Examen de Programación Orientada a Objetos (Plan 2014)

29 de octubre de 2018

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

Problema 1. (3 puntos)

Tenemos una clase `Punto2D` que encapsula la abstracción de un punto en el plano:

```
public class Punto2D {
      private int x, y;
      public Punto2D (int x, int y){
            this.x = x;
            this.y = y;
      }
      public boolean equals(Punto2D p){
            return (this.x == p.getX() && this.y == p.getY());
      }
      public int getX(){return this.x};
      public int getY(){return this.y};
}
```

Se pide definir una clase `Punto3D`, que herede de `Punto2D` y que encapsule la abstracción de un punto en el espacio. La clase deberá tener al menos el constructor y los métodos:

```
      public Punto3D (int x, int y, int z)
      public boolean equals(Punto3D p);
      public int getZ();
```

```
public class Punto3D extends Punto2D {
      private int z;
      public Punto3D (int x, int y, int z){
            super(x, y);
            this.z = z;
      }
      public boolean equals(Punto3D p){
            return super.equals(p) && this.z == p.getZ();
      }
      public int getZ(){return this.z};
}
```

Problema 2. (*3 puntos*).

Se está desarrollando una solución para la gestión de la facturación de empresas del sector energético. En este momento se está definiendo una clase `ReciboElectrico`, válida para todas las empresas energéticas, que encapsula la abstracción de un recibo de consumo de energía eléctrica. Para calcular el gasto se aplica la fórmula: gastoTotal = gastoDist + (gastoKw*numKw) dónde:

1) numKw es el número de Kilovatios a facturar en el recibo.
2) `gastoDist` es el gasto de distribución, que es único para todos los recibos, y que aprueba el Consejo de Ministros de manera periódica.
3) `gastoKw` es el gasto por kilovatio, que establece cada empresa periódicamente según las leyes del mercado.

Se pide completar la clase `ReciboElectrico`, implementando los métodos necesarios para fijar los valores de `gastoDist` y `gastoKw` y el método `getTotalFactura`, que recibe como parámetro `numKw` y devuelve el total del gasto a facturar. Este método deberá generar la excepción `ValorNegativoEnKilovatios`, que supondremos previamente definida. No se pide escribir el resto de los posibles miembros de la clase.

```
public class ReciboElectrico {
     private static double gastoDist;
     private double gastoKw;

     public static void setGastoDist (double gastoDist){
            this.gastoDist = gastoDist;
     }

     public void setGastoKw (double gastoKw){
            this.gastoKw = gastoKw;
     }

     public double getTotalFactura(double numKw)
                    throws ValorNegativoEnKilovatios {
        if(numKw<0d) throw new ValorNegativoEnKilovatios();
            return this.gastoDist + (this.gastoKw * numKw);
     };









 }
```

Problema 3. (*4 puntos*).

Teniendo en cuenta interfaz Shape:

```
public interface Shape {
    double getPerimeter();
    double getArea();
}
```

Se pide completar la cabecera/declaración de la clase `Polygon` que la implementa y que debe representar a cualquier polígono, cuyo constructor recibe como parámetros un `array de enteros` que representan a los lados del polígono y el número de lados del polígono. También se pide definir la clase `Square` que representa a un cuadrado. Esta última deberá tener dos constructores, uno que recibe como parámetro un `array de enteros` que representan a los lados del cuadrado y otro con un único parámetro de tipo `entero` que indica la longitud del lado. Se podrán definir tantas propiedades y métodos como se consideren necesarios.

```
public abstract class Polygon implements Shape {
  private int[] sides;
  private int numberOfSides;
  public Polygon(int numberOfSides, int[] sides) {
    this.sides = sides;
    this.numberOfSides = numberOfSides;
  }
  public int[] getSides() {
    return sides;
  }

  @Override
  public double getPerimeter(){
      int aux = 0;
      for(int side: sides){
        aux += side;
      }
      return aux;
  }
  @Override
  public abstract double getArea();


}

 public class Square extends Polygon {
    public Square(int side) {
      this(new int[]{side, side, side, side});
    }
    public Square(int[] sides) {
      super(4, sides);
    }
    @Override
    public double getArea() {
       return getSides()[0]*getSides()[1];
    }
 }
```