Programación orientada a objetos

1. **¿Cuál es la salida del siguiente programa?**

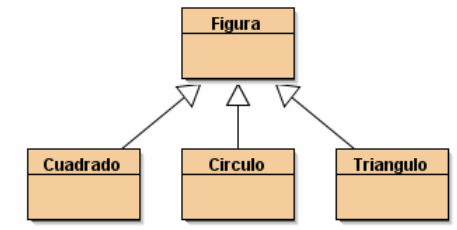
```
class C {
      void p (C c) { System.out.println("CC "); }
      void p (D d) { System.out.println("CD "); }
}

class D extends C {
      void p (C c) { System.out.println("DD"); };
}

public class InvocacionMetodos {

      public static void main(String[] args){
            C obj = new D();
            obj.p(obj);
            ((D)obj).p(obj);
            obj.p((D) obj);
      }
}
```

2. **Escribe un programa que implemente la siguiente jerarquía de clases:**



Implementar aquellos atributos y métodos necesarios para que se pueda ejecutar el siguiente programa:

```
public class Jerarquía {
      public static void main(String[] args) {
          Vector<Figura> figuras = new Vector<Figura>();
          figuras.add(new Circulo(10)); // Radio=10
          figuras.add(new Cuadrado(10)); // Lado=10
          figuras.add(new Triangulo(10,5)); // Base=10, Altura=5;
          for (Figura f: figuras)
              System.out.println("Área: "+f.area());
        }
}
```

Al ejecutar el programa, deberá aparecer por pantalla el área de cada una de las figuras creadas.

3. **Define una clase abstracta *Cuenta* con los siguientes atributos:**

- numerocuenta : entero largo
- saldo : double
- cliente : atributo de la clase *Persona* (que tiene nombre y apellidos, y NIF).

y con los siguientes métodos:

- Constructor parametrizado que recibe un cliente y un número de cuenta
- Accedentes para los tres atributos
- ingresar(double): permitirá ingresar una cantidad en la cuenta.

- abstract retirar(double): permitirá sacar una cantidad de la cuenta (si hay saldo). No se implementa, depende del tipo de cuenta
- actualizarSaldo(): actualizará el saldo de la cuenta, pero cada cuenta lo hace de una forma diferente

Define las subclases de *Cuenta* que se describen a continuación:

- *CuentaCorriente*: Cuenta normal con un interés fijo del 1.5%. Incluir constructor parametrizado y método *toString()*.
- CuentaAhorro: Esta cuenta tiene como atributos el interés variable a lo largo del año y un saldo mínimo necesario. Al retirar dinero hay que tener en cuenta que no se sobrepase el saldo mínimo. Incluir constructor parametrizado, método *toString()* y método para cambiar el interés.

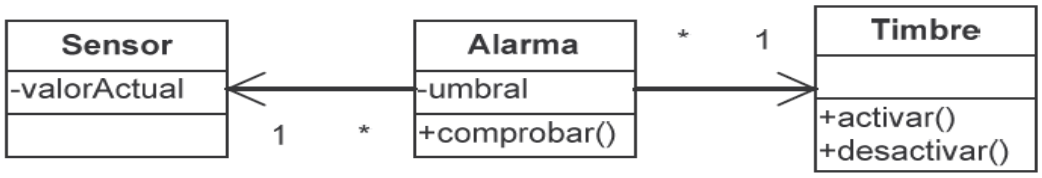
Crea un programa que cree varias cuentas y pruebe sus características.

4. Se trata de crear una pequeña base de datos de personas de una universidad. De momento definiremos y probaremos las siguientes clases:

- Direccion:
  - atributos: calle, ciudad, código postal, pais
  - Constructores predeterminado y parametrizado.
- Persona: Clase ya creada (con nombre, apellidos y NIF, ver ejercicio anterior) a la que añadiremos el atributo dirección y sus métodos accedentes y mutadores. Esta clase implementa la interface Humano, con un método indentificate(), que muestra el tipo de la clase que lo implementa (el tipo de persona, en este caso).
- Estudiante: Subclase de Persona.
  - Atributos: ID de estudiante
  - Constructores : predeterminado y constructor parametrizado que admita el ID.
  - Métodos accedentes y mutadores y toString().
- Profesor: Subclase de Persona.
  - Atributos : despacho
  - Constructores: predeterminado y constructor parametrizado que admita el despacho.
  - Métodos accedentes y mutadores y toString()

Crea una lista de personas (con la clase Vector) y prueba a añadir varios alumnos y varios profesores a la lista y sus operaciones.

5. Crea una clase denominada Alarma cuyos objetos activen un objeto de tipo Timbre cuando el valor medido por un Sensor supere un umbral preestablecido.



Implementa en Java todo el código necesario para el funcionamiento de la alarma, suponiendo que la alarma comprueba si debe activar o desactivar el timbre cuando se invoca el método comprobar().

Crea una subclase de Alarma denominada **AlarmaLuminosa** que, además de activar el timbre, encienda una luz (que se representará con un objeto de tipo Luz).

NOTA: Procurar eliminar la aparición de código duplicado al crear la subclase de Alarma y asegúrarse de que, cuando se activa la alarma luminosa se enciende la luz de alarma y también suena la señal sonora asociada al timbre.

6. Define una jerarquía de clases que permita almacenar datos sobre los planetas y satélites que forman parte del sistema solar (junto con el sol).

Algunos atributos que puede ser interesante recoger son: la masa del cuerpo, su diámetro medio, el período de rotación sobre el propio eje, período de traslación alrededor del cuerpo que orbitan, distancia

media a ese cuerpo, excentricidad de la órbita, etc.

Define un método que, dado un objeto del sistema solar (planeta o satelite), imprima toda la información de que se dispone sobre el mismo.

7. Define los siguientes elementos:

- interface **Puerta** : con los métodos abrir y cerrar.
- interface **PuertaBloqueable** : derivada de Puerta, con los métodos bloquea y desbloquea.
- interface **Alarma** :con los métodos alarmaActivada?, activarAlarma y desactivarAlarma.
- clase **ComponentedeCoche**: con los atributos descripción, peso y coste, y un método que muestre dichos atributos.
- clase **PuertaCoche**, con el atributo boolean estaBloqueada, y que extienda la clase ComponentedeCoche e implemente las interfaces Alarma y Puertabloqueable.

Escribe un programa que pruebe la clase PuertaCoche y todas las operaciones que admite.

8. Dado el siguiente código:

```
public class Electrodomestico {
       private double consumo;
       private String modelo;
       public Electrodomestico (double c, String m) {
         consumo = c;
         modelo = m;
        }
        public String toString () {
          return "Modelo: " + modelo + " Consumo: " + consumo ;
       }
}

 public interface MuestraInformacion {
      public String muestra();
 }
```

Se pide desarrollar la clase **Frigorifico**:

```
public class Frigorifico extends Electrodomestico
implements MuestraInformacion {
  double temperatura;
  public Frigorifico (double c, String m, double t) { ... }
  public String toString () { ... }
  public String muestra () { ... }
}
```

Utilizando cuando sea más conveniente los métodos de la clase Electrodomestico, se pide:

- Implementar el constructor de Frigorifico.
- Implementar el método toString de Frigorifico.
- Implementar el método muestra de Frigorifico para que muestre su temperatura.

9. Implementa las siguientes clases:

- clase **Nota**. Una nota contiene un identificador numérico y una línea de texto. Define constructor, accedentes, mutadores y toString.
- clase **NotaAlarma**. Una nota que además contiene la hora en la que sonará la alarma. Define constructor, accedentes, mutadores y toString.
- clase **BlocNotas** que modela un bloc de notas en el que se pueden introducir notas, listar todas las notas, eliminar una nota mediante su posición en el bloc de notas o saber cuantas notas contiene el bloc de notas. Debes utilizar una colección.
- clase **Prueba** que cree un bloc de Notas de ejemplo y pruebe las operaciones que soporta. .