main de la clase UsandoPrismaCuadrangular se define un objeto C creado a partir de la clase Cuadrado dando como valor 5 por defecto para el atributo lado. Se imprime el área del cuadrado y luego se crea el objeto P creado a partir de la clase PrismaRectangular dando como valores por defecto 5 y 7 para los atributos lado y altura y posteriormente se imprime el área de la base del prisma y el área total del mismo.

9.4. Programas resueltos en Java usando NetBeans

1) Crea una clase Punto con los atributos x, y de tipo de dato numérico real. A partir de la clase Punto se debe crear la clase Circulo con el atributo radio y en donde se debe tener el método de área. Elabora un programa que permita hacer uso de las clases Punto y Circulo para calcular el área de un círculo específico. Tener en cuenta el siguiente diagrama de clases:




```
1   package Capitulo_II.Sesion_09;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   class Punto {
7
8       protected double x;
9       protected double y;
```

```
10
11      public Punto() {
12          x = 0;
13          y = 0;
14      }
15
16      public Punto(double a, double b) {
17          x = a;
18          y = b;
19      }
20
21      public void setX(double a) {
22          x = a;
23      }
24
25      public void setY(double b) {
26          y = b;
27      }
28
29      public double getX() {
30          return x;
31      }
32
33      public double getY() {
34          return y;
35      }
36
37      @Override
38      public String toString() {
39          return "Punto en (" + x + "," + y + ")";
40
41      }
42  }
43
44  class Circulo extends Punto {
45
46      protected double radio;
47
48      public Circulo() {
49          super();
50          radio = 0;
51      }
52
53      public Circulo(double a, double b, double r) {
54          super(a, b);
55          radio = r;
56      }
57
58      public void setRadio(double a) {
59          radio = a;
60      }
61
62      public double getRadio() {
63          return radio;
64      }
65
66      public double area() {
67          return Math.PI * Math.pow(radio, 2);
68      }
69
```

```
71      public String toString() {
72          return super.toString() + " con Radio = " + radio;
73      }
74
75  }
76
77  public class HerenciaCirculo {
78
79      public static void imprime(String m)
80      {
81          System.out.print(m);
82      }
83
84      public static void main(String args[]) throws IOException {
85
86          double x, y, r;
87          imprime("*********************************\n");
88          imprime("  CALCULO DEL AREA DEL CIRCULO   \n");
89          imprime("*********************************\n");
90          imprime("Dado que las coordenadas del punto \n");
91          imprime("X = ");
92          x=LecturaDatos.leerDatoReal();
93          imprime("Y = ");
94          y=LecturaDatos.leerDatoReal();
95          imprime("Valor del radio : ");
96          r=LecturaDatos.leerDatoReal();
97          Circulo C = new Circulo(x,y,r);
98          imprime("El area del circulo es "+C.area()+"\n");
99
100     }
```

Interpretación de la programación

La clase Circulo hereda de la clase Punto que en código de Java se escribe

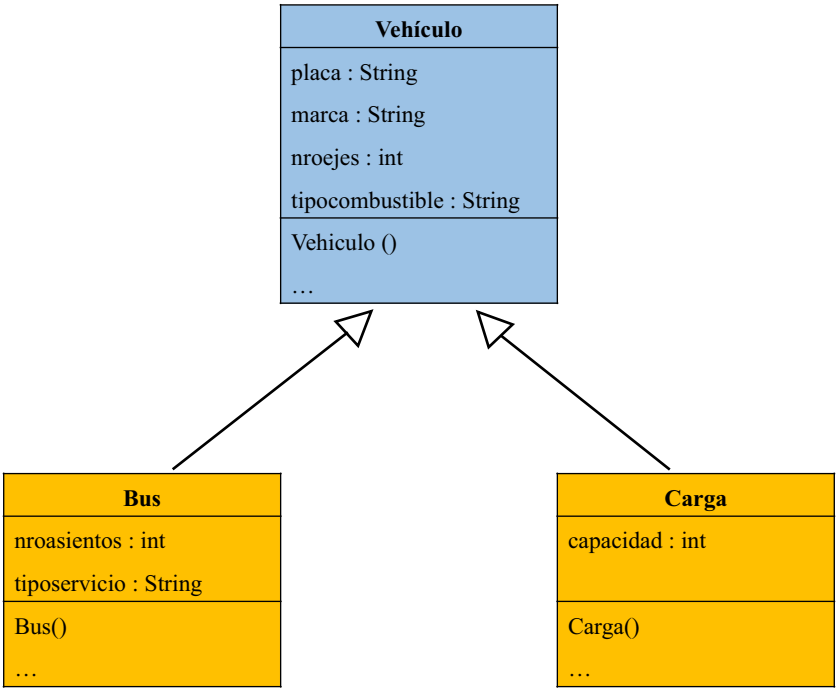
Class Círculo extends Punto

La palabra reservada extends significa extendido que es lo mismo heredado, es decir, la clase Circulo hereda de la clase Punto todos sus atributos y métodos, siempre y cuando ninguno de los miembros de la clase Punto tenga nivel de acceso tipo private. Los métodos constructores de la clase Circulo puede invocar a los métodos constructores de la clase Punto. Para poder invocar al constructor de la clase Punto se hace uso el método o referencia super. La subclase puede nuevamente implementar un método heredado de la superclase, es decir, se puede sobrescribir lo realizado y se necesita tomar algo programado en la superclase se le invoca con la referencia super. En el método toString de la clase Circulo se muestra en el retorno super.toString() lo que significa que invoca la ejecución del método toString de la clase Punto.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
**********************************
  CALCULO DEL AREA DEL CIRCULO
**********************************
Dado que las coordenadas del punto
X = 4
Y = 3
Valor del radio : 9
El area del circulo es 254.46900494077323
BUILD SUCCESSFUL (total time: 10 seconds)
```

2) Crea una clase vehículo con los atributos placa, marca, nroejes y tipocombustible. A partir de la clase Vehiculo se crea la clase Bus con los atributos propios nroasientos y tiposervicio y la clase Carga con el único atributo propio denominado capacidad. Elabora un programa que permita el ingreso de datos de un bus o de un vehículo de carga. Tener en cuenta el siguiente diagrama de clases:



```java
package Capitulo_II.Sesion_09;

import java.io.*;
import Capitulo_I.Sesion_06.LecturaDatos;

class Vehiculo
{
    protected String placa;
    protected String marca;
    protected int nroejes;
    protected String tipocombustible;

    public Vehiculo(String placa, String marca, int nroejes, String tipocombustible) {
        this.placa = placa;
        this.marca = marca;
        this.nroejes = nroejes;
        this.tipocombustible = tipocombustible;
    }

    public void setPlaca(String placa) {
        this.placa = placa;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public void setNroejes(int nroejes) {
        this.nroejes = nroejes;
    }

    public void setTipocombustible(String tipocombustible) {
        this.tipocombustible = tipocombustible;
    }

    public String getPlaca() {
        return placa;
    }

    public String getMarca() {
        return marca;
    }

    public int getNroejes() {
        return nroejes;
    }

    public String getTipocombustible() {
        return tipocombustible;
    }

    @Override
    public String toString() {
        return "Vehiculo[" + "placa=" + placa + ", marca=" + marca +
                ", numero de ejes=" + nroejes + ", tipo de combustible=" +
                tipocombustible + "}\n";
    }

}

class Bus extends Vehiculo
{
    private int nroasientos;
    private String tiposervicio;
```

```
66      public Bus(int nroasientos, String tiposervicio, String placa, String marca,
67              int nroejes, String tipocombustible) {
68          super(placa, marca, nroejes, tipocombustible);
69          this.nroasientos = nroasientos;
70          this.tiposervicio = tiposervicio;
71      }
72
73      public void setNroasientos(int nroasientos) {
74          this.nroasientos = nroasientos;
75      }
76
77      public void setTiposervicio(String tiposervicio) {
78          this.tiposervicio = tiposervicio;
79      }
80
81      public int getNroasientos() {
82          return nroasientos;
83      }
84
85      public String getTiposervicio() {
86          return tiposervicio;
87      }
88
89      @Override
90      public String toString() {
91          return super.toString()+"Bus{" + "numero de asientos=" + nroasientos +
92                  ", tipo de servicio=" + tiposervicio + "}\n";
93      }
94  }
95
96  class Carga extends Vehiculo
97  {
98      private int capacidad;
99
100     public Carga(int capacidad, String placa, String marca, int nroejes,
101             String tipocombustible) {
102         super(placa, marca, nroejes, tipocombustible);
103         this.capacidad = capacidad;
104     }
105
106     public void setCapacidad(int capacidad) {
107         this.capacidad = capacidad;
108     }
109
110     public int getCapacidad() {
111         return capacidad;
112     }
113
114     @Override
115     public String toString() {
116         return super.toString()+"Carga{" + "capacidad=" + capacidad + " toneladas}\n";
117     }
118 }
119
120 public class HerenciaBusCarga {
121
122     public static void mensaje(String m)
123     {
124         System.out.print(m);
125     }
126
127     public static void main(String args[])throws IOException
128     {
129         int tipo,ne,na,ca;
130         String p,m,tc,ts;
```

```
131             mensaje("==============================\n");
132             mensaje("DATOS DE BUS O VEHICULO DE CARGA\n");
133             mensaje("==============================\n");
134             mensaje("Indique 1 si es bus o 2 si es vehiculo de carga : ");
135             tipo=LecturaDatos.leerDatoEntero();
136             mensaje("Ingrese la placa del vehiculo : ");
137             p=LecturaDatos.leerDatoTexto();
138             mensaje("Ingrese la marca del vehiculo : ");
139             m=LecturaDatos.leerDatoTexto();
140             mensaje("Ingrese el numero de ejes del vehiculo : ");
141             ne=LecturaDatos.leerDatoEntero();
142             mensaje("Ingrese el tipo de combustible del vehiculo : ");
143             tc=LecturaDatos.leerDatoTexto();
144             if(tipo==1)
145             {
146                 mensaje("Ingrese el numero de asientos del bus : ");
147                 na=LecturaDatos.leerDatoEntero();
148                 mensaje("Ingrese el tipo de servivio del bus : ");
149                 ts=LecturaDatos.leerDatoTexto();
150                 Bus B=new Bus(na,ts,p,m,ne,tc);
151                 mensaje(B+"\n");
152             }
153             if(tipo==2)
154             {
155                 mensaje("Ingrese la capacidad en toneladas del vehiculo de carga : ");
156                 ca=LecturaDatos.leerDatoEntero();
157                 Carga C=new Carga(ca,p,m,ne,tc);
158                 mensaje(C+"\n");
159             }
160         }
161
162     }
```

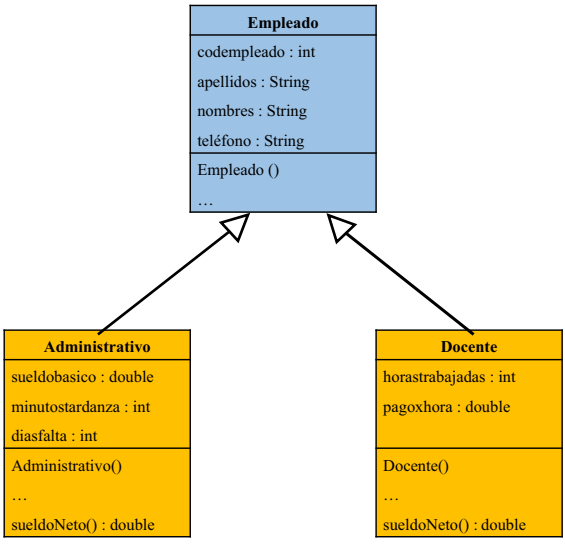
Interpretación de la programación

En el programa se aprecia que la clase Bus y la clase Carga hereda de la clase Vehículo. Esta vez se establece en las subclases y en la superclase el uso del método toString. En este método se hace uso del método toString de la clase Vehículo logrando obtener una cadena de texto con los datos de los atributos obtenidos de la superclase y de la subclase. En la clase HerenciaBusCarga en el método main se solicita el ingreso de los datos ya sea del vehículo Bus o del vehículo Carga, se instancia un objeto de la clase Bus o de clase Carga y se muestra luego los datos ingresados.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
==============================
DATOS DE BUS O VEHICULO DE CARGA
==============================
Indique 1 si es bus o 2 si es vehiculo de carga : 1
Ingrese la placa del vehiculo : BL-3425
Ingrese la marca del vehiculo : Mercedes Benz
Ingrese el numero de ejes del vehiculo : 3
Ingrese el tipo de combustible del vehiculo : Petroleo
Ingrese el numero de asientos del bus : 50
Ingrese el tipo de servivio del bus : Bus cama
Vehiculo{placa=BL-3425, marca=Mercedes Benz, numero de ejes=3, tipo de combustible=Petroleo}
Bus{numero de asientos=50, tipo de servicio=Bus cama}

BUILD SUCCESSFUL (total time: 1 minute 33 seconds)
```

3) Crea una aplicación que permita realizar la planilla de trabajadores administrativos o trabajadores docentes. Los administrativos tienen un sueldo básico, pueden tener minutos de tardanza (se descuenta 0.50 soles por minuto) y días de falta (se aplica un descuento igual a los días de falta por sueldo básico entre 30). Aplica herencia de clases, arreglo de objetos para ingresar datos de varios trabajadores y lista los datos indicando el sueldo neto de cada trabajador. Tener en cuenta el siguiente diagrama de clases:



```java
package Capitulo_II.Sesion_09;

import java.io.*;
import Capitulo_I.Sesion_06.LecturaDatos;

class Empleado
{
    protected int codempleado;
    protected String apellido;
    protected String nombre;
    protected String telefono;

    public Empleado(int codempleado, String apellido, String nombre,
            String telefono) {
        this.codempleado = codempleado;
        this.apellido = apellido;
        this.nombre = nombre;
        this.telefono = telefono;
    }

    public void setCodempleado(int codempleado) {
        this.codempleado = codempleado;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public int getCodempleado() {
        return codempleado;
    }

    public String getApellido() {
        return apellido;
    }

    public String getNombre() {
        return nombre;
    }

    public String getTelefono() {
        return telefono;
    }
}

class Administrativo extends Empleado
{
    private double sueldobasico;
    private int minutostardanza;
    private int diasfalta;

    public Administrativo(double sueldobasico, int minutostardanza, int diasfalta,
            int codempleado, String apellido, String nombre, String telefono) {
        super(codempleado, apellido, nombre, telefono);
        this.sueldobasico = sueldobasico;
        this.minutostardanza = minutostardanza;
        this.diasfalta = diasfalta;
    }

```

```
68      public void setSueldobasico(double sueldobasico) {
69          this.sueldobasico = sueldobasico;
70      }
71
72      public void setMinutostardanza(int minutostardanza) {
73          this.minutostardanza = minutostardanza;
74      }
75
76      public void setDiasfalta(int diasfalta) {
77          this.diasfalta = diasfalta;
78      }
79
80      public double getSueldobasico() {
81          return sueldobasico;
82      }
83
84      public int getMinutostardanza() {
85          return minutostardanza;
86      }
87
88      public int getDiasfalta() {
89          return diasfalta;
90      }
91
92      public double sueldoNeto()
93      {
94          return sueldobasico - 0.5*minutostardanza - diasfalta*sueldobasico/30;
95      }
96
97  }
98
99  class Docente extends Empleado
100 {
101     private int horastrabajadas;
102     private double pagoxhora;
103
104     public Docente(int horastrabajadas, double pagoxhora, int codempleado,
105             String apellido, String nombre, String telefono) {
106         super(codempleado, apellido, nombre, telefono);
107         this.horastrabajadas = horastrabajadas;
108         this.pagoxhora = pagoxhora;
109     }
110
111     public int getHorastrabajadas() {
112         return horastrabajadas;
113     }
114
115     public void setHorastrabajadas(int horastrabajadas) {
116         this.horastrabajadas = horastrabajadas;
117     }
118
119     public double getPagoxhora() {
120         return pagoxhora;
121     }
122
123     public void setPagoxhora(double pagoxhora) {
124         this.pagoxhora = pagoxhora;
125     }
126
127     public double sueldoNeto()
128     {
129         return horastrabajadas*pagoxhora;
130     }
131
132 }
```

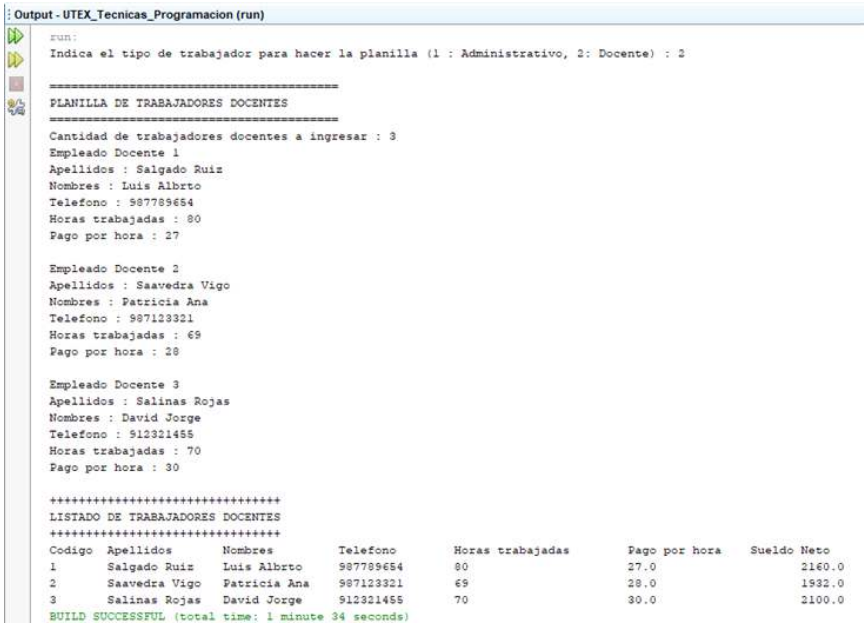
```
133
134  public class HerenciaAdministrativoDocente {
135
136      public static void imprime(String m)
137      {
138          System.out.print(m);
139      }
140
141      public static Administrativo [] leerDatosAdministrativos(Administrativo a[])
142              throws IOException
143      {
144          int i, mt, df;
145          String ape,nom,f,t;
146          double sb;
147
148          for(i=0;i<a.length;i++)
149          {
150              imprime("Empleado Administraivo "+(i+1)+"\n");
151              imprime("Apellidos : ");
152              ape=LecturaDatos.leerDatoTexto();
153              imprime("Nombres : ");
154              nom=LecturaDatos.leerDatoTexto();
155              imprime("Telefono : ");
156              t=LecturaDatos.leerDatoTexto();
157              imprime("Sueldo basico : ");
158              sb=LecturaDatos.leerDatoReal();
159              imprime("Minutos de tardanza : ");
160              mt=LecturaDatos.leerDatoEntero();
161              imprime("Dias de falta : ");
162              df=LecturaDatos.leerDatoEntero();
163              Administrativo A=new Administrativo(sb,mt,df,(i+1),ape,nom,t);
164              a[i]=A;
165              imprime("\n");
166          }
167          imprime("");
168          return a;
169      }
170
171      public static Docente [] leerDatosDocentes(Docente d[])
172              throws IOException
173      {
174          int i, ht;
175          String ape,nom,f,t;
176          double ph;
177
178          for(i=0;i<d.length;i++)
179          {
180              imprime("Empleado Docente "+(i+1)+"\n");
181              imprime("Apellidos : ");
182              ape=LecturaDatos.leerDatoTexto();
183              imprime("Nombres : ");
184              nom=LecturaDatos.leerDatoTexto();
185              imprime("Telefono : ");
186              t=LecturaDatos.leerDatoTexto();
187              imprime("Horas trabajadas : ");
188              ht=LecturaDatos.leerDatoEntero();
189              imprime("Pago por hora : ");
190              ph=LecturaDatos.leerDatoReal();
191              Docente D=new Docente(ht,ph,(i+1),ape,nom,t);
192              d[i]=D;
193              imprime("\n");
194          }
195          imprime("");
196          return d;
197      }
198
```

```
    public static void listarAdministrativos(Administrativo a[])
    {
        int i;
        imprime("+++++++++++++++++++++++++++++++++++++++\n");
        imprime("LISTADO DE TRABAJADORES ADMINISTRATIVOS\n");
        imprime("+++++++++++++++++++++++++++++++++++++++\n");
        imprime("Codigo\tApellidos\tNombres\t\tTelefono\tSueldo Basico\t"
                + "Min Tardanza\tDias Falta\t Sueldo Neto\n");
        for(i=0;i<a.length;i++)
        {
            imprime(a[i].getCodempleado()+"\t"+a[i].getApellido()+"\t"
                +a[i].getNombre()+"\t"+a[i].getTelefono()+"\t"+a[i].getSueldobasico()
                +"\t"+a[i].getMinutostardanza()+"\t"+a[i].getDiasfalta()+"\t"
                +a[i].sueldoNeto()+"\n");
        }
    }

    public static void listarDocentes(Docente d[])
    {
        int i;
        imprime("+++++++++++++++++++++++++++++++++\n");
        imprime("LISTADO DE TRABAJADORES DOCENTES\n");
        imprime("+++++++++++++++++++++++++++++++++\n");
        imprime("Codigo\tApellidos\tNombres\t\tTelefono\tHoras trabajadas\t"
                + "Pago por hora\t Sueldo Neto\n");
        for(i=0;i<d.length;i++)
        {
            imprime(d[i].getCodempleado()+"\t"+d[i].getApellido()+"\t"
                +d[i].getNombre()+"\t"+d[i].getTelefono()+"\t"+d[i].getHorastrabajadas()
                +"\t\t\t"+d[i].getPagoxhora()+"\t\t\t"+d[i].sueldoNeto()+"\n");

        }
    }

    public static void main(String args[])throws IOException
    {
        int n,opcion;
        imprime("Indica el tipo de trabajador para hacer la planilla"
                + " (1 : Administrativo, 2: Docente) : ");
        opcion=LecturaDatos.leerDatoEntero();
        imprime("\n");
        imprime("=======================================\n");
        if(opcion==1)
            imprime("PLANILLA DE TRABAJADORES ADMINISTRATIVOS\n");
        else
            imprime("PLANILLA DE TRABAJADORES DOCENTES\n");
        imprime("=======================================\n");
        if(opcion==1)
        {
            do {
                System.out.print("Cantidad de trabajadores administrativos a ingresar : ");
                n = LecturaDatos.leerDatoEntero();
            } while (n <= 0 || n > 20);

            Administrativo administrativos[]=new Administrativo [n];
            administrativos=leerDatosAdministrativos(administrativos);
            listarAdministrativos(administrativos);
        }
        if(opcion==2)
        {
            do {
                System.out.print("Cantidad de trabajadores docentes a ingresar : ");
                n = LecturaDatos.leerDatoEntero();
            } while (n <= 0 || n > 20);

            Docente docentes[]=new Docente [n];
            docentes=leerDatosDocentes(docentes);
            listarDocentes(docentes);
        }
    }
}
```

Interpretación de la programación

En el programa se aprecia que las clases Administrativo y Docente hereda los atributos y métodos de la clase Empleado. El método sueldoNeto tienen implementación diferente por la naturaleza de cálculo de los sueldos de los administrativos y de los docentes. En la clase HerenciaAdministrativoDocente se establece métodos para leer datos para los arreglos de objetos definidos con las clases Administrativo y Docente y los métodos para listar los datos ingresados mostrando los sueldos netos de todos los trabajadores.

Ejecución del programa:



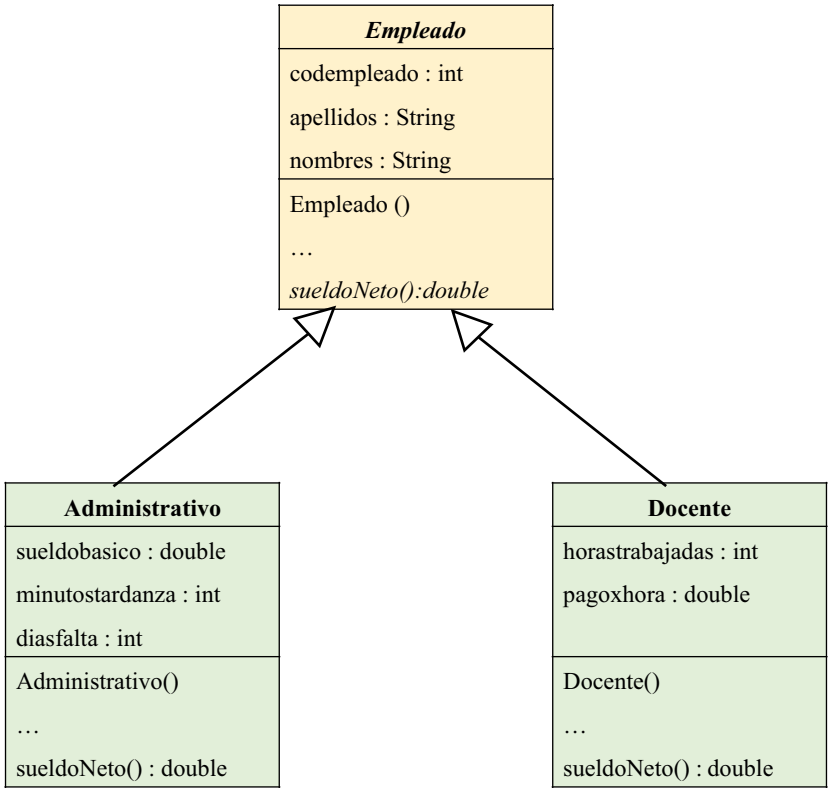
10. CLASES ABSTRACTAS

10.1. ¿Qué son clases abstractas?

Java proporciona un tipo especial de clase, denominada clase abstracta, que es una forma de ayudar a la jerarquización de clases basadas en métodos comunes. Una clase abstracta permite colocar los nombres de los métodos comunes en una sola clase (sin tener que escribir el código que los implemente). Después, al crear nuevas clases, éstas pueden heredar de una clase abstracta que contiene todos los métodos definidos en la clase abstracta. Un método abstracto define qué debe hacerse, pero no indica cómo hacerlo. Para definir un método abstracto se debe anteponer la palabra abstract al momento de la definición, y colocar el carácter ';' en vez de la apertura y cierre de llaves. (4)

Por ejemplo:

En el diagrama de clases siguiente la clase Empleado es una clase abstracta porque tiene un método abstracto llamado sueldoNeto(), es decir, que el método sueldoNeto() en la clase abstrcta Empleado no está implementado (no está programado). En el diagrama se distingue la clase abstracta y el método abstracto escrito con letra cursiva.



10.2. Características de las clases abstractas

Los métodos abstractos tienen una estructura que muestra el nombre del método seguido de una lista de parámetros. No contiene el código o la programación que implementa el método. Estos métodos se implementan en las clases derivadas. Las características de las clases abstractas son:

- Las clases que contienen métodos abstractos se conocen como clases abstractas. Basta que exista un método abstracto para que la clase sea una clase abstracta.
- Las clases abstractas pueden contener una mezcla de métodos abstractos y no abstractos (métodos concretos). Los métodos concretos contienen la instrumentación del método.

- Un aplicación o programa no puede crear instancias de una clase abstracta de forma directa, pero se puede crear instancias de las subclases cuya clase superclase es una clase abstracta.
- Cualquier subclase que herede de una clase abstracta debe proceder a la implementación de todos los métodos abstractos. Basta que un método abstracto no se implementado en la subclase, esta quedará como clase abstracta.

En base al diagrama de clases anterior la implementación de la clase Empleado sería:

```
abstract class Empleado
{
        protected int codempleado;
        protected String apellido;
        protected String nombre;
        protected String telefono;

        public Empleado(int codempleado, String apellido, String
                                nombre, String telefono)
        {
                this.codempleado = codempleado;
                this.apellido = apellido;
                this.nombre = nombre;
                this.telefono = telefono;
        }
        public void setCodempleado(int codempleado)
        {
                this.codempleado = codempleado;
        }
        public int getCodempleado()
```

```
        {
                return codempleado;
        }
        . . .
        public abstract double sueldoNeto();
}
```

El método sueldoNeto() es un método abstracto y por lo tanto no esta implementado. Este método hace que la clase Empleado sea una clase abstracta. A continuación, se hace uso de herencias de clase donde la clase abstracta Empleado es la superclase.

```
class Administrativo extends Empleado
{
        private double sueldobasico;
        private int minutostardanza;
        private int diasfalta;

        public Administrativo(double sueldobasico, int
                minutostardanza, int diasfalta, int codempleado,
                String apellido, String nombre, String telefono) {
                 super(codempleado, apellido, nombre, telefono);
                 this.sueldobasico = sueldobasico;
                 this.minutostardanza = minutostardanza;
                 this.diasfalta = diasfalta;
        }
        public void setSueldobasico(double sueldobasico) {
                 this.sueldobasico = sueldobasico;
        }
        public void setMinutostardanza(int minutostardanza) {
                 this.minutostardanza = minutostardanza;
        }
```

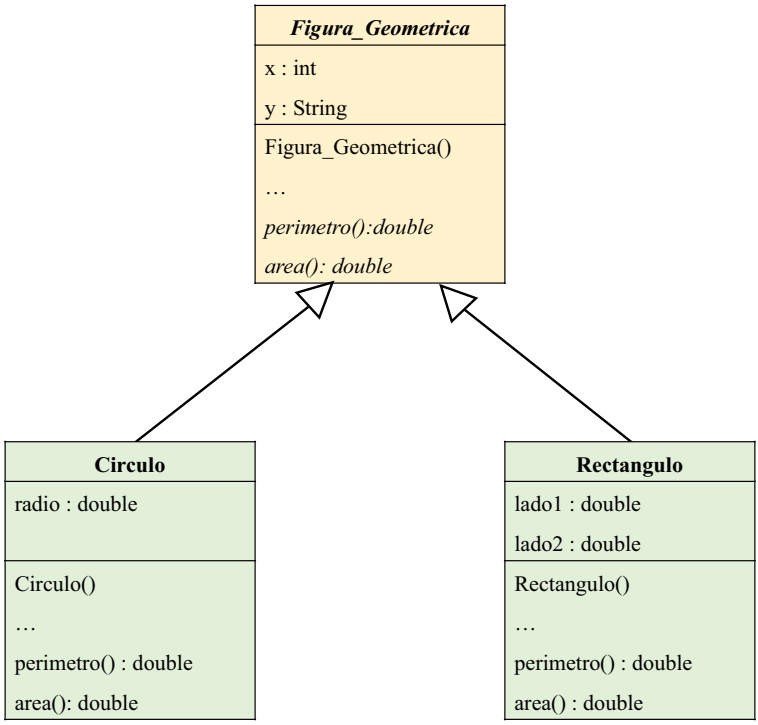
```
        public void setDiasfalta(int diasfalta) {
                this.diasfalta = diasfalta;
        }
        public double getSueldobasico() {
                return sueldobasico;
        }
        public int getMinutostardanza() {
                return minutostardanza;
        }
        public int getDiasfalta() {
                return diasfalta;
        }
        public double sueldoNeto()
        {
                return sueldobasico - 0.5*minutostardanza -
                                diasfalta*sueldobasico/30;
        }
}
```

La clase Administrativo hereda de la clase abstracta Empleado, donde el método sueldoNeto() es implementado con lo cual la clase Administrativa se pude usar para crear o instanciar objetos. El mismo modo se trabaja con la clase Docente donde el método sueldoNeto() debe ser implementado.

10.3. Programas resueltos en Java usando NetBeans

1) Se tiene el siguiente diagrama de clases.

   Construye una aplicación que permita calcular el perímetro y el área del rectángulo y del círculo.




```java
package Capitulo_II.Sesion_10;

import java.io.*;
import Capitulo_I.Sesion_06.*;

abstract class FiguraGeometrica {

    protected double x;
    protected double y;

    public FiguraGeometrica(double coordenadax, double coordenaday) {
        coordenadax = x;
        coordenaday = y;
    }

    public abstract double area();

    public abstract double perimetro();
}

class Circulo extends FiguraGeometrica {

    private double radio;

    public Circulo(double x, double y, double radio) {
        super(x, y);
        this.radio = radio;
    }
```

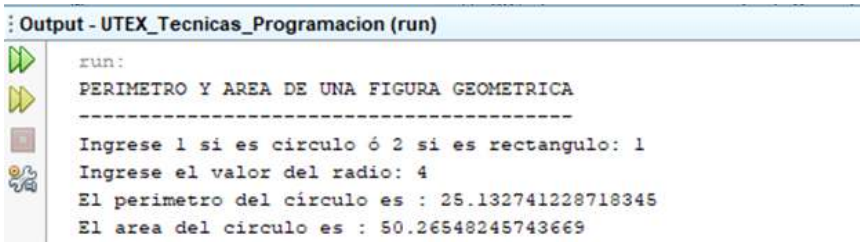
```
29
30      public double area() {
31          return Math.PI * radio * radio;
32      }
33
34      public double perimetro()
35      {
36          return 2*Math.PI*radio;
37      }
38
39  }
40
41  class Rectangulo extends FiguraGeometrica {
42
43      private double lado1;
44      private double lado2;
45
46      public Rectangulo(double coordenadax, double coordenaday, double l1, double l2) {
47          super(coordenadax, coordenaday);
48          lado1 = l1;
49          lado2 = l2;
50      }
51
52      public double area() {
53          return lado1 * lado2;
54      }
55
56      public double perimetro()
57      {
58          return 2*lado1+2*lado2;
59      }
60  }
61
62  public class UsoFigurasGeometricas {
63
64      public static void imprime(String m)
65      {
66          System.out.print(m);
67      }
68
69      public static void main(String[] args) throws IOException {
70          int tipo;
71          double lado1, lado2, radio;
72          imprime("PERIMETRO Y AREA DE UNA FIGURA GEOMETRICA\n");
73          imprime("-----------------------------------\n");
74          imprime("Ingrese 1 si es circulo ó 2 si es rectangulo: ");
75          tipo = LecturaDatos.leerDatoEntero();
76          if (tipo == 1) {
77              imprime("Ingrese el valor del radio: ");
78              radio = LecturaDatos.leerDatoReal();
79              Circulo C = new Circulo(0, 0, radio);
80              imprime("El perimetro del circulo es : "+C.perimetro()+"\n");
81              imprime("El area del circulo es : " + C.area() + "\n");
82          }
83          if (tipo == 2) {
84              System.out.print("Ingrese el valor del lado 1: ");
85              lado1 = LecturaDatos.leerDatoReal();
86              System.out.print("Ingrese el valor del lado 2: ");
87              lado2 = LecturaDatos.leerDatoReal();
88              Rectangulo R = new Rectangulo(0, 0, lado1, lado2);
89              imprime("El perimetro del rectangulo es : "+R.perimetro()+"\n");
90              System.out.println("El area del rectagulo es: " + R.area()+"\n");
91          }
92      }
93  }
```

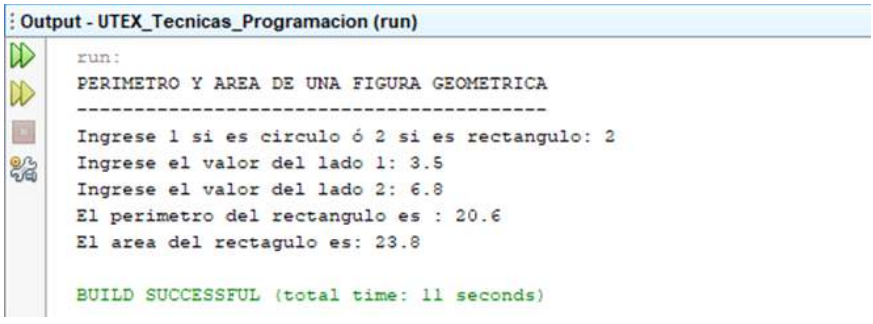
Interpretación de la programación

En el programa se tiene la clase abstracta FuguraGeometrica con dos métodos abstractos llamados perímetro y área. La clase Circulo y Rectángulo heredan de la clase abstracta FiguraGeometrica y se procede a implementar los métodos abstractos de acuerdo a los cálculos de las figuras geométricas.

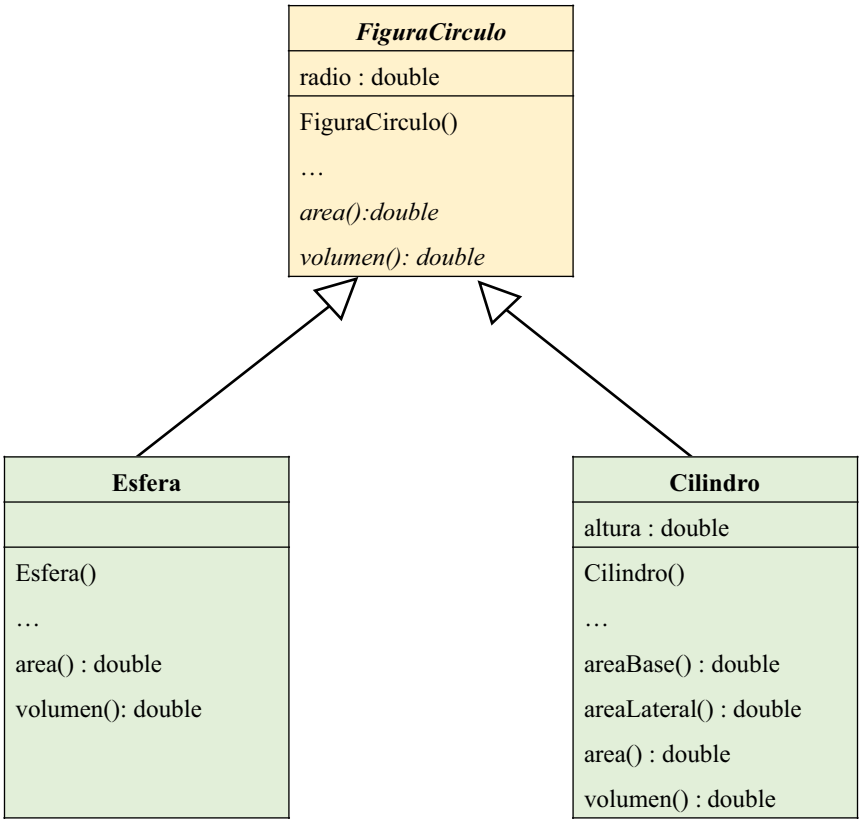
En la clase UsoFigurasGeomtricas se solicita el ingreso del tipo de figura geométrica (Circulo o Rectáingulo) y según lo seleccionado se solicita los datos para proceder a crear los objetos y mostrar los resultados invocando a los métodos implementados en la clase Circulo y Rectángulo.

Ejecución del programa:





2) Elabora una aplicación que permita el cálculo del área y volumen de una esfera y el área de la base, el área lateral, el área total y el volumen de un cilindro aplicando clase abstracta en su implementación.




```java
package Capitulo_II.Sesion_10;

import java.io.*;
import Capitulo_I.Sesion_06.*;

abstract class FiguraCirculo
{
    protected double radio;

    public FiguraCirculo(double r)
    {
        radio=r;
    }

    public void setRadio(double r)
    {
        radio=r;
    }

```

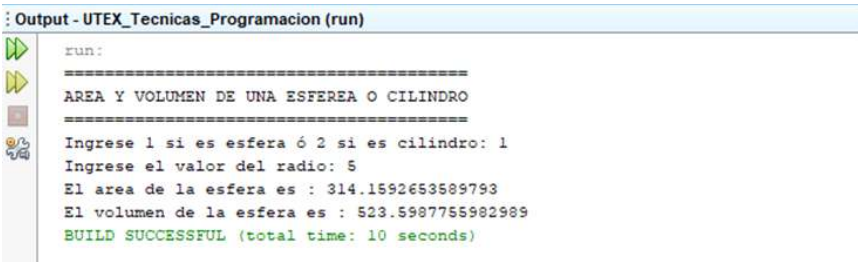
```
20          public double getRadio()
21          {
22              return radio;
23          }
24
            public abstract double volumen();
            public abstract double area();
27      }
28
29      class Esfera extends FiguraCirculo
30      {
31          public Esfera(double r)
32          {
33              super(r);
34          }
35
            public double area()
37          {
38              return 4*Math.PI*Math.pow(radio, 2);
39          }
40
            public double volumen()
42          {
43              return 4*Math.PI*Math.pow(radio, 3)/3.0;
44          }
45      }
46
47      class Cilindro extends FiguraCirculo
48      {
49          protected double altura;
50
51          public Cilindro(double r, double h)
52          {
53              super(r);
54              altura=h;
55          }
56
57          public void setAltura(double h)
58          {
59              altura=h;
60          }
61
62          public double getAltura()
63          {
64              return altura;
65          }
66
67          public double areaBase()
68          {
69              return Math.PI*radio*Math.pow(radio,2);
70          }
71
72          public double areaLateral()
73          {
74              return 2*Math.PI*radio*altura;
75          }
76
            public double area()
78          {
79              return 2*areaBase()+areaLateral();
80          }
81
            public double volumen()
83          {
84              return areaBase()*altura;
85          }
86      }
87
```

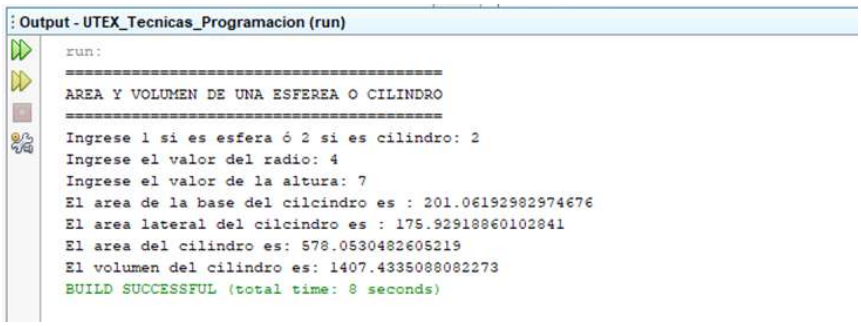
```
88  public class UsoEsferaCilindro {
89
90      public static void imprime(String m) {
91          System.out.print(m);
92      }
93
94      public static void main(String[] args) throws IOException {
95          int tipo;
96          double altura, radio;
97          imprime("=======================================\n");
98          imprime("AREA Y VOLUMEN DE UNA ESFEREA O CILINDRO\n");
99          imprime("=======================================\n");
100         imprime("Ingrese 1 si es esfera ó 2 si es cilindro: ");
101         tipo = LecturaDatos.leerDatoEntero();
102         if (tipo== 1) {
103             imprime("Ingrese el valor del radio: ");
104             radio = LecturaDatos.leerDatoReal();
105             Esfera E = new Esfera( radio);
106             imprime("El area de la esfera es : " + E.area() + "\n");
107             imprime("El volumen de la esfera es : " + E.volumen() + "\n");
108         }
109         if (tipo== 2) {
110             System.out.print("Ingrese el valor del radio: ");
111             radio = LecturaDatos.leerDatoReal();
112             System.out.print("Ingrese el valor de la altura: ");
113             altura = LecturaDatos.leerDatoReal();
114             Cilindro C = new Cilindro(radio,altura);
115             imprime("El area de la base del cilcindro es : " + C.areaBase() + "\n");
116             imprime("El area lateral del cilcindro es : " + C.areaLateral() + "\n");
117             imprime("El area del cilindro es: " + C.area() + "\n");
118             imprime("El volumen del cilindro es: " + C.volumen() + "\n");
119         }
120     }
121 }
```

Interpretación de la programación

En el programa se tiene la clase abstracta FiguraCirculo con los dos métodos abstractos área y volumen. En la clase abstracta FiguraCirculo se define el atributo radio. En la subclase Esfera se implementa los métodos área y volumen y no se define ningún atributo, por lo tanto, el método constructor de la clase Esfera invoca al método constructor de la clase abstracta FIguraCirculo. En la subclase Cilindro se implementa los métodos área y volumen, pero antes se encuentra implementado el método areaBase y areaLateral, eta programación define las instrucciones de los métodos área y volumen del cilindro. En la clase UsoEsferaCilindro se solicita primero el tipo de figura geométrica (esfera o cilindro) y se procede a solicitar los datos como el radio y en el caso del cilindro se pide adicionalmente el valor de la altura. Se instancia el objeto respectivo y se muestra los reultados.

Ejecución de la programación:





11. INTERFACES Y HERENCIA MÚLTIPLE

11.1. ¿Qué son interfaces?

Las clases abstractas para que sean como tal basta que uno de los métodos de la clase sea abstracto. En el lenguaje de programación de Java una interface es aquella en que todos sus métodos son abstractos, es decir, se podría decir que es una clase abstracta pura. Los métodos de la interface se establece los nombres, los argumentos o parámetros y el tipo de retorno dejando para después la implementación de dichos métodos. Las interfaces también pueden tener atributos pero deben ser declarados con static o final. Por lo tanto, una interface establece un protocolo o modelo a seguir en la estructura de una jerarquía de clases.

11.2. Declaración de una interface en Java

En Java una interface se utiliza la palabra reservada interface en lugar de colocar la expresión class. Se puede declarar con el modificador de acceso public o sin él, donde todos sus métodos siempre deben ser public. Una vez declarado o definido la interface, para ser utilizado en una clase se debe usar la palabra reservada implements, esta clase implementada a partir de la interface debe ser implementado los métodos provenientes de la interface. Una clase puede hacer uso de uno o varios interfaces en el momento de la implementación si existe la necesidad de hacerlo.

La sintaxis de una interface se define:

```
interface nombre_interface
{
   tipo_retorno nombre_metodo (argumentos o parámetros) ;
   . . .
}
```

Por ejemplo:

```
interface GeneraNumeros
{
        int siguienteNumero();
        void comienza();
        void comenzarEn(int n);
}

class NumerosPares implements GeneraNumeros
{
        int dato;

        public NumerosPares()
        {
                dato=0;
        }

        public int siguienteNumero()
        {
                dato=dato+2;
                return dato;
        }
```

```
        public void comienza()
        {
                dato=0;
        }

        public void comenzarEn(int n)
        {
                dato=n;
                if(n%2==1)
                dato--;
        }
}
```

La interface GeneraNumeros establece tres métodos siguienteNumero, comienza y comenzarEn. Sólo el primer método retornará el número que genere cuando este sea implementado en la clase que use esta interface. El método comenzarEn tiene un parámetro para recibir el numero a tomar en cuenta para realizar la generación de los números. La clase NumerosPares implementa cada uno de los métodos colocando el nivel de acceso public. En el método constructor se inicializa en 0 al atributo dato, en el método siguienteNumero se genera el siguiente número sumando más 2 y en el método comenzarEn es para definir el valor inicial del atributo dato, en caso este valor es impar se disminuye en 1 para poder luego generar los pares. A continuación, la clase NumerosImpares que tiene una programación similar a la clase NumerosPares.

```
class NumerosImpares implements GeneraNumeros
{
   int dato;
```

```
    public NumerosImpares()
    {
        dato=-1;
    }

    public int siguienteNumero()
    {
        dato=dato+2;
        return dato;
    }

    public void comienza()
    {
        dato=-1;
    }

    public void comenzarEn(int n)
    {
        dato=n;
        if(n%2==0)
            dato--;
    }
}
```

Si se desea implementar el método main para hacer uso de las clases mencionadas anteriormente y creadas a partir de la interface GeneraNumeros puede ser de la siguiente manera:

```
public class Generacion_Pares_Impares {

   public static void main(String args[])throws IOException
   {
      BufferedReader leer = new BufferedReader(new
                                 InputStreamReader(System.in));
      int n, tipo, i, vi;
      System.out.print("¿Generación de pares o impares? (1:Pares,
                           2:Impares): ");
      tipo=Integer.parseInt(leer.readLine());
      if(tipo==1)
      {
         System.out.print("Indique cuanto numeros pares desea
                              generar desde cero: ");
         n=Integer.parseInt(leer.readLine());
         NumerosPares NP=new NumerosPares();
         System.out.println("Los primeros "+n+" numeros pares
                                 son:");
         for(i=1;i<=n;i++)
         {
            System.out.print(NP.siguienteNumero()+"  ");
         }
         System.out.println("");
         System.out.print("Indique despues de que numero desea se
                              genere los pares: ");
         vi=Integer.parseInt(leer.readLine());
         System.out.print("Indique cuanto numeros pares desea
                              generar desde "+vi+": ");
         n=Integer.parseInt(leer.readLine());
         NP.comenzarEn(vi);
         System.out.println("Los "+n+" numeros pares generados
```

```
                        despues de "+vi+" son:");
    for(i=1;i<=n;i++)
    {
       System.out.print(NP.siguienteNumero()+"  ");
    }
    System.out.println("");
}
if(tipo==2)
{
    System.out.print("Indique cuanto numeros impares desea
                        generar desde cero: ");
    n=Integer.parseInt(leer.readLine());
    NumerosImpares NI=new NumerosImpares();
    System.out.println("Los primeros "+n+" numeros impares
                         son:");
    for(i=1;i<=n;i++)
    {
       System.out.print(NI.siguienteNumero()+"  ");
    }
    System.out.println("");
    System.out.print("Indique despues de que numero desea se
                        genere los impares: ");
    vi=Integer.parseInt(leer.readLine());
    System.out.print("Indique cuanto numeros impares desea
                         generar desde "+vi+": ");
    n=Integer.parseInt(leer.readLine());
    NI.comenzarEn(vi);
    System.out.println("Los "+n+" numeros impares generados
                        despues de "+vi+" son:");
    for(i=1;i<=n;i++)
    {
```

```
            System.out.print(NI.siguienteNumero()+"  ");
        }
        System.out.println("");
    }
  }
}
```

11.3. Referencias a interfaces

Se puede crear referencias a interfaces, pero lo que no puede crear son instancias de las interfaces. Una referencia a una interface puede recibir lo definido de un objeto, este objeto debe ser implementado de la interface.

Por ejemplo:

```
int n;
GeneraNumeros numero=new NumeroPares();
n=numero.siguienteNumero();

GeneraNumeros números=GeneraNumeros(); //no se puede
                                        instanciar (Error)
```

11.4. Uso de constantes en interfaces

Las interfaces pueden tener atributos o miembros datos que todos sean static o final, pues las interfaces al no poder ser instanciadas es una buena forma de definir grupos de constantes para luego ser utilizados en las aplicaciones.

Por ejemplo:

```
public interface Meses {
  int ENERO = 1 , FEBRERO = 2, MARZO = 3, ABRIL = 4,
```

```
        MAYO = 5, JUNIO = 6, JULIO = 7, AGOSTO = 8,
        SETIEMBRE = 9, OCTUBRE = 10, NOVIEMBRE =  11,
        DICIEMBRE = 12;

    String NOMBRES_MESES []= { " " , "Enero" , "Febrero" ,
            "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto",
            "Setiembre", "Octubre", "Noviembre", "Diciembre" };

}
```

Al momento de usarlo en una aplicación se puede escribir:

```
System.out.println("Mes de
                    "+Meses.NOMBRES_MESES[NOVIEMBRE]);
```

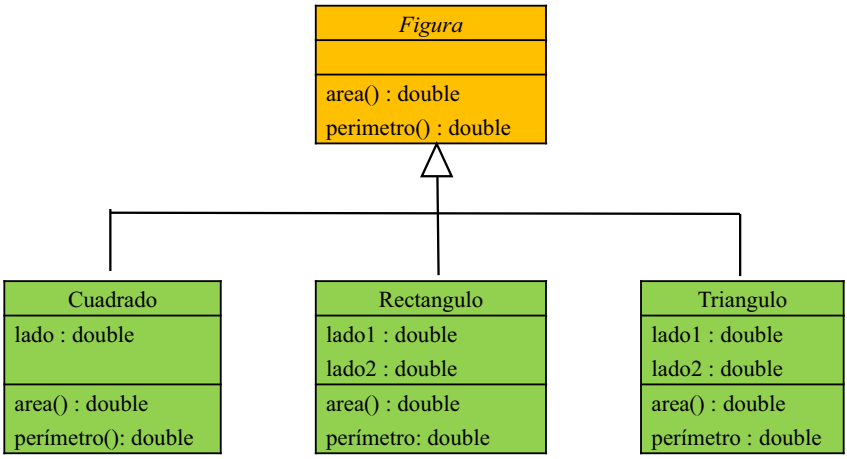
11.5. Herencia múltiple

Una clase sólo puede heredar de una clase, pero puede hacer uso en su implementación a partir de uno o varias interfaces, a esto es lo que se denomina herencia múltiple. Hay aplicaciones que son útiles la herencia a partir de una clase y de interfaces permitiendo dar soluciones a ciertos requerimientos de los usuarios.

11.6. Programas resueltos en Java usando NetBeans

1) Elabora una aplicación usando interface para calcular el área de y el perímetro del cuadrado, del rectángulo y del triángulo.

   Para dar solución a esta aplicación se tiene el siguiente diagrama de clases donde Figura es la interfaz y los métodos no implementados son área y perímetro.



La programación en Java es la siguiente:

```
1   package Capitulo_II.Sesion_11;
2
3   import java.io.*;
4   import Capitulo_I.Sesion_06.LecturaDatos;
5
6   interface Figura
7   {
8       double area();
9       double perimetro();
10  }
11
12  class Cuadrado implements Figura
13  {
14      private double lado;
15
16      public Cuadrado(double lado) {
17          this.lado = lado;
18      }
19
20      public double getLado() {
21          return lado;
22      }
23
24      public void setLado(double lado) {
25          this.lado = lado;
26      }
27
28      public double area()
29      {
30          return lado*lado;
31      }
32
33      public double perimetro()
34      {
35          return 4*lado;
36      }
37  }
```

```
38
39  class Rectangulo implements Figura
40  {
41      private double lado1;
42      private double lado2;
43
44      public Rectangulo(double lado1, double lado2) {
45          this.lado1 = lado1;
46          this.lado2 = lado2;
47      }
48
49      public double getLado1() {
50          return lado1;
51      }
52
53      public void setLado1(double lado1) {
54          this.lado1 = lado1;
55      }
56
57      public double getLado2() {
58          return lado2;
59      }
60
61      public void setLado2(double lado2) {
62          this.lado2 = lado2;
63      }
64
65      public double area()
66      {
67          return lado1*lado2;
68      }
69
70      public double perimetro()
71      {
72          return 2*lado1+2*lado2;
73      }
74  }
75
76  class Triangulo implements Figura
77  {
78      private double lado1;
79      private double lado2;
80      private double lado3;
81
82      public Triangulo(double lado1, double lado2, double lado3) {
83          this.lado1 = lado1;
84          this.lado2 = lado2;
85          this.lado3 = lado3;
86      }
87
88      public double getLado1() {
89          return lado1;
90      }
91
92      public void setLado1(double lado1) {
93          this.lado1 = lado1;
94      }
95
96      public double getLado2() {
97          return lado2;
98      }
99
```

```
100     public void setLado2(double lado2) {
101         this.lado2 = lado2;
102     }
103
104     public double getLado3() {
105         return lado3;
106     }
107
108     public void setLado3(double lado3) {
109         this.lado3 = lado3;
110     }
111
113     public double area()
114     {
115         double p, r;
116         p=(lado1+lado2+lado3)/2.0;
117         r=p+(p-lado1)*(p-lado2)*(p-lado3);
118         if(r>0)
119             return Math.sqrt(r);
120         else
121             return 0.0;
122     }
123
124     public double perimetro()
125     {
126         return lado1+lado2+lado3;
127     }
128 }
129
130 public class AreaPerimetroFiguras {
131
132     public static void imprime(String m)
133     {
134         System.out.print(m);
135     }
136
137     public static void main(String args[])throws IOException
138     {
139         int opcion;
140         double lado1, lado2, lado3;
141         imprime("=========================================\n");
142         imprime(" AREA Y PERIMETRO DE UNA FIGURA GEOMETRICA \n");
143         imprime("=========================================\n");
144         imprime("Ingres 1: Cuadrado, 2:Rectangulo, 3: Trangulo\n");
145         imprime("Opción elegida: ");
146         opcion=LecturaDatos.leerDatoEntero();
147         switch(opcion)
148         {
149             case 1:
150                 imprime("Ingrese el lado del cuadrado: ");
151                 lado1=LecturaDatos.leerDatoReal();
152                 Cuadrado C=new Cuadrado(lado1);
153                 imprime("El area del cuadrado es "+C.area()+"\n");
154                 imprime("El perimetro delcuadrado es "+C.perimetro()+"\n");
155                 break;
156             case 2:
157                 imprime("Ingrese el lado 1 del rectangulo: ");
158                 lado1=LecturaDatos.leerDatoReal();
159                 imprime("Ingrese el lado 2 del rectangulo: ");
160                 lado2=LecturaDatos.leerDatoReal();
161                 Rectangulo R=new Rectangulo(lado1,lado2);
162                 imprime("El area del rectangulo es "+R.area()+"\n");
163                 imprime("El perimetro del rectangulo es "+R.perimetro()+"\n");
                    break;
```

```
164                 case 3:
165                     imprime("Ingrese el lado 1 del triangulo: ");
166                     lado1=LecturaDatos.leerDatoReal();
167                     imprime("Ingrese el lado 2 del triangulo: ");
168                     lado2=LecturaDatos.leerDatoReal();
169                     imprime("Ingrese el lado 3 del triangulo: ");
170                     lado3=LecturaDatos.leerDatoReal();
171                     Triangulo T=new Triangulo(lado1,lado2,lado3);
172                     imprime("El area del triangulo es "+T.area()+"\n");
173                     imprime("El perimetro del triangulo es "+T.perimetro()+"\n");
174                     break;
175             }
176         }
177     }
```

Interpretación de la programación

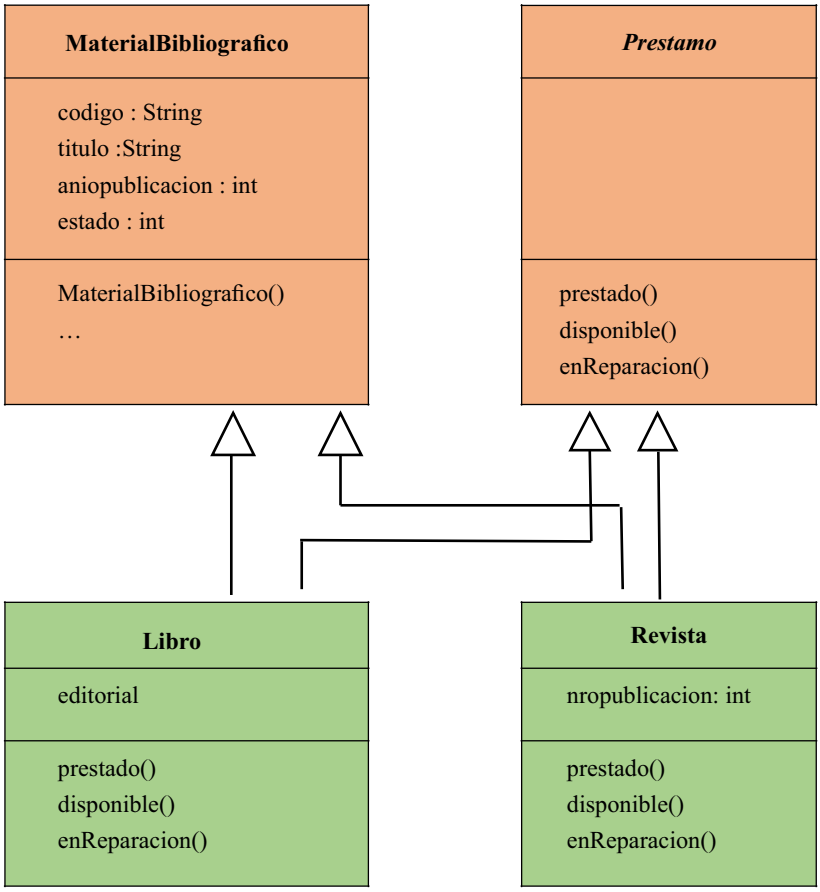
La interface Figura tiene dos métodos no implementados llamados area y perímetro. La clase Cuadrado se implementa de Figura, establece un atributo llamado lado, define los métodos constructores, getter y setter y se implementa los métodos área y perímetro que fueron declarados en la interface Figura. De la misma manera se trabaja con la clase Rectangulo y Triangulo. En el método main se solicita elegir la figura geométrica a calcular el área y el perímetro, luego se ingresa los datos para los atributos, se instancia el objeto respectivo (C para cuadrado, R para rectángulo o T para triangulo). Finalmente se imprime los resultados invocando a los métodos área y perímetro a partir del objeto instanciado.

Ejecución de la programación:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
==========================================
 AREA Y PERIMETRO DE UNA FIGURA GEOMETRICA
==========================================
Ingres 1: Cuadrado, 2:Rectangulo, 3: Trangulo
Opción elegida: 1
Ingrese el lado del cuadrado: 4.2
El area del cuadrado es 17.64
El perimetro delcuadrado es 16.8
BUILD SUCCESSFUL (total time: 7 seconds)
```

```
Output - UTEX_Tecnicas_Programacion (run)
run:
==========================================
 AREA Y PERIMETRO DE UNA FIGURA GEOMETRICA
==========================================
Ingres 1: Cuadrado, 2:Rectangulo, 3: Trangulo
Opción elegida: 3
Ingrese el lado 1 del triangulo: 4
Ingrese el lado 2 del triangulo: 7
Ingrese el lado 3 del triangulo: 5
El area del triangulo es 4.47213595499958
El perimetro del triangulo es 16.0
BUILD SUCCESSFUL (total time: 9 seconds)
```

2) Elabora una aplicación que permita determinar los estados de un material bibliográfico que son: disponible, prestado y en reparación a través del uso de una interface. Crear las clases a partir del siguiente diagrama.



La programación en Java es la siguiente:

```
1  package Capitulo_II.Sesion_11;
2
3  import java.io.*;
4  import Capitulo_I.Sesion_06.LecturaDatos;
5
6  class MaterialBibliografico
7  {
```

```
8       protected String codigo;
9       protected String titulo;
10      protected int aniopublicacion;
11      protected int estado;
12
13      public MaterialBibliografico(String codigo, String titulo, int aniopublicacion,
14              int estado) {
15          this.codigo = codigo;
16          this.titulo = titulo;
17          this.aniopublicacion = aniopublicacion;
18          this.estado = estado;
19      }
20
21      public String getCodigo() {
22          return codigo;
23      }
24
25      public void setCodigo(String codigo) {
26          this.codigo = codigo;
27      }
28
29      public String getTitulo() {
30          return titulo;
31      }
32
33      public void setTitulo(String titulo) {
34          this.titulo = titulo;
35      }
36
37      public int getAniopublicacion() {
38          return aniopublicacion;
39      }
40
41      public void setAniopublicacion(int aniopublicacion) {
42          this.aniopublicacion = aniopublicacion;
43      }
44
45      public int getEstado() {
46          return estado;
47      }
48
49      public void setEstado(int estado) {
50          this.estado = estado;
51      }
52  }
53
    interface Prestamo
55  {
        void prestado();
        void disponible();
        void enReparacion();
59  }
60
61  class Revista extends MaterialBibliografico implements Prestamo
62  {
63      private int nropublicacion;
64
65      public Revista(int nropublicacion, String codigo, String titulo,
66              int aniopublicacion, int estado) {
67          super(codigo, titulo, aniopublicacion, estado);
68          this.nropublicacion = nropublicacion;
69      }
70
71      public int getNropublicacion() {
72          return nropublicacion;
73      }
74
75      public void setNropublicacion(int nropublicacion) {
76          this.nropublicacion = nropublicacion;
77      }
```

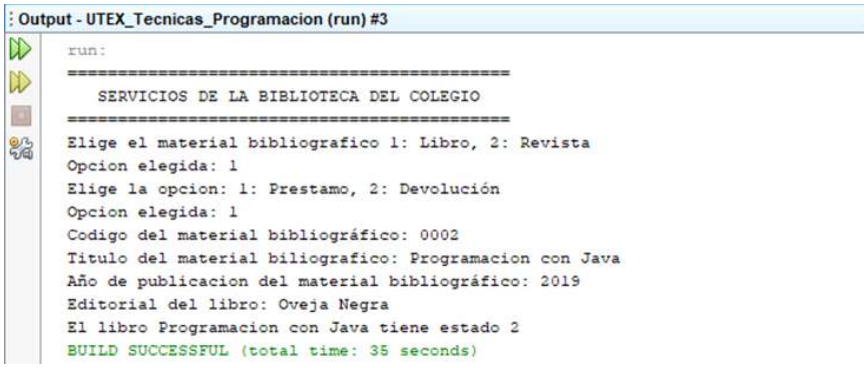
```
78
79      public void prestado()
80      {
81          super.estado=1;
82      }
83
84      public void disponible()
85      {
86          super.estado=2;
87      }
88
89      public void enReparacion()
90      {
91          super.estado=3;
92      }
93  }
94
95  class Libro extends MaterialBibliografico implements Prestamo
96  {
97      private String editorial;
98
99      public Libro(String editorial, String codigo, String titulo,
100             int aniopublicacion, int estado) {
101         super(codigo, titulo, aniopublicacion, estado);
102         this.editorial = editorial;
103     }
104
105     public String getEditorial() {
106         return editorial;
107     }
108
109     public void setEditorial(String editorial) {
110         this.editorial = editorial;
111     }
112
113     public void prestado()
114     {
115         setEstado(1);
116     }
117
118     public void disponible()
119     {
120         setEstado(2);
121     }
122
123     public void enReparacion()
124     {
125         setEstado(3);
126     }
127 }
128
129 public class Biblioteca {
130
131     public static void imprime(String m)
132     {
133         System.out.print(m);
134     }
135
136     public static void main(String args[])throws IOException
137     {
138         int opcion1, opcion2, aniopublica, nropublica, estado=3;
139         double lado1, lado2, lado3;
140         String codigo, titulo, editorial;
141         Libro L;
```

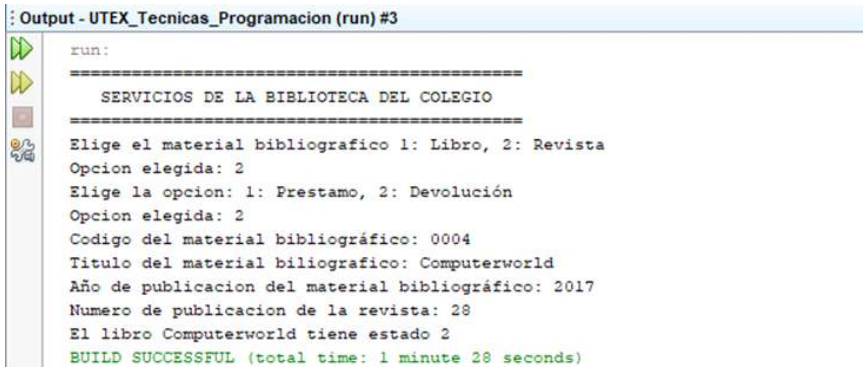
```
142             Revista R;
143             imprime("============================================\n");
144             imprime("   SERVICIOS DE LA BIBLIOTECA DEL COLEGIO \n");
145             imprime("============================================\n");
146             imprime("Elige el material bibliografico 1: Libro, 2: Revista\n");
147             imprime("Opcion elegida: ");
148             opcion1=LecturaDatos.leerDatoEntero();
149             imprime("Elige la opcion: 1: Prestamo, 2: Devolución\n");
150             imprime("Opcion elegida: ");
151             opcion2=LecturaDatos.leerDatoEntero();
152             imprime("Codigo del material bibliográfico: ");
153             codigo=LecturaDatos.leerDatoTexto();
154             imprime("Titulo del material biliografico: ");
155             titulo=LecturaDatos.leerDatoTexto();
156             imprime("Año de publicacion del material bibliográfico: ");
157             aniopublica=LecturaDatos.leerDatoEntero();
158             if(opcion1==1)
159             {
160                 imprime("Editorial del libro: ");
161                 editorial=LecturaDatos.leerDatoTexto();
162                 if(opcion2==1)
163                     estado=1;
164                 if(opcion2==2)
165                     estado=2;
166                 L=new Libro(editorial,codigo,titulo,aniopublica,2);
167                 imprime("El libro "+L.getTitulo()+ " tiene estado "+L.getEstado()+"\n");
168             }
169             if(opcion1==2)
170             {
171                 imprime("Numero de publicacion de la revista: ");
172                 nropublica=LecturaDatos.leerDatoEntero();
173                 if(opcion2==1)
174                     estado=1;
175                 if(opcion2==2)
176                     estado=2;
177                 R=new Revista(nropublica,codigo,titulo,aniopublica,estado);
178                 imprime("El libro "+R.getTitulo()+ " tiene estado "+R.getEstado()+"\n");
179             }
180         }
181     }
```

Interpretación de la programación

En esta aplicación se está usando el concepto de herencia múltiple. Se crea una superclase llamada MaterialBibliografico y una interface Préstamo que sirven para aplicar lo que se ha definido como herencia múltiple. La clase revista y la clase Libro hereda de la clase MaterialBibliográfico y se implementa de la interface Préstamo. Los métodos denominados prestado, disponible y enReparación se implementa en las subclases Libro y Revista. En el método main se hace uso de las subclases instanciado el objeto respectivo, luego de solicitar el ingreso del tipo de material bibliográfico como la indicación de prestar o devolver dicho material.

Ejecución de la programación:





12. POLIMORFISMO

12.1. ¿Qué es polimorfismo?

El polimorfismo es la capacidad que tienen los objetos en usar valores y comportamientos de diferentes tipos logrado ejecutar operaciones definidas en otras clases. Se pueden implementar programas que procesen o ejecuten objetos de aquellas clases creadas a partir de una jerarquía. Cuando las clases pertenecen a una jerarquía entonces se proceder a ejecutar lo definido en sus valores y operaciones (métodos) en la misma estructura de la clase. Por ejemplo, se tiene la clase abstracta Empleado que tiene el método abstracto ingresos. A continuación, se tiene la programación de la clase abstracta Empleado

```
import java.io.*;

abstract class Empleado
{
        protected String apellidos;
        protected String nombres;

        public Empleado(String ape, String nom)
        {
                apellidos=ape;
                nombres=nom;
        }
```

```
        public void setApellidos(String ape)
        {
                apellidos=ape;
        }

        public void setNombres(String nom)
        {
                nombres = nom;
        }

        public String getApellidos()
        {
                return apellidos;
        }

        public String getNombres()
        {
                 return nombres;
        }

         abstract double ingresos();
}
```

A partir de la clase abstracta Empleado se aplica herencia para crear las clases Gerente, EmpleadoGanaComision, EmpleadoXTarea y EmpleadoPorHora.

```
class Gerente extends Empleado
{
        public double salario;

        public Gerente(String ape, String nom, double s)
        {
                super(ape,nom);
                salario=s;
```

```
        }

        public void setSalario(double s)
        {
                salario=s;
        }

        public double getSalario()
        {
                return salario;
        }

        public double ingresos()
        {
                return salario;
        }

        public String toString()
        {
                return "El gerente se llama "+nombres+"
                        "+apellidos;
        }
}

class EmpleadoGanaComision extends Empleado
{
        private double salarioBasico;
        private double comisionXProducto;
        private int cantidadProductos;

        public EmpleadoGanaComision(String ape, String nom,
                        double sb, double comision, int cantpro)
        {
                super(ape,nom);
                salarioBasico=sb;
```

```
            comisionXProducto=comision;
            cantidadProductos=cantpro;
    }

    public void setSalarioBasico(double sb)
    {
            salarioBasico=sb;
    }

    public void setComisionXProducto(double comision)
    {
            comisionXProducto=comision;
    }

    public void setCantidadProductos(int cantpro)
    {
            cantidadProductos=cantpro;
    }

    public double getSalarioBasico()
    {
            return salarioBasico;
    }

    public double getComisionXProducto()
    {
            return comisionXProducto;
    }

    public int getCantidad()
    {
            return cantidadProductos;
    }

    public double ingresos()
```

```
        {
                return salarioBasico+
                        comisionXProducto*cantidadProductos;
        }

        public String toString()
        {
                return "Empleado que gana por Comision se llama
                        "+nombres+"  "+apellidos;
        }
}

class EmpleadoXTarea extends Empleado{
        private double salarioXTareaCumplida;
        private int cantidadTarea;

        public EmpleadoXTarea(String ape, String nom, double st,
                                int cant)
        {
                super(ape,nom);
                salarioXTareaCumplida=st;
                cantidadTarea=cant;
        }

        public void setSalarioXTareaCumplida(double st)
        {
                salarioXTareaCumplida = st;
        }

        public void setCantidadTarea(int cant)
        {
                cantidadTarea=cant;
        }

        public double getSalarioXTareaCumplida()
        {
```

```
                return salarioXTareaCumplida;
        }

        public double getCantidadTarea()
        {
                return cantidadTarea;
        }

        public double ingresos()
        {
                return salarioXTareaCumplida*cantidadTarea;
        }

        public String toString()
        {
                return "Empleado por cumplimiento de Tareas se
                        llama "+nombres+"  "+apellidos;
        }
}

class EmpleadoPorHora extends Empleado
{
        protected double pagoPorHora;
        protected double horasTrabajadas;

        public EmpleadoPorHora(String ape, String nom, double
                                        ph, double ht)
        {
                super(ape,nom);
                pagoPorHora= ph;
                horasTrabajadas=ht;
        }

        public void setPagoPorHora(double sh)
        {
                pagoPorHora=sh;
```

```
    }

    public void setHorasTrabajadas(double ht)
    {
        horasTrabajadas=ht;
    }

    public double getPagoPorHora()
    {
        return pagoPorHora;
    }

    public double getHorasTrabajadas()
    {
        return horasTrabajadas;
    }

    public double ingresos()
    {
        return pagoPorHora*horasTrabajadas;
    }

    public String toString()
    {
        return "Empleado que gana por horas se llama
            "+apellidos+"  "+nombres;
    }
}
```

Con las subclases creadas a partir de la clase abstracta Empleado se procede a crear la clase UsandoEmpleado que tiene al método estático main

```
public class UsandoEmpleado{
    public static void imprimir(String m)
    {
```

```
        System.out.print(m);
}

public static void main(String args[])
{
        BufferedReader leer=new BufferedReader(new
                        InputStreamReader (System.in));
        String ape,nom;
        int c, ch;
        double salario, sb, comision, ph, st;
        Empleado E;
        imprimir("============================\n");
        imprimir("  DATOS DE EMPLEADO - GERENTE\n");
        imprimir("============================\n");
        imprimir("Ingrese el apellido: ");
        ape=leer.readLine();
        imprimir("Ingrese el nombre: ");
        nom=leer.readLine();
        imprimir("Ingrese el salario: ");
        salario=Double.parseDouble(leer.readLine());
        Gerente G = new Gerente(ape, nom, salario);
        imprimir("============================\n");
        imprimir("  DATOS DE EMPLEADO QUE GANA POR
                COMISION\\n");
        imprimir("============================\n");
        imprimir("Ingrese el apellido: ");
        ape=leer.readLine();
        imprimir("Ingrese el nombre: ");
        nom=leer.readLine();
        imprimir("Ingrese el salario básico: ");
        sb=Double.parseDouble(leer.readLine());
        imprimir("Ingrese la comisión por producto: ");
        comision=Double.parseDouble(leer.readLine());
        imprimir("Ingrese la cantidad de productos
                vendidos: ");
        c=Integer.parseInt(leer.readLine());
```

```
EmpleadoGanaComision C=new
          EmpleadoGanaComision(ape,nom,sb,comision,c);
imprimir("============================\n");
imprimir("  DATOS DE EMPLEADO QUE GANA POR
            TAREAS\n");
imprimir("============================\n");
imprimir("Ingrese el apellido: ");
ape=leer.readLine();
imprimir("Ingrese el nombre: ");
nom=leer.readLine();
imprimir("Ingrese el salario por tarea cumplida: ");
st=Double.parseDouble(leer.readLine());
imprimir("Ingrese la cantidad de tareas realizadas: ");
c=Integer.parseInt(leer.readLine());
EmpleadoXTarea T=new
                    EmpleadoXTarea(ape,nom,st,c);
imprimir("============================\n");
imprimir("  DATOS DE EMPLEADO QUE GANA POR
            HORAS\n");
imprimir("============================\n");
imprimir("Ingrese el apellido: ");
ape=leer.readLine();
imprimir("Ingrese el nombre: ");
nom=leer.readLine();
imprimir("Ingrese el pago por hora: ");
ph=Double.parseDouble(leer.readLine());
imprimir("Ingrese la cantidad de horas trabajadas: ");
ch=Integer.parseInt(leer.readLine());
EmpleadoPorHora H=new
                  EmpleadoPorHora(ape,nom,ph,ch);

E = G;
imprimir(E.toString() + " y percibe " + E.ingresos()+"\n");

E = C;
imprimir(E.toString() + " y percibe " + E.ingresos()+"\n");
```

```
                E = T;
                imprimir(E.toString() + " y percibe " + E.ingresos()+"\n");

                E = H;
                imprimir(E.toString() + " y percibe " + E.ingresos()+"\n");
        }
}
```

En la línea de programación E = G; la referencia E declarada a partir de la clase abstracta Empleado recibe al objeto G de la subclase Gerente, aquí se presenta un comportamiento polimórfico. La instrucción E.toString() invoca a la ejecución del método toString() del objeto al que E hace referencia, es decir, se ejecuta el método toString() de la clase Gerente. La instrucción E.ingresos() solicita la ejecución del método ingresos del objeto al que E hace referencia y por lo tanto ejecuta lo definido en la clase Gerente. Luego, en la instrucción o línea de programación E = C; E hace referencia al objeto C que fue creado a partir de la clase EmpleadoGanaComision por lo que E puede hacer uso de los métodos ingresos y toString.

12.2. Tipos de polimorfismo

Existen varios tipos de polimorfismo tales como:

a) Polimorfismo de asignación, que consiste en que una variable hace referencia a más de un tipo de clase, tal como el ejemplo anterior que se trató de la clase abstracta Empleado y se creó las subclases para luego en el método main se haga uso de referencias.

b) Polimorfismo de sobrecarga, donde dos o más funciones tienen el mismo nombre, pero la lita de argumentos es distinta.

c) Polimorfismo de inclusión, que consiste en redefinir un método por completo en su implementación por lo que será distinto su funcionamiento en la superclase como en la subclase.

12.3. Programas resueltos en Java usando NetBeans

1) Crea una aplicación que permita calcular el área y el volumen de un cilindro y de una esfera aplicando sobreescritura de métodos, clase abstracta, herencia y polimorfismo.

```
1   package Capitulo_II.Sesion_12;
2
3   import Capitulo_I.Sesion_06.LecturaDatos;
4   import java.io.*;
5
6   abstract class FiguraGeo {
7
8       public double area() {
9           return 0;
10      }
11
12      public double volumen() {
13          return 0;
14      }
15
16      public abstract String obtenerNombre();
17  }
18
19  class Punto extends FiguraGeo {
20
21      protected double x;
22      protected double y;
23
24      public Punto(double a, double b) {
25          x = a;
26          y = b;
27      }
28
29      public void setX(double a) {
30          x = a;
31      }
32
33      public void setY(double b) {
34          y = b;
35      }
36
37      public double getX() {
38          return x;
39      }
40
41      public double getY() {
42          return y;
43      }
44
45      public String obtenerNombre() {
46          return "Punto";
47      }
48
49      public String toString() {
50          return "(" + x + "," + y + ")";
51      }
52
53  }
54
```

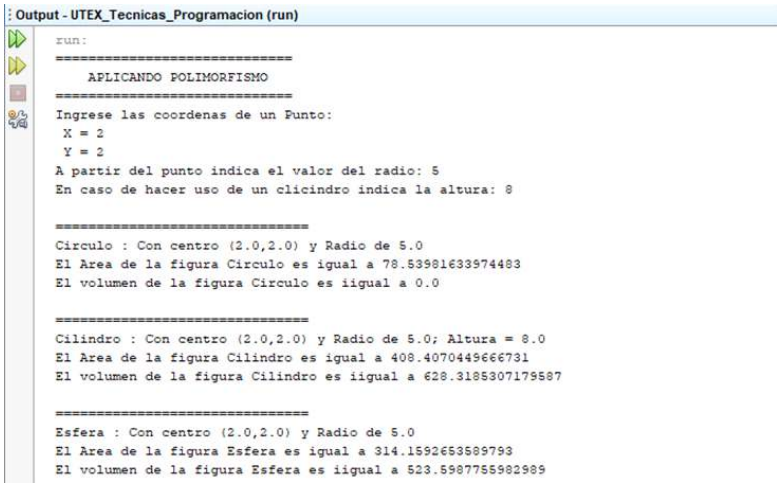
```
class Circulo extends Punto {

    protected double radio;

    public Circulo(double a, double b, double r) {
        super(a, b);
        radio = r;
    }

    public void setRadio(double r) {
        radio = r;
    }

    public double getRadio() {
        return radio;
    }

    public double area() {
        return Math.PI * Math.pow(radio, 2);
    }

    public String obtenerNombre() {
        return "Circulo";
    }

    public String toString() {
        return "Con centro " + super.toString() + " y Radio de " + radio;
    }
}

class Cilindro extends Circulo {

    protected double altura;

    public Cilindro(double a, double b, double r, double h) {
        super(a, b, r);
        altura = h;
    }

    public void setAltura(double h) {
        altura = h;
    }

    public double getAltura() {
        return altura;
    }

    public double area() {
        return super.area() * 2 + 2 * Math.PI * radio * altura;
    }

    public double volumen() {
        return super.area() * altura;
    }

    public String obtenerNombre() {
        return "Cilindro";
    }

    public String toString() {
        return super.toString() + "; Altura = " + altura;
    }
}
```

```
119   class Esfera extends Circulo {
120
121       public Esfera(double a, double b, double r) {
122           super(a, b, r);
123       }
124
126       public double area() {
127           return super.area() * 4;
128       }
129
130       public double volumen() {
131           return super.area() * radio * 4 / 3;
132       }
133
134       public String obtenerNombre() {
135           return "Esfera";
136       }
137
138   }
139   public class UsandoFiguraGeo {
140
141       public static void imprimir(String m)
142       {
143           System.out.print(m);
144       }
145
146       public static void main(String args[]) throws IOException {
147           double x, y, r, h;
148           imprimir("============================\n");
149           imprimir("    APLICANDO POLIMORFISMO\n");
150           imprimir("============================\n");
151           imprimir("Ingrese las coordenas de un Punto:\n");
152           imprimir(" X = ");
153           x=LecturaDatos.leerDatoReal();
154           imprimir(" Y = ");
155           y=LecturaDatos.leerDatoReal();
156           imprimir("A partir del punto indica el valor del radio: ");
157           r=LecturaDatos.leerDatoReal();
158           imprimir("En caso de hacer uso de un clicindro indica la altura: ");
159           h=LecturaDatos.leerDatoReal();
160           Punto P = new Punto(x, y);
161           Circulo C = new Circulo(x, y, r);
162           Cilindro CI = new Cilindro(x, y, r, h);
163           Esfera E=new Esfera(x,y,r);
164           FiguraGeo Figuras[] = new FiguraGeo[4];
165           Figuras[0] = P;
166           Figuras[1] = C;
167           Figuras[2] = CI;
168           Figuras[3] = E;
169           imprimir("\n");
170           for (int i = 1; i < Figuras.length; i++) {
171               imprimir("============================\n");
172               imprimir(Figuras[i].obtenerNombre() + " : " + Figuras[i]+"\n");
173               imprimir("El Area de la figura "+Figuras[i].obtenerNombre()+
174                       " es igual a " + Figuras[i].area()+"\n");
175               imprimir("El volumen de la figura "+Figuras[i].obtenerNombre()+
176                       " es iigual a " + Figuras[i].volumen()+"\n");
177               imprimir("\n");
178           }
179       }
180   }
```

Interpretación de la programación

La clase abstracta FiguraGeo define 3 métodos: área, volumen y obtenerNombre, este último es el método abstracto. A partir de la clase abstracta FiguraGeo se crea por herencia las clases Punto, Circulo, Cilindro y Esfera. Se establece una jerarquía de clases que se aprecia cuando la clase Punto hereda de FiguraGeo, la clase Cilindro hereda de Punto y la clase Cilindro hereda de la clase Circulo. Los métodos de area y volumen se implementan en las subclases y se implementa el método obtenerNombre. La clase Esfera hereda de la clase Circulo. Luego, al momento de hacer uso de las clases antes mencionadas en el método main se procede a solicitar las coordenadas de un punto, luego el valor de un radio y la altura de un cilindro. Se crea los objetos P, C, CI y E a partir de los datos ingresados. El polimorfismo se aprecia desde el momento que se crea una arreglo Figuras[] de tamaño 4 a partir de la clase abstracta FiguraGeo y luego cada elemento del arreglo recibe los objetos P, C, CI y E, haciendo que el arreglo haga referencias a cada uno de los objetos antes mencionado. A través de una sentencia repetitiva for se procede a mostrar información de cada figura usando los métodos implementados dado que el arreglo hace referencia a cada objeto teniendo un comportamiento polimórfico.

Ejecución de la programación:



2) Crea una aplicación que permita calcular el perímetro y el área de polígonos como el cuadrado, rectángulo y triángulo aplicando clase abstracta, herencia de clases y polimorfismo.

```
1   package Capitulo_II.Sesion_12;
2
3   import Capitulo_I.Sesion_06.LecturaDatos;
4   import java.io.*;
5
7   abstract class Poligono
8   {
9       protected int nrolados;
10
11      public Poligono(int lados)
12      {
13          nrolados=lados;
14      }
15
16      public void setNroLados(int lados)
17      {
18          nrolados=lados;
19      }
20
21      public int getNroLados()
22      {
23          return nrolados;
24      }
25
26      public String toString()
27      {
28          return "Poligono de "+nrolados+" lados";
29      }
30
31      public abstract String obtenerNombre();
32      public abstract double perimetro();
33      public abstract double area();
34  }
35
36  class Cuadrado extends Poligono
37  {
38      private double lado;
39
40      public Cuadrado(double l)
41      {
42          super(4);
43          lado=l;
44      }
45
46      public void setLado(double l)
47      {
48          lado=l;
49      }
50
51      public double getLado()
52      {
53          return lado;
54      }
55
56      public String obtenerNombre()
57      {
58          return "Cuadrado";
59      }
```

```
59
60          public String toString()
61          {
62              return super.toString()+ " iguales se llama "+obtenerNombre();
63          }
64
65          public double perimetro()
66          {
67              return 4*lado;
68          }
69
70          public double area()
71          {
72              return lado*lado;
73          }
74      }
75
76      class Rectangulo extends Poligono
77      {
78          private double lado1;
79          private double lado2;
80
81          public Rectangulo(double l1, double l2)
82          {
83              super(4);
84              lado1=l1;
85              lado2=l2;
86          }
87
88          public void setLado1(double l1)
89          {
90              lado1=l1;
91          }
92
93          public void setLado2(double l2)
94          {
95              lado2=l2;
96          }
97
98          public double getLado1()
99          {
100             return lado1;
101         }
102
103         public double getLado2()
104         {
105             return lado2;
106         }
107
108         public String obtenerNombre()
109         {
110             return "Rectangulo";
111         }
112
113         public String toString()
114         {
115             return super.toString()+" con dos pares de lados iguales se llama "+
116                     obtenerNombre();
117         }
118
119         public double perimetro()
120         {
121             return 2*lado1+2*lado2;
122         }
123
```

```
    public double area()
    {
        return lado1*lado2;
    }
}

class Triangulo extends Poligono
{
    private double lado1;
    private double lado2;
    private double lado3;

    public Triangulo(double l1, double l2, double l3)
    {
        super(3);
        lado1=l1;
        lado2=l2;
        lado3=l3;
    }

    public void setLado1(double l1)
    {
        lado1=l1;
    }

    public void setLado2(double l2)
    {
        lado2=l2;
    }

    public void setLado3(double l3)
    {
        lado2=l3;
    }

    public double getLado1()
    {
        return lado1;
    }

    public double getLado2()
    {
        return lado2;
    }

    public double getLado3()
    {
        return lado3;
    }

    public String obtenerNombre()
    {
        return "Triangulo";
    }

    public String toString()
    {
        return super.toString()+ " y se llama "+obtenerNombre();
    }

    public double perimetro()
    {
        return lado1+lado2+lado3;
    }
```

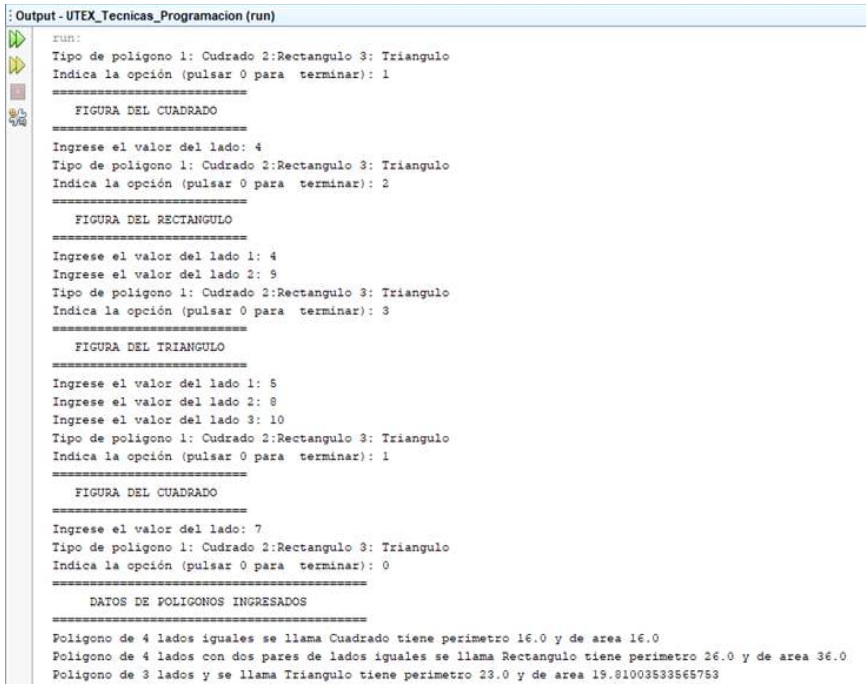
```
188
189         public double area()
190         {
191             double p, r;
192             p=(lado1+lado2+lado3)/2;
193             r=p*(p-lado1)*(p-lado2)*(p-lado3);
194             if(r>0)
195                 return Math.sqrt(r);
196             else
197                 return 0.0;
198         }
199     }
200
201     public class UsandoPoligono {
202
203         static Poligono poligonos[]=new Poligono[20];
204         static int cuenta=0;
205
206         public static void imprimir(String m)
207         {
208             System.out.print(m);
209         }
210
211         public static void main(String args[])throws IOException
212         {
213             leerTipoPoligono();
214             mostrarDatosPoligonos();
215         }
216
217         public static void leerTipoPoligono()throws IOException {
218             int tipo;
219             do {
220                 do {
221                     imprimir("Tipo de poligono 1: Cudrado 2:Rectangulo 3: Triangulo \n");
222                     imprimir("Indica la opción (pulsar 0 para  terminar): ");
223                     tipo = LecturaDatos.leerDatoEntero();
224                 } while (tipo < 0 || tipo > 3);
225                 if (tipo != 0) {
226                     switch (tipo) {
227                         case 1:
228                             leerCuadrado();
229                             break;
230                         case 2:
231                             leerRectangulo();
232                             break;
233                         case 3:
234                             leerTriangulo();
235                             break;
236                     }
237                 }
238             } while (tipo != 0);
239         }
240
241         public static void leerCuadrado()throws IOException
242         {
243             double lado;
244             imprimir("==========================\n");
245             imprimir("   FIGURA DEL CUADRADO\n");
246             imprimir("==========================\n");
247             do
248             {
249                 imprimir("Ingrese el valor del lado: ");
250                 lado=LecturaDatos.leerDatoReal();
251             }while(lado<=0);
252             Cuadrado C=new Cuadrado(lado);
```

```
253             poligonos[cuenta]=C;
254             cuenta++;
255         }
256
257         public static void leerRectangulo() throws IOException {
258             double lado1, lado2;
259             imprimir("=========================\n");
260             imprimir("   FIGURA DEL RECTANGULO\n");
261             imprimir("=========================\n");
262             do
263             {
264                 imprimir("Ingrese el valor del lado 1: ");
265                 lado1=LecturaDatos.leerDatoReal();
266             }while(lado1<=0);
267             do
268             {
269                 imprimir("Ingrese el valor del lado 2: ");
270                 lado2=LecturaDatos.leerDatoReal();
271             }while(lado2<=0);
272             Rectangulo R=new Rectangulo(lado1,lado2);
273             poligonos[cuenta]=R;
274             cuenta++;
275         }
276
277         public static void leerTriangulo() throws IOException {
278             double lado1, lado2, lado3;
279             imprimir("=========================\n");
280             imprimir("   FIGURA DEL TRIANGULO\n");
281             imprimir("=========================\n");
282             do
283             {
284                 imprimir("Ingrese el valor del lado 1: ");
285                 lado1=LecturaDatos.leerDatoReal();
286             }while(lado1<=0);
287             do
288             {
289                 imprimir("Ingrese el valor del lado 2: ");
290                 lado2=LecturaDatos.leerDatoReal();
291             }while(lado2<=0);
292             do
293             {
294                 imprimir("Ingrese el valor del lado 3: ");
295                 lado3=LecturaDatos.leerDatoReal();
296             }while(lado3<=0);
297             Triangulo T=new Triangulo(lado1,lado2, lado3);
298             poligonos[cuenta]=T;
299             cuenta++;
300         }
301
302         public static void mostrarDatosPoligonos()
303         {
304             imprimir("=======================================\n");
305             imprimir("     DATOS DE POLIGONOS INGRESADOS\n");
306             imprimir("=======================================\n");
307             int i;
308             for(i=0;i<cuenta;i++)
309             {
310                 imprimir(poligonos[i] +" tiene perimetro "+poligonos[i].perimetro()
311                         +" y de area "+poligonos[i].area()+"\n");
312             }
313         }
314     }
```

Interpretación de la programación

La clase abstracta Polígono tiene 3 métodos abstractos: obtenerNombre, perímetro y área. A partir de la clase Polígono se aplica herencia para crear las subclases Cuadrado, Rectángulo y Triangulo donde se implementa los métodos abstractos definidos en la clase abstracta Polígono. La clase Usando Polígono se crea varios métodos estáticos, uno de ellos es el método leerTipoPoligono donde se implementa la selección de tipo de polígono y según el tipo se invoca a métodos como leerCuadrado, leerRectángulo y leerTríangulo. En la clase UsandoPoligono se crea un arreglo estático llamado polígonos a partir de la clase Polígono y una variable de memoria entera estática cuenta con valor inicial 0. El programa está elaborado para poder ingresar los datos de 20 polígonos distintos. El programa culmina en mostrar los datos del perímetro y el área de los polígonos ingresados donde se aprecia el uso del polimorfismo.

Ejecución del programa:



13. PAQUETES Y EXCEPCIONES

13.1. ¿Qué es un paquete?

Un paquete es la agrupación de clases que tienen alguna afinidad. En otros lenguajes de programación lo tienen bajo el nombre de librerías. Cuando se define una clase, esta debe pertenecer a un paquete. La implementación de la clase puede hacer uso de la implementación de otras clases del mismo paquete o de otros paquetes. Toda clase tiene un nombre y debe ser único dentro del paquete, pudiendo existir dos clases del mismo nombre siempre y cuando se encuentren en paquetes distintos.

La clausula package se logra indicar el paquete donde la clase pertenece, siendo la sintaxis:

package <nombre del paquete>;

Cuando se hace uso de la clausula package, esta debe ser la primera sentencia en la elaboración de un programa. Por ejemplo:

```
package mipaquete;
class Aritmetica {
}
```

Aquí se puede apreciar que la clase Aritmetica pertenece al paquete mipaquete. El uso de la cláusula package es opcional, el hecho de no usarlo implica que la clase pertenece a un paquete por defecto que no tiene nombre.

13.2. La claúsula import

En un paquete si existe la clase A y la clase B y se desea hacer la implementación de la clase C en el mismo paquete, en la programación de la clase C se puede usar lo programado en la clase A y la clase B dado que todo se está implementando en un único paquete. Por ejemplo:

```
package Matematica;
. . .
class Cilindro {
   Circulo circulo;
   . . .
}
```

En el paquete Matemática se desea implementar la clase Cilindro, donde se declara una variable a partir de la clase Circulo. Para hacer uso de esta clase sin ningún problema se debe tener implementado la clase Circulo en el paquete Matemática. En caso que la clase Cilindro se define en otro paquete entonces se debe usar la cláusula import. Siguiendo el ejemplo anterior la cláusula import se puede usar de la siguiente manera:

```
package Geometría;
class Círculo {
   private int radio;
   public Círculo(int r)
   {
         radio=r;
   }
}
```

Ahora para hacer uso de la clase Círculo en la clase Cilindro se debe escribir lo siguiente:

```
package Matematica;

import Geometria.*;

class Cilindro {
   Circulo circulo;
   . . .
}
```

Con la cláusula import Geometría.*; se puede hacer uso de todos las clases e interfaces del paquete y también de sus miembros. Si la intención es sólo usar la clase Circulo entonces se puede colocar la siguiente declaración: import Geométria.Círculo;

En la programación con paquetes es posible acceder mencionado el nombre del paquete seguido de la clase que se desea usar así, por ejemplo

```
package Matemática;

class Cilindro {
   Geometria.Círculo círculo;
   . . .
}
```

Sin embargo, si no se usa la cláusula import es necesario especificar el nombre del package cada vez que se desee usar la clase Circulo. La cláusula import se puede hacer uso en un mismo programa varias veces. Se usa después de la declaración de la palabra reservada package y antes de la definición de las clases. El compilador de Java cuando no encuentra la clase en el mismo paquete o paquete actual busca en los paquetes indicados a través de import. Hacer uso de import no implica copiar código hacia el nuevo programa sino más bien una extensión del programa actual hacia aquellas clases o interfaces que se encuentran en otros paquetes.

13.3. Paquetes existentes en Java

En el lenguaje de programación de Java existe paquetes que tienen definidos muchas clases o interfaces que son de utilidad en muchas aplicaciones. Los paquetes más usados son:

a) java.awt, provee de clases para la construcción de interfaces gráficas, es decir, para aplicaciones visuales.
b) java.swing, proporciona al igual del paquete awt las clases o interfaces para la elaboración de aplicaciones visuales.
c) java.io, proporciona de clases e interfaces para el manejo de datos de entrada y salida en las aplicaciones
d) java.net, proporciona todas las clases para el trabajo en red.
e) Java.uitl, es el paquete para determinar la forma de leer los datos a partir de lo indicado en las instrucciones del programa.

13.4. Los nombres de los paquetes

Los paquetes se les puede nombrar con nombres compuestos, los mismos que pueden ser separados por puntos, así por ejemplo mispaqutes.Matematicas.Geometria, esto se interpreta que el subpaquete Geometria está dentro del subpaquete Matematicas y esta a su vez al paquete mispaquetes. Es así que se establece una jerarquía a nivel de paquetes.

Los paquetes establecen una ruta física donde se encuentra las clases que una vez compilado se genera los archivos de extensión class, asi por ejemplo si tengo un paquete llamado Acceso y dentro de ella una clase llamada Operaciones, deberá existir después de una compilación el archivo Operaciones.class dentro de una carpeta Acceso. Además, si una clase no es declarada con modificador de acceso public solo puede ser utilizada por clases que pertenezcan al mismo paquete.

13.5. Manejo de Excepciones

Los programas es un conjunto de instrucciones que siguen una lógica de ejecución, pero durante esta ejecución puede presentarse

errores que no lo ocasionan dichas instrucciones sino por diversos factores, como por ejemplo un dato mal ingresado por el usuario. Si ocurre un error el programa dejará de ejecutar y mostrará el error no pudiendo tener la posibilidad de corregir lo ingresado. Es por ello que se necesita interceptar el error para tener la opción de corregir y que el programa prosiga en la ejecución. Esto se puede lograr en el lenguaje de programación de Java a través del uso de excepciones.

La excepción se puede entender como un suceso excepcional y como suceso vendría ser un evento que sucede durante la ejecución del programa ocasionando el normal desarrollo de las instrucciones. Cuando sucede el error el lenguaje de programación Java crea un objeto de tipo exception y lo maneja aparte del flujo normal de instrucciones del programa. El objeto de tipo exception maneja información sobre el error ocurrido y puede ser tratado a nivel de programación a través de un catch. En el catch se establece el tipo de excepción y se programa para ese tipo de excepción. Puede existir muchas excepciones y por lo tanto debe haber muchos catch. Cuando ocurre el error y detecta la excepción ocurrida recurre al catch correspondiente, ejecuta lo programado y vuelve a la siguiente instrucción de donde se lanzó la excepción. En el caso que el error ocurrido no tiene el programa apropiado para dicha excepción entonces el programa se detiene y no puede continuar con la ejecución de las instrucciones.

```
import Capitulo_I.Sesion_06.LecturaDatos;
import java.io.*;

public class Excepcion_Aritmetica
{
   public static void imprimir(String m)
   {
      System.out.print(m);
   }
```

```
   public static void main(String args[])throws IOException
   {
      int a, b;
      imprimir("============================\n");
      imprimir("   OPERACIONES BASICAS\n");
      imprimir("============================\n");
      try
      {
         imprimir("Ingrese el primer numero: ");
         a=LecturaDatos.leerDatoEntero();
         imprimir("Ingrese el segundo numero: ");
         b=LecturaDatos.leerDatoEntero();
         imprimir("La suma de los dos numeros es: "+(a+b)+"\n");
         imprimir("La resta de los dos numeros es: "+(a-b)+"\n");
         imprimir("La multiplicacion de los dos numeros es:
                     "+(a*b)+"\n");
         imprimir("La division de los dos numeros es: "+(a/b)+"\n");
      }
      catch(ArithmeticException e)
      {
         imprimir("No se puede dividir entre CERO\n");
      }
   }
}
```

En el programa anterior se solicita el ingreso de dos números enteros y en caso que el segundo número es CERO entonces ocurrirá un error en la línea de instrucción siguiente:

Imprimir ("La division de los dos numeros es: "+(a/b)+"\n");

La división entre cero no es posible, por lo que esta instrucción lanzará una excepción y será capturada por catch con el tipo de excepción ArithmeticException que imprimirá un mensaje indicado que no se puede dividir entre cero.

13.6. Tipos de Excepciones

Hay muchas excepciones en Java y así tenemos, por ejemplo:

a) ArithmeticException, excepción que se crea a partir de un error aritmético como es el caso de la división entre cero.

b) NumberFormatException, excepción que se dispara cuando ocurre un error que vaya en contra al tipo de dato como por ejemplo se debe ingresar un dato entero para una variable entera y resulta que el usuario ingresa un número real.

c) NullPointerException, ocurre cuando se declara una variable y no se crea el objeto y por lo tanto no contiene nada la variable.

d) IOException, para capturar errores provenientes de la entrada y salida de datos en la ejecución del programa.

e) SQLException, proporciona información acerca de los errores ocasionados a partir de la conexión y acceso a la base de datos.

f) ArrayIndexOutOfBoundsException, se dispara esta excepción cuando se hace uso de índice incorrecto de un arreglo.

13.7. Programas resueltos en Java usando NetBeans

1. Crea una aplicación que permita el ingreso de dos números enteros y se proceda a mostrar el resulto de las cuatro operaciones de la aritmética. Hacer uso de excepciones en el cálculo de la división y en el ingreso de los datos.

```
1   package Capitulo_II.Sesion_13;
2
3   import Capitulo_I.Sesion_06.LecturaDatos;
4   import java.io.*;
5
6   public class Excepcion_Aritmetica {
7
8       public static void imprimir(String m) {
9           System.out.print(m);
10      }
```

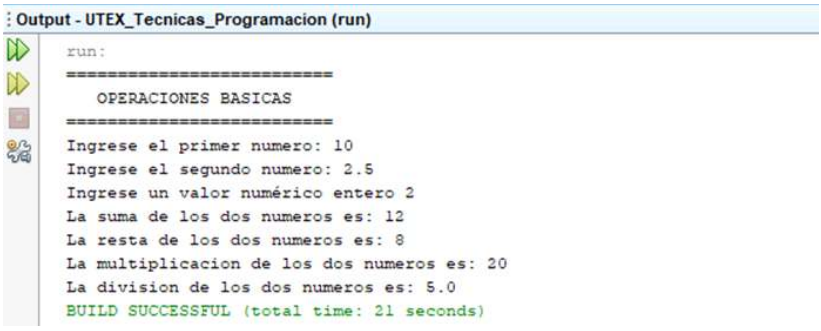
```
11
12      public static void main(String args[]) throws IOException {
13          int a = 0, b = 0;
14          imprimir("=========================\n");
15          imprimir("   OPERACIONES BASICAS\n");
16          imprimir("=========================\n");
17          try {
18              imprimir("Ingrese el primer numero: ");
19              a = LecturaDatos.leerDatoEntero();
20          } catch (NumberFormatException e) {
21              try {
22                  imprimir("Ingrese un valor numérico entero ");
23                  a = LecturaDatos.leerDatoEntero();
24              } catch (NumberFormatException ex) {
25                  imprimir("Vuelve a ejecutar el programa\n");
26                  return;
27              }
28          }
29          try {
30              imprimir("Ingrese el segundo numero: ");
31              b = LecturaDatos.leerDatoEntero();
32          } catch (NumberFormatException e) {
33              try {
34                  imprimir("Ingrese un valor numérico entero ");
35                  b = LecturaDatos.leerDatoEntero();
36              } catch (NumberFormatException ex) {
37                  imprimir("Vuelve a ejecutar el programa\n");
38                  return;
39              }
40          }
41
42          try {
43              imprimir("La suma de los dos numeros es: " + (a + b) + "\n");
44              imprimir("La resta de los dos numeros es: " + (a - b) + "\n");
45              imprimir("La multiplicacion de los dos numeros es: " + (a * b) + "\n");
46              imprimir("La division de los dos numeros es: " + ((double)a / b) + "\n");
47          } catch (ArithmeticException e) {
48              imprimir("No se puede dividir entre CERO\n");
49          }
50      }
51  }
```

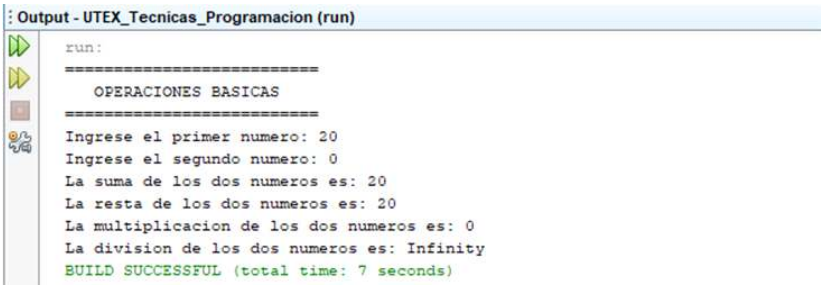
Interpretación de la programación

La clase Excepcion_Aritmetica pertenece al paquete Capitulo_II.Sesion_13 y para esta aplicación se está importando la clase LecturaDatos que se encuentra en el paquete CapituloI.Sesion_06. En el método main se hace uso del bloque try{ …} que permite proteger las instrucciones ante el posible error que pueda cometer el usuario de la aplicación. Las instrucciones en el bloque try una vez que se inicia la ejecución del programa y se presente algún error dispara una excepción que deberá ser recepcionada por un catch que ejecuta lo programado. En este programa se ha establecido el uso de dos excepciones para cuando el usuario cometa el error de ingresar un dato no adecuado como por

ejemplo un número con decimales cuando debe ingresar un entero. También está programado cuando el usuario ingrese como segundo número un cero y éste sea usado en la división. No se pude dividir un numero cualquiera entre cero.

Ejecución del programa:





2. Crea una aplicación que permita el ingreso de datos de Artefactos como el código, la descripción, el tipo, el stock y el precio unitario usando un arreglo de objetos. Para el ingreso de datos validar los datos numéricos con excepciones y luego listar los datos ingresados.

```
1   package Capitulo_II.Sesion_13;
2
3   import Capitulo_I.Sesion_06.LecturaDatos;
4   import java.io.*;
5
6   class Artefacto
7   {
8       private int codartefacto;
9       private String descripcion;
10      private String tipoartefacto;
11      private int stock;
12      private double preciounitario;
13
```

```
14      public Artefacto(int codartefacto, String descripcion, String tipoartefacto,
15              int stock, double preciounitario) {
16          this.codartefacto = codartefacto;
17          this.descripcion = descripcion;
18          this.tipoartefacto = tipoartefacto;
19          this.stock = stock;
20          this.preciounitario = preciounitario;
21      }
22
23      public int getCodartefacto() {
24          return codartefacto;
25      }
26
27      public void setCodartefacto(int codartefacto) {
28          this.codartefacto = codartefacto;
29      }
30
31      public String getDescripcion() {
32          return descripcion;
33
34      }
35
36      public void setDescripcion(String descripcion) {
37          this.descripcion = descripcion;
38      }
39
40      public String getTipoartefacto() {
41          return tipoartefacto;
42      }
43
44      public void setTipoartefacto(String tipoartefacto) {
45          this.tipoartefacto = tipoartefacto;
46      }
47
48      public int getStock() {
49          return stock;
50      }
51
52      public void setStock(int stock) {
53          this.stock = stock;
54      }
55
56      public double getPreciounitario() {
57          return preciounitario;
58      }
59
60      public void setPreciounitario(double preciounitario) {
61          this.preciounitario = preciounitario;
62      }
63
64  }
65
66  public class Excepcion_Arreglo {
67
68      public static void imprimir(String m)
69      {
70          System.out.print(m);
71      }
72
74      public static void listarArtefactos(Artefacto a[])
75      {
76          imprimir("=========================================\n");
77          imprimir("         LISTADO DE ARTEFACTOS\n");
78          imprimir("=========================================\n");
          int i;
```

```
79          imprimir("CODIGO\tDESCRIPCION\tTIPO\t\tSTOCK\t\tPRECIO UNITARIO\n");
80          for(i=0;i<a.length;i++)
81          {
82              imprimir(a[i].getCodartefacto()+"\t"+a[i].getDescripcion()+"\t"+
83                      a[i].getTipoartefacto()+"\t\t"+a[i].getStock()+"\t\t"+
84                      a[i].getPreciounitario()+"\n");
85          }
86      }
87
88      public static void main(String args[])throws IOException
89      {
90          int n, cuenta, cod=0, s=0;
91          String des, tipo;
92          double precio=0.0;
93          boolean verifica=true;
94          imprimir("========================\n");
95          imprimir("  REGISTRO DE ARTEFACTOS\n");
96          imprimir("========================\n");
97          imprimir("Indica la cantidad de artefactos a ingresar: ");
98          n=LecturaDatos.leerDatoEntero();
99          Artefacto artefactos[]=new Artefacto[n];
100         cuenta=0;
101         while(cuenta<n)
102         {
103             imprimir("===== PRODUCTO Nro "+(cuenta+1)+" =====\n");
104             while(verifica)
105             {
106                 imprimir("Codigo: ");
107                 try
108                 {
109                     cod=LecturaDatos.leerDatoEntero();
110                     verifica=false;
111                 }
112                 catch(Exception e)
113                 {
114                     imprimir("Codigo NO VALIDO. Vuelva a ingresar\n");
115                     continue;
116                 }
117             }
118             verifica=true;
119             imprimir("Descripcion: ");
120             des=LecturaDatos.leerDatoTexto();
121             imprimir("Tipo: ");
122             tipo=LecturaDatos.leerDatoTexto();
123
124             while(verifica)
125             {
126                 imprimir("Stock: ");
127                 try
128                 {
129                     s=LecturaDatos.leerDatoEntero();
130                     verifica=false;
131                 }
132                 catch(Exception e)
133                 {
134                     imprimir("Stock NO VALIDO. Vuelva a ingresar\n");
135                     continue;
136                 }
137             }
138             verifica=true;
139
140             while(verifica)
141             {
142                 imprimir("Precio unitario: ");
143                 try
144                 {
145                     precio=LecturaDatos.leerDatoReal();
146                     verifica=false;
147                 }
```

```
                catch(Exception e)
                {
                    imprimir("Precio NO VALIDO. Vuelva a ingresar\n");
                    continue;
                }
            }
            Artefacto A=new Artefacto(cod,des,tipo,s,precio);
            artefactos[cuenta]=A;
            imprimir("\n");
            verifica=true;
            cuenta++;
        }
        listarArtefactos(artefactos);
    }
}
```

Interpretación de la programación

La clase Artefacto contiene los atributos codartefacto, descripcion, tipoartefacto, stock y preciounitario. Además, tiene el método constructor y los métodos set y get para dar y obtener los datos a cada uno de los atributos. En la clase Excepcion_Arreglo, en el método main inicia con la solicitud de cuántos artefactos se van ingresar. A partir de este dato se crea el arreglo de objetos artefactos a partir de la clase Artefacto. Luego se procede al ingreso de datos de cada artefacto usando excepciones como por ejemplo en el caso del ingreso del código de artefacto se valida el ingreso de un, valor numérico entero para lo cual se hace uso de la excepción general denominada Exception programado en el catch. Existe una variable booleana verifica que permite manejar mejor el ingreso de datos. Al final se crea un objeto A, a partir de la clase Artefacto y con los datos ingresados del artefacto y posteriormente, se invoca la ejecución del método listarArtefactos que visualiza todos los datos de los artefactos.

Ejecución del programa:

```
Output - UTEX_Tecnicas_Programacion (run)
run:
========================
  REGISTRO DE ARTEFACTOS
========================
Indica la cantidad de artefactos a ingresar: 2
===== PRODUCTO Nro 1 =====
Codigo: 1
Descripcion: TV LED 55 PULGADAS LG
Tipo: Video
Stock: 20
Precio unitario: t
Precio NO VALIDO. Vuelva a ingresar
Precio unitario: 2340

===== PRODUCTO Nro 2 =====
Codigo: 2
Descripcion: MINICOMPONENTE SONY
Tipo: Audio
Stock: 10
Precio unitario: 890

=============================================
          LISTADO DE ARTEFACTOS
=============================================
CODIGO  DESCRIPCION      TIPO          STOCK          PRECIO UNITARIO
1       TV LED 55 PULGADAS LG   Video          20              2340.0
2       MINICOMPONENTE SONY     Audio          10              890.0
```

14. DESARROLLO DE UN PROYECTO

El desarrollo de un proyecto es necesario comprender los requerimientos de lo que se desea implementar. El caso a desarrollar es un pequeño sistema de Tienda.

14.1. PROYECTO: TIENDA

Este proyecto desarrollado una parte con programación en Java usando el entorno de desarrollo NetBeans consta de 5 archivos de extensión java:

Almacen.java

DetallePedido.java

Pedido.java

Producto.java

Sistema_Tienda.java

Hace uso clases, atributos y métodos como también de arreglos, relaciones de clases, arreglos de objetos entre otras cosas. Queda por implementar el registro de pedidos y el listado de pedidos, estando implementado la creación de almacenes, el registro de los datos de los productos, la modificación de datos del producto y el listado de productos.

Sistema_Tienda.java *contiene:*

```
1   package Capitulo_II.Sesion_14;
2
3   import Capitulo_I.Sesion_06.LecturaDatos;
4   import java.io.*;
5   import java.util.HashSet;
6   import java.util.Set;
7
8   public class Sistema_Tienda {
9
10      public static void imprimir(String m)
11      {
12          System.out.print(m);
13      }
14
15      //método para visualizar el menu principal y retorna la opcion elegida
16      public static int mostrarMenuPrincipal() throws IOException {
17          int opcion;
18          imprimir("==============\n");
19          imprimir("MENU PRINCIPAL\n");
20          imprimir("==============\n");
21          imprimir("\n");
22          imprimir("1. Adicionar Almacen\n");
23          imprimir("2. Mantenimiento de Productos\n");
24          imprimir("3. Realizar Pedidos\n");
25          imprimir("4. Listar Productos\n");
26          imprimir("5. Listar Pedidos\n");
27          imprimir("6. Salir\n");
28          imprimir("\n");
29          imprimir("Ingresa una opcion: ");
30          opcion = LecturaDatos.leerDatoEntero();
31          return opcion;
32      }
33
34      //método para visualizar el menú de productos t retorna la opcion elegida
35      public static int mostrarMenuProductos() throws IOException {
36          int opcion;
37          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
38          imprimir("MENU PARA PRODUCTOS\n");
39          imprimir("==================\n");
40          imprimir("\n");
41          imprimir("1. Adicionar Producto\n");
42          imprimir("2. Modificar Producto\n");
43          imprimir("3. Salir\n");
44          imprimir("\n");
45          imprimir("Ingresa una opcion: ");
46          opcion = LecturaDatos.leerDatoEntero();
47          return opcion;
48      }
49
50      public static void main(String args[]) throws IOException {
51          String nombrealmacen, nompro;
53          int rpta = 1, cant, opcion, r = 0, numpro, i;
54          float precio;
55          Almacen A = null;
56          Producto p;
57          do {
58              opcion = mostrarMenuPrincipal();
59              switch (opcion) {
60                  case 1:
                        imprimir("Ingrese el nombre del almacen: ");
```

```
61                  nombrealmacen = LecturaDatos.leerDatoTexto();
62                  A = new Almacen(nombrealmacen);
63                  if (A != null) {
64                      imprimir("El Almacen ha sido creado !!!\n");
65                  }
66                  break;
67              case 2:
68                  do {
69                      r = mostrarMenuProductos();
70                      switch (r) {
71                          case 1:
72                              imprimir("\n");
73                              imprimir("====================\n");
74                              imprimir("INGRESO DE PRODUCTOS\n");
75                              imprimir("====================\n");
76                              do {
77                                  // solicita los datos del producto
78                                  imprimir("CODIGO DEL PRODUCTO: ");
79                                  numpro = LecturaDatos.leerDatoEntero();
80                                  imprimir("NOMBRE: ");
81                                  nompro = LecturaDatos.leerDatoTexto();
82                                  System.out.print("STOCK: ");
83                                  cant = LecturaDatos.leerDatoEntero();
84                                  System.out.print("PRECIO: ");
85                                  precio = LecturaDatos.leerDatoFloat();
86                                  imprimir("¿Desea ingresar los datos de otro producto?"
87                                          + " (Si=1, No=0): ");
88                                  rpta = LecturaDatos.leerDatoEntero();
89                                  imprimir("\n");
90                                  //Creando el Objeto P a partir de la Clase Producto
91                                  p = new Producto(numpro, nompro, cant, precio);
92                                  //Adiciona el objeto al Arreglo de Productos que es
93                                  //un atributo de Almacen
94                                  A.adicionarProducto(p);
95                              } while (rpta == 1);
96                              break;
97                          case 2:
98                              imprimir("\n");
99                              imprimir("======================\n");
100                             imprimir("MODIFCACION DE PRODUCTOS\n");
101                             imprimir("======================\n");
102                             do {
103                                 // solicita el codigo del producto
104                                 imprimir("CODIGO DEL PRODUCTO: ");
105                                 numpro = LecturaDatos.leerDatoEntero();
106                                 //obtengo el producto buscando por codigo
107                                 p=A.buscarProductoXcod(numpro);
108                                 if(p!=null)
109                                 {
110                                     //si lo encuentra muestra los datos del producto
111                                     imprimir(p.toString()+"\n");
112                                     imprimir("\n");
113                                     //solicta el nuevo nombre del producto
114                                     imprimir("CAMBIAR EL NOMBRE: ");
115                                     nompro = LecturaDatos.leerDatoTexto();
116                                     //solicta el nuevo stock
117                                     imprimir("CAMBIAR EL STOCK: ");
118                                     cant = LecturaDatos.leerDatoEntero();
119                                     //solicta el nuevo precio
120                                     imprimir("CAMBIAR EL PRECIO: ");
121                                     precio = LecturaDatos.leerDatoFloat();
122                                     //se procede a cambiar los datos
123                                     p.setNombre(nompro);
124                                     p.setStock(cant);
125                                     p.setPrecio(precio);
126                                     //se procede a modificar los datos del producto
127                                     A.modificarProducto(p);
```

```
                                }
                                else
                                    imprimir("No existe código del producto");
                                imprimir("¿Desea cambiar los datos de otro producto?"
                                        + " (Si=1, No=0): ");
                                rpta = LecturaDatos.leerDatoEntero();
                                imprimir("\n");
                            } while (rpta == 1);
                            break;
                        }
                    } while (r != 3);
                    break;
                case 3:
                    // Aqui falta implementar la realizacion del pedido
                    break;
                case 4:
                    imprimir("==================\n");
                    imprimir("LISTA DE PRODUCTOS\n");
                    imprimir("==================\n");
                    imprimir("Nombre\tStock\tPrecio\n");
                    int total;
                    total = A.getNproductos();
                    if (total > 0) {
                        for (i = 0; i < total; i++) {
                            p = A.obtenerProducto(i);
                            imprimir(p.getNombre() + "\t" + p.getStock() + "\t"
                                    + p.getPrecio()+"\n");
                        }
                    }
                    break;
                case 5:
                    // Falta hacer el listado de pedidos
            }
        } while (opcion != 6);
        imprimir("");
        imprimir("Gracias por hacer uso del Sistema de Tienda");
    }
}
```

En el archivo Sistema_Tienda,java se inicia con la visualización de un menú principal, el mismo que una vez se seleccione una opción puede llevar a mostrar otro menú. Por ejemplo, si se elige la opción 1 que es adicionar un almacén sólo le solicitará el nombre del almacén, pero si selecciona la opción 2 que es Mantenimiento de Productos muestra otro menú con opciones para el ingreso de un nuevo producto o para modificar los datos de un producto ya ingresado. En este programa se puede apreciar comentarios de las instrucciones donde se puede evidenciar el uso de variables de objeto creado a partir de clases y el uso de arreglos de objetos.

Almacen.java *contiene:*

```
1   package Capitulo_II.Sesion_14;
2
3   public class Almacen
4   {
5       private String nombre;
6       private int nroproductos;
7       //atributo productos declarado a partir de la clase Producto
8       private Producto[] productos;
9
10      public Almacen(String nom)
11      {
12          nombre=nom;
13          nroproductos = 0;
14          //creación del arreglos de objetos productos
15          productos = new Producto[100];
16      }
17
18      //Método para adicionar o grabar un producto nuevo para un almacen especifico
19      public String adicionarProducto(Producto p)
20      {
21          //Variable que almacenará el mensaje de que si se registró o no el producto
22          String respuesta;
23          //Consulta si el arreglo tiene menos de 100 datos ingresados
24          if (nroproductos < 100) {
25              //Si tiene menos de 100 datos lo agrega al arreglo
26              productos[nroproductos] = p;
27              //Incrementa la cantidad de datos que tiene el arreglo
28              nroproductos++;
29              //Se define el mensaje guardando de que si se registró el producto
30              respuesta = "Se registro el producto";
31          } else {
32              //Se define un mensaje guardando que no se pudo registrar el producto
33              respuesta = "No se registro el producto";
34          }
35          return respuesta;
36      }
37
38      //Método para obtener los datos del producto dado el indice
39      public Producto obtenerProducto(int indice)
40      {
41          //El indice tiene que ser menor a la cantidad de datos que se tiene
42          //en el arreglo
43          if (indice < 0 || indice >= nroproductos) {
44              //Si el indice esta fuera del rango se retorna nulo.
45              return null;
46          } else {
47              //Retorma un objeto de tipo Producto según el elemento
48              //indicado por la variable indice
49              return productos[indice];
50          }
51      }
52
53      //Método que permite buscar Producto por su nombre
54      public Producto buscarProducto(String nom) {
55          Producto p=null;
56          //Hace un recorrido con los elementos del arreglo de productos
57          for(int i=0; i<nroproductos; i++) {
58              //Verifica si existe el Producto x su nombre
59              if(nom.equalsIgnoreCase(productos[i].getNombre())) {
60                  p=obtenerProducto(i);
61                  break; //sale de la sentencia repetitiva for
62              }
63          }
64          return p;
65      }
```

```
66
67      public Producto buscarProductoXcod(int cod) {
68          Producto p=null;
69          //Hace un recorrido con los elementos del arreglo de productos
70          for(int i=0; i<nroproductos; i++) {
71              //Verifica si existe el Producto x su nombre
72              if(cod==productos[i].getCodPro()) {
73                  p=obtenerProducto(i);
74                  break; //sale de la sentencia repetitiva for
75              }
76          }
77          return p;
78      }
79
80
81      //Método que permite la modificación de los datos de un producto
82      public String modificarProducto(Producto p) {
83          //Variable que nos indicara si realmente se registro el producto
84          String respuesta;
85          int indice=-1;
86          for(int i=0; i<this.nroproductos; i++) {
87              //Verifica si existe el Producto por su nombre
88              if(p.getNombre().equals(this.productos[i].getNombre())) {
89                  //ObjTabla.setValue(valor,nroFila,nroColumna)
90                  indice=i; //en indice se guarda la posición del producto buscado
91                  break;//sale de la sentencia repetitiva for
92              }
93          }
94          //Consulta si el arreglo tiene menos de 100 items
95          if (indice < 100) {
96              //Si tiene menos de 100 items lo agrega al arreglo
97              this.productos[indice] = p;
98              //Almacena un mensaje en la variable respuesta indicando que si se
99              //modificó los datos del producto
100             respuesta = "Se modificó los datos del producto";
101         } else {
102             //Almacea un mensaje en a variable respuesta indicando que si se
103             //modifico los datos del producto
104             respuesta = "No se modificó los datos del producto";
105         }
106         return respuesta;
107     }
108
109
110     public int getNproductos() {
111         return nroproductos;
112     }
113     public String getNombre() {
114         return nombre;
115     }
116 }
```

La clase Almacén tiene como atributos nombre, nroproductos y productos, este último es definido a partir de la clase Producto y creado con 100 elementos en el método constructor. Además, la clase Almacén se tiene el método adicionarProducto, el método buscarProducto, buscarProductoXcod y el método modificarProducto, estos métodos sirven para ingresar y modificar los datos de productos.

Producto.java *contiene:*

```
1   package Capitulo_II.Sesion_14;
2
3   public class Producto
4   {
5       //Atributos de la clase
6       private int codpro;
7       private String nombre;
8       private float precio;
9       private int stock;
10
11      //Método constructor
12      public Producto(int codpro, String nombre, int stock,float precio ) {
13          this.codpro = codpro;
14          this.nombre = nombre;
15          this.precio = precio;
16          this.stock = stock;
17      }
18
19      //Los Metodos Set y Get para cada atributo
20      public int getCodPro()
21      {
22          return codpro;
23      }
24
25      public String getNombre() {
26          return nombre;
27      }
28
29      public void setNombre(String nombre) {
30          this.nombre = nombre;
31      }
32
33      public int getStock() {
34          return stock;
35      }
36
37      public void setStock(int stock) {
38          this.stock = stock;
39      }
40
41      public float getPrecio() {
42          return precio;
43      }
44
45      public void setPrecio(float precio) {
46          this.precio = precio;
47      }
48
49      //Método para calcular el inventario de un producto
50      public float inventario()
51      {
52          return stock*precio;
53      }
54
55      //Método que retorna los datos del produto como dato de cadena de texto
56      public String toString() {
57          return "Producto{" +"CodPro="+codpro+ " Nombre=" + nombre + ", Precio=" +
58                  precio + ", Stock=" + stock + '}';
59      }
60  }
```

En la clase Producto contiene los atributos codpro, nombre, precio y stock. Se determina como métodos al método constructor, métodos de set y get de cada uno de los atributos y el método de cálculo inventario. Además, se establece la implementación del método toString.

Pedido.java *contiene:*

```
package Capitulo_II.Sesion_14;

public class Pedido {
    //Atributos de la clase Pedido
    private int nropedido;
    private String fecha;
    private String cliente;

    //Atributos resultado de la relacion con la clase DetallePedido
    private int nrodetalles;
    //Se declara el arreglo objeto detallespedido
    private DetallePedido[] detallespedido;

    //Método constructor
    public Pedido(int num, String fec,String cli) {
        nropedido = num;
        fecha = fec;
        cliente = cli;
        //Se inicializa en cero ya que será el primer elemento del arreglo a usar
        nrodetalles = 0;
        //Crea el arreglo de objetos de tamaño 100
        detallespedido = new DetallePedido[100];
    }

    //Se obtiene el numero del pedido
    public int getNroPedido() {
        return nropedido;
    }

    //Se obtiene la fecha de emisión del pedido
    public String getFecha() {
        return fecha;
    }

    //Se obtiene el nombre de cliente que realizó el pedido
    public String getCliente() {
        return cliente;
    }

    //Se obtiene la cantidad de detalles que tiene el Pedido
    public int getNrodetalles() {
        return nrodetalles;
    }

    //Sirve para adicionar un detalle de pedido al pedido
    public String adicionarDetalle(DetallePedido det) {
        //Variable que nos indica si realmente se registro el detalle de pedido
        String respuesta;
        //Consulta si el arreglo tiene menos de 50 items
```

```
50          if (this.nrodetalles < 50) {
51              //Si tiene menos de 50 items lo agrega al arreglo
52              this.detallespedido[this.nrodetalles] = det;
53              //Incremente la cantidad de elementos que tiene el arreglo
54              this.nrodetalles++;
55              //Manda un mensaje diciendo que se registro el detalle
56              respuesta = "Se registro el detalle de pedido";
57          } else {
58              //Manda un mensaje diciendo que no se pudo registrar
59              respuesta = "No se registro el detalle de pedido";
60          }
61          return respuesta;
62      }
63
64      //Método para calcular el total de pedido
65      public float calcularTotalPedido() {
66          float totalpedido = 0;
67          //Tenemos que recorrer todos los detallesPedido hasta el valor de la
68          //cantidad que nos indique la variable nrodetalles
69          for (int i = 0; i < nrodetalles; i++) {
70              //obtiene la cantidad
71              int cantidad = detallespedido[i].getCantidad();
72              //obtiene el precio
73              float precio = detallespedido[i].getPrecio();
74              //acumula en la variable totalpedido
75              totalpedido = totalpedido + cantidad * precio;
76          }
77          return totalpedido;
78      }
79
80      // Sirve para obtener un detalle de pedido segun el indice
81      public DetallePedido obtenerDetalle(int indice) {
82          //El indice tiene que ser menor a la cantidad de
83          //elementos que tenemos en el arreglo
84          if (indice < 0 || indice >= nrodetalles) {
85              //Evalua si el indice esta fuera del rango
86              return null;
87          } else {
88              //Retorma el elemento en la posicion que deseamos
89              return detallespedido[indice];
90          }
91      }
92
93  }
```

DetallePedido.java *contiene:*

```
1   package Capitulo_II.Sesion_14;
2
3   public class DetallePedido
4   {
5       //Atributos de la clase
6       private int cantidad;
7       private float precio;
8       //Atributo que se obtiene de la relacion con la clase de Producto
9       private Producto producto;
10
11      //Método contructor
12      public DetallePedido(int cantidad, float precio, Producto producto) {
13          this.cantidad = cantidad;
14          this.precio = precio;
15          this.producto = producto;
16      }
```

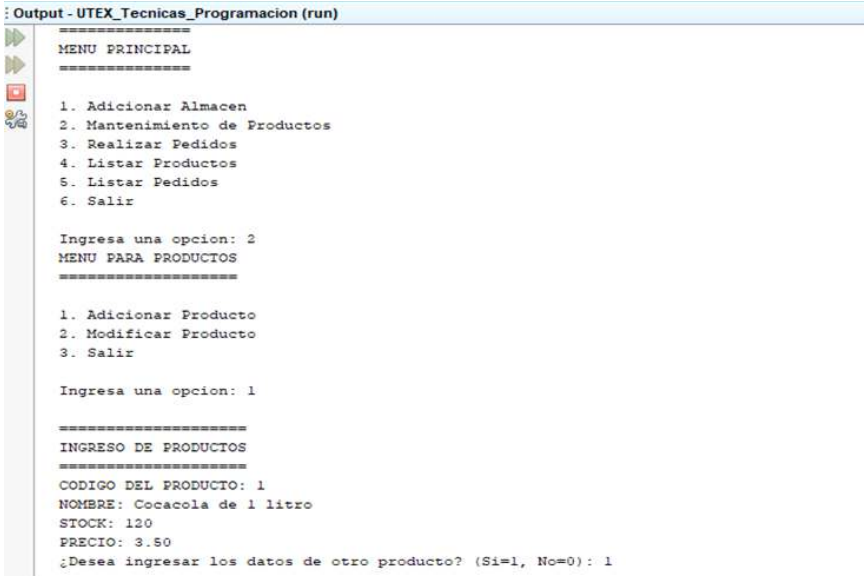
```
17
18      //Se obtiene la cantidad de Productos del detalle de pedido
19      public int getCantidad() {
20          return cantidad;
21      }
22
23      //Se obtiene el precio del Producto
24      public float getPrecio() {
25          return precio;
26      }
27
28      //Se obtiene el producto que esta en el detalle de pedido
29      public Producto getProducto() {
30          return producto;
31      }
32  }
```

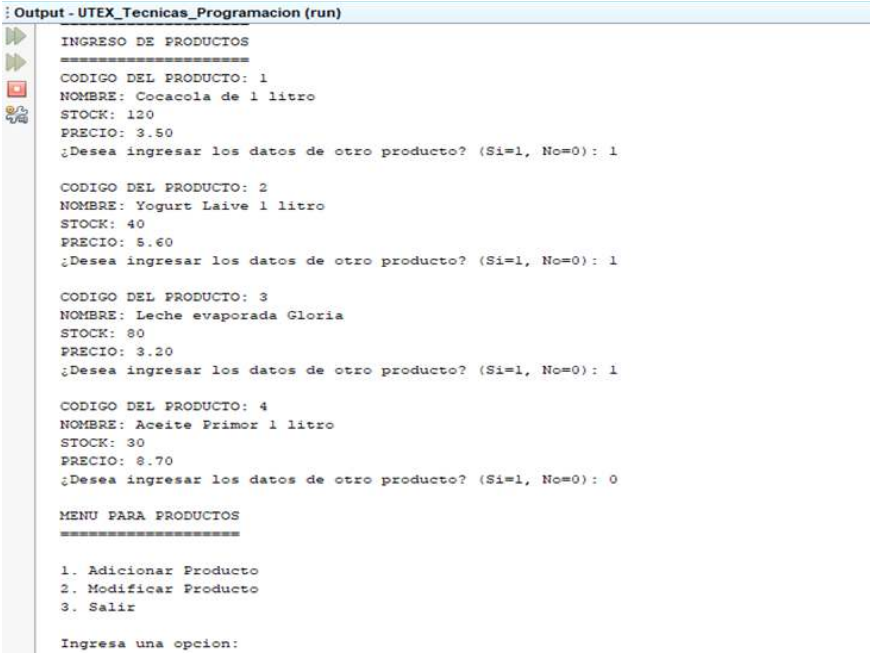
Ejecución de lo programado:

Ingresando el nombre de un almacén luego de seleccionar la opción1 del menú principal.

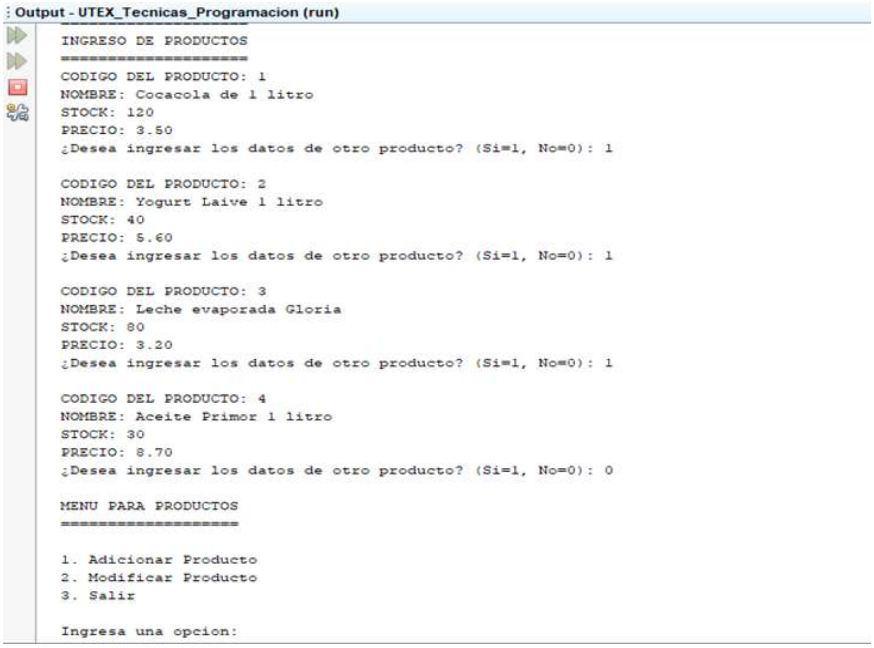
```
Output - UTEX_Tecnicas_Programacion (run)
run:
==============
MENU PRINCIPAL
==============

1. Adicionar Almacen
2. Mantenimiento de Productos
3. Realizar Pedidos
4. Listar Productos
5. Listar Pedidos
6. Salir

Ingresa una opcion: 1
Ingrese el nombre del almacen: Gamarra
El Almacen ha sido creado !!!
==============
MENU PRINCIPAL
==============

1. Adicionar Almacen
2. Mantenimiento de Productos
3. Realizar Pedidos
4. Listar Productos
5. Listar Pedidos
6. Salir

Ingresa una opcion:
```

Selecciona la opción 2 para mantenimiento de productos y luego adiciona productos.





Selecciona la opción 3 y luego lista los productos ingresados.



AUTOEVALUACIÓN

a) Responde a las siguientes preguntas:

1. Usando herencia de clases se logra heredar los atributos y métodos de una superclase. (verdadero o falso): _____

2. Los miembros protected de una superclase sólo están accesibles para los métodos de la superclase, los métodos de las subclases y los métodos de otras clases del mismo paquete (verdadero falso): _____

3. Se le denomina clase abstracta pura cuando todos sus métodos son abstractos, este concepto se refiere a: _____

4. Se puede escribir programas que procesen genéricamente objetos de todas las clases existentes en una jerarquía, se refiere a _____

5. Agrupación de clases afines se les denomina paquetes (verdadero o falso): _____

6. Es un evento que ocurre durante la ejecución de un programa que interrumpe el flujo normal de las sentencias: _____

7. Dos clases con el mismo nombre en dos paquetes distintos no pueden coexistir e incluso no pueden usadas en el mismo programa (verdadero o falso): _____

8. Controla un único aspecto dentro de un programa, como puede ser supervisar la entrada en un determinado periférico o controlar toda la entrada / salida del disco se denomina: _____

9. Muestra una lista de excepciones que te propone Java y la manera como se debe interpretarlos al momento que se muestre en la ejecución de un programa.

b) Desarrolla las siguientes aplicaciones:

1. Crea una clase denominada FigurasGeometricas y calcula el área y el perímetro del cuadrado, rectángulo y del triángulo. Hacerlo con métodos constructores sobrecargados y el operador this.

2. Crea una clase denominada Curso con los atributos codigocur y nombrecur. Luego aplica herencia y crea la clase Curso_Extension a partir de la clase Curso con los atributos horas_clase y costocur. También crea la clase Curso_Carrera con los atributos nombrecarrera y ciclo. Luego crea objetos a partir de las subclases

3. Una empresa vende equipos celulares, que puede ser utilizados únicamente para CLARO o para MOVISTAR. Se desea tener información de la marca, el modelo y el costo del equipo. Además, se puede adquirir adicionalmente el chip teniendo un descuento del 10% del costo del chip si es comprado conjuntamente con el celular. Por la compra de cualquier equipo, el cliente puede adquirir un seguro de pérdida o robo pagando el 15% del costo del equipo. El pago por el equipo celular puede ser en efectivo o con tarjeta. Si el pago es con tarjeta se incrementa el costo del equipo en un 2%. Utiliza una programación usando clases abstracta y herencia para calcular el precio del equipo y el monto del pago del seguro en caso decidiera acogerse. Usar clase abstracta.

4, Aplicando polimorfismo, implementar los atributos, métodos (con las validaciones correspondientes) y constructores de las siguientes clases con sus atributos indicados entre paréntesis: Figura (símbolo), Rectángulo (filas: entero >1, columnas: entero>1), Triángulo (filas: entero>1) y Rombo (filas: entero>2 e impar). Crear el método dibujar para cada caso. Crear la clase TestFiguras, que pruebe los métodos y constructores implementados. Como ejemplo de la ejecución del método dibujar para cada objeto creado, en caso que se haya asignado

como símbolo de la figura al asterisco (*) y los valores para las filas y columnas indicados a continuación, se tendría:

Objeto Rectángulo: Filas=2 Columnas=6

```
******
******
```

Objeto Triángulo: Filas=3

```
  *
 ***
*****
```

Objeto Rombo: Filas=7

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

REFERENCIAS BIBLIOGRÁFICAS

1. Blasco F. Programación orientada a objetos en Java [En Línea]. Ediciones de la U, 2019 [consultado 16 Nov 2020]. Disponible en: https://elibro.net/es/ereader/uladech/127125?page=172

2. Flórez Fernández H.A. Programación orientada a objetos usando java [En Línea]. Bogotá: Ecoe Ediciones, 2012 [consultado 24 Nov 2020]. Disponible en: https://elibro.net/es/ereader/uladech/69236?page=43

3. Moreno Pérez J. Programación orientada a objetos [En Línea]. RA-MA Editorial, 2015 [consultado 17 Nov 2020]. Disponible en: https://elibro.net/es/ereader/uladech/106461?page=91

4. García Llinás L.F. Todo lo básico que debería saber: sobre programación orientada a objetos en Java [En Línea]. Bogotá: Ediciones de la U, 2010 [consultado 17 Nov 2020]. Disponible en: https://elibro.net/es/ereader/uladech/69813?page=108

TÉCNICAS DE PROGRAMACIÓN
es una publicación del
Fondo Editorial de la Universidad Católica
Los Ángeles de Chimbote, Perú

FONDO EDITORIAL DE LA UNIVERSIDAD CATÓLICA
LOS ÁNGELES DE CHIMBOTE

ISBN: 978-612-4308-32-1