Ejercitación Práctica

1. Considere la siguiente clase. Determine la salida que se produce luego de ejecutar el main.

public class Raro {

public static void *main* (String[] args){

Raro raro= new Raro();

raro.primero();

raro.tercero();

raro.segundo();

raro.tercero();

}

public void primero (){

System. *out* .println( "Dentro del método primero." );

}

public void segundo (){

System. *out* .println( "Dentro del método segundo." );

primero();

}

public void tercero () {

System. *out* .println( "Dentro del método tercero." );

primero();

segundo();

}

}

SALIDA

* Dentro del método primero.
* Dentro del método tercero.
  + Dentro del método primero.
  + Dentro del método segundo.
  + Dentro del método primero.
* Dentro del método segundo.
* Dentro del método primero.
* Dentro del método tercero.

Dentro del método primero.

* + Dentro del método segundo.
  + Dentro del método primero.

2. Considere el siguiente programa:

public class Ejemplo{

public static void mostrarReglas (){

System . out . println ( "La primera regla " );

System . out . println ( "del club de Java es" );

System . out . println ( "" );

System . out . println ( "no se habla del club de Java!" );

}

public static void main ( String [] args ){

System . out . println ( "Las reglas del club de Java." );

Ejemplo . mostrarReglas ();

Ejemplo . mostrarReglas();

}

}

¿Qué sucedería si se realizan los siguientes cambios a la clase Ejemplo?

Considerar cada cambio de forma independiente de los otros.

Para cada cambio, considerar sólo tres posibilidades:

● “Nada”: Si no ocurrirá ningún cambio en el programa.

● “Error”. Si produjera que el programa no compile o que de un error durante su ejecución.

● “Salida”. Si cambiase la salida de la ejecución.

|  |  |
| --- | --- |
| Cambiar la línea 1 por public class Demostracion{ | ERROR |
| Cambiar la línea 8 por public static void MAIN ( String [] args ){ | NADA |
| Insertar una nueva línea debajo de la línea 10 que diga Ejemplo.mostrarReglas (); | SALIDA |
| Cambiar la línea 2 a public static void imprimirMensaje (){ | ERROR |
| Cambiar la línea 2 a public static void mostrarMensaje (){ y cambiar las líneas 10 y 11 a Ejemplo.mostrarMensaje (); | NADA |
| Reemplazar las líneas 3-4 con System.out.println ( "La primera regla del club de Java es, " ); | SALIDA |

3. Dadas las siguientes clases, analice el comportamiento en común. A partir del análisis implemente una interface que lo abstraiga.

public class Gato implements Animal {

private String objeto;

public Gato (String objeto){

this.objeto=objeto;

}

@Override

public void comer (){

System. *out* .println( "El gato come "+objeto);

}

@Override

public void jugar (){

System. *out* .println( "El gato juega con" + objeto );

}

}

public class Pez implements Animal {

private String objeto;

public Pez (String objeto){

this.objeto=objeto;

}

@Override

public void comer (){

System. *out* .println( "El pez come "+objeto );

}

@Override

public void jugar (){

System. *out* .println( "El PEZ juega con "+objeto );

}

}

Public interface animal{

Public abstract void comer();

Public abstract void jugar();

}

Public class Principal {

import Clases.Gato;

import Clases.Pez;

public static void main(String[] args) {

JOptionPane.showMessageDialog(null,"AHORA UTILIZAREMO POLIMORFISMO CON INTERFACE!!");

Gato gato=new Gato("Pelota");

Gato.comer();

Gato.jugar();

Pez pez=new pez(“Pepino de mar”);

Pez.comer();

Pez.jugar();

}

4. Considere el siguiente método

public String ifElseMisterioso ( int x , int y ) {

int z = 4;

if ( z <= x ) {

z = x + 1;

} else {

z = z + 9;

}

if ( z <= y ) {

y ++;

}

return z + " " + y ;

}

Para cada una de las siguientes invocaciones, determinar el retorno de la invocación:

ifElseMisterioso ( 3 , 20 ); 4+21

ifElseMisterioso ( 4 , 5 ); 5+6

ifElseMisterioso ( 5 , 5 ); 14+5

ifElseMisterioso ( 6 , 10 ); 14+10

5. Considere el siguiente método:

public void misterio ( int [] a , int [] b ) {

for ( int i = 0 ; i < a . length ; i ++) {

a [ i ] += b [ b . length - 1 - i ];

}

}

Dados los siguientes arreglos:

int [] a1 = { 1 , 3 , 5 , 7 , 9 };

int [] a2 = { 1 , 4 , 9 , 16 , 25 };

Determine los valores de los elementos en el arreglo a1 luego de ejecutar la siguiente invocación al método: misterio(a1,a2)

6. Determinar los valores almacenados en el arreglo array luego de que se ejecute el siguiente fragmento de código

int [] array = { 2 , 18 , 6 ,- 4 , 5 , 1};

for ( int i = 0 ; i < array . length ; i ++)

array [ i ] = array [ i ] + ( array [ i ] / array [ 0 ]);

array={3,27,9,-4,7.5,3/2}

**7)** Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular y cantidad (puede tener decimales).

El titular será obligatorio y la cantidad es opcional (controlar cantidad). Crea dos constructores que cumpla lo anterior.

Crea sus métodos get, set y toString.

Tendrá dos métodos especiales:

* ingresar (double cantidad): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
* retirar (double cantidad): se retira una cantidad a la cuenta, si restando la cantidad actual a la que nos pasan es negativa, la cantidad de la cuenta pasa a ser 0.