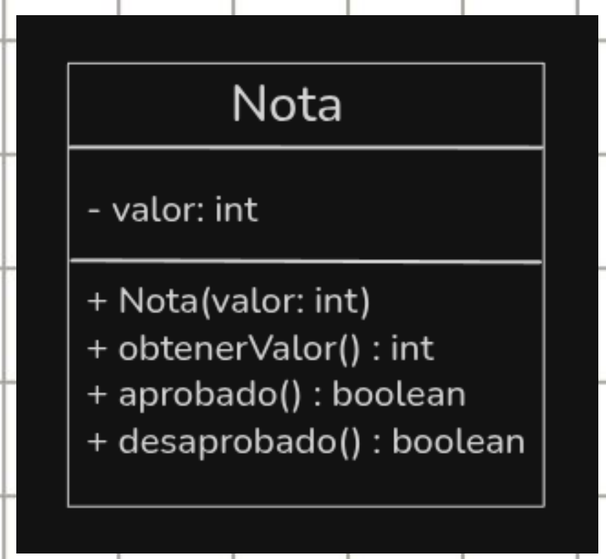
1. Hacer un diagrama UML e Implementar la clase Nota para cumplir con la siguiente interfaz:

```
public class Nota {
      /* pre: valor esta comprendido entre 1 y 10.
      * post: inicializa la Nota con el valor indicado .
      */
      Nota (int valor);

      /* post: devuelve el valor numérico de la Nota ,
      */
      int obtenerValor ();

      /* post: indica si la Nota permite o no la aprobación.
            Se aprueba con 4 */
      boolean aprobado();

      /* post: indica si la Nota implica la desaprobación. */
      boolean desaprobado ();
      }
```

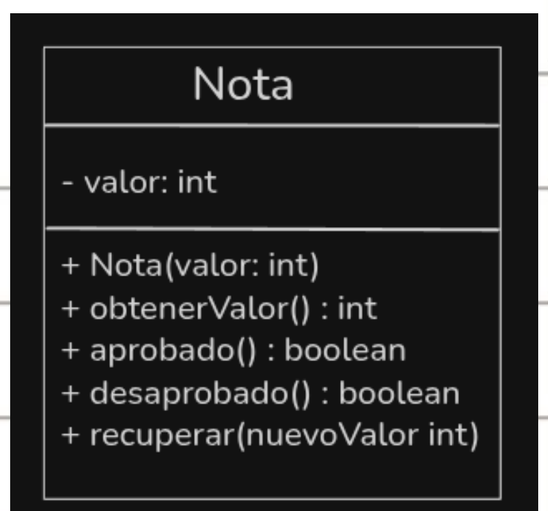
```java
public class Nota {  no usages

    private int valor;  4 usages

    Nota(int valor) {  no usages
        if (valor >= 0 && valor <= 10) {
            this.valor = valor;
        } else {
            throw new IllegalArgumentException("La nota debe estar entre 1 y 10.");
        }
    }

    public int obtenerValor() {  no usages
        return valor;
    }

    public boolean aprobado() {  no usages
        return valor >= 4;
    }

    public boolean desaprobado() {  no usages
        return valor < 4;
    }
```



2. Agregar a la clase Nota el siguiente método:

```
/* pre : nuevoValor está comprendido entre 1 y 10.
* post : modifica el valor numérico de la Nota, cambiándolo
* por nuevoValor , si nuevoValor es superior al
* valor numérico actual de la Nota .
*/

void recuperar (int nuevoValor );
```

```java
public void recuperar(int nuevoValor) {
    if(valor < nuevoValor) {
        valor = nuevoValor;
    }
}
```

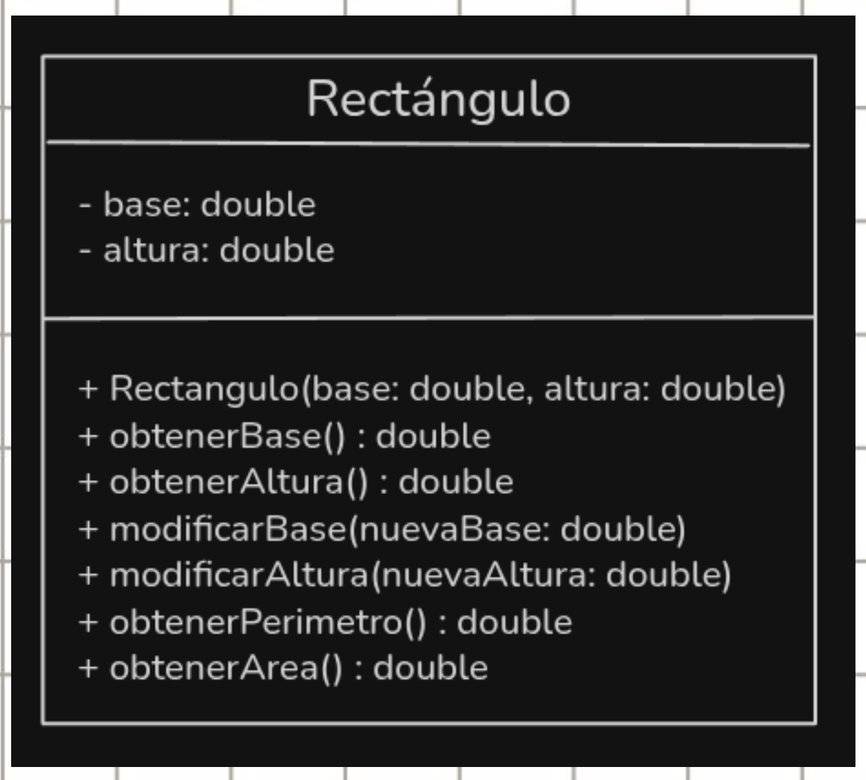


3. Implementar la clase Punto según el siguiente diagrama UML (agregar pre y pos):

Punto
- x: double - y: double
+ Punto() + Punto(double, double) + obtenerX(): double + obtenerY(): double + cambiarX(double) + cambiarY(double) + esOrigen(): boolean + estaEnEjeX(): boolean + estaEnEjeY(): boolean + distanciaAlOrigen(): double

```java
public class Punto {  no usages

    private double x;  6 usages
    private double y;  6 usages

    /*
    * post: inicializa los valores x e y
    */
    public Punto(double x, double y) {  no usages
        this.x = x;
        this.y = y;
    }

    /*
    * post : devuelve el valor de x
    */
    public double obtenerX() {  no usages
        return x;
    }

    /*
     * post : devuelve el valor de y
     */
    public double obtenerY() {  no usages
        return y;
    }

    /*
    * post : modifica el valor de x
     */
    public void cambiarX(double valor) {  no usages
        this.x = valor;
    }

    /*
     * post : modifica el valor de y
     */
    public void cambiarY(double valor) {  no usages
        this.y = valor;
    }

    /*
     * post : devuelve verdadero si las coordenadas son las del origen del plano
     */
    public boolean esOrigen() {  no usages
        return x == 0 && y == 0;
    }

    /*
     * post : devuelve verdadero si el punto del plano que definen x e y se encuentra sobre el eje y
     */
    public boolean estaEnEjeY() {  no usages
        return x == 0;
    }

    /*
     * post : devuelve verdadero si el punto del plano que definen x e y se encuentra sobre el eje x
     */
    public boolean estaEnEjeX() {  no usages
        return y == 0;
    }

    /*
     * post : devuelve la distancia al origen del punto del plano que definen x e y
     */
    public double distanciaAlOrigen() {  no usages
        return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
    }
}
```

4. Implementar la clase Rectangulo, con atributos base y altura y los métodos para modificar y obtener sus valores. Además, deben implementar métodos para obtener el perímetro y el área. Antes de la implementación, hacer el diagrama UML.




```java
public class Rectangulo {  no usages

    private double base;  5 usages
    private double altura;  5 usages

    public Rectangulo(double base, double altura) {  no
        this.base = base;
        this.altura = altura;
    }

    public double obtenerBase() {  no usages
        return base;
    }
    public double obtenerAltura() {  no usages
        return altura;
    }

    public void modificarBase(double nuevaBase) {  no us
        base = nuevaBase;
    }

    public void modificarAltura(double nuevaAltura) {
        altura = nuevaAltura;
    }

    public double obtenerPerimetro() {  no usages
        return (base + altura) * 2;
    }

    public double obtenerArea() {  no usages
        return base * altura;
    }
}
```

5. Implementar la clase TarjetaBaja a partir de la siguiente declaración (agregar las pre y pos necesarias):

```
public class TarjetaBaja {

      public TarjetaBaja ( double saldoInicial );

      public double obtenerSaldo ()

      public void cargar ( double monto );

      /* Las secciones son 1, 2 o 3, y los valores correspondientes son
      * $408.24, $454.78 y $489.82
      */
      public void pagarViajeEnColectivo (int seccion );

      /* El subte cuesta $832.
      */
      public void pagarViajeEnSubte ();

      public int viajesRealizadosSubte ();

      public int viajesRealizadosColectivo ();
}
```

```
public class TarjetaBaja {  no usages

    private double saldo;  7 usages
    private int viajesRealizadosSubte;  2 usages
    private int viajesRealizadosColectivo;  1 usage

    public TarjetaBaja (double saldoInicial){  no usages
        saldo = saldoInicial;
        viajesRealizadosSubte = 0;
    }

    public double obtenerSaldo() {  no usages
        return saldo;
    }

    public void cargar (double monto) {  no usages
        saldo += monto;
    }

    /* Las secciones son 1, 2 o 3, y los valores correspondientes son
     * $408.24, $454.78 y $489.82
     */
    public void pagarViajeEnColectivo (int seccion) {  no usages
        switch (seccion) {
            case 1:
                saldo -= 408.24;
                break;
            case 2:
                saldo -= 454.78;
                break;

            case 3:
                saldo -= 489.82;
                break;
            default:
                System.out.println("La sección tiene que ser un número entre 1 y 3");
        }
    }

    /* El subte cuesta $832.
     */
    public void pagarViajeEnSubte() {  no usages
        saldo -= 832;
    }

    public int viajesRealizadosSubte() {  no us
        return viajesRealizadosSubte;
    }

    public int viajesRealizadosColectivo() {
        return viajesRealizadosColectivo;
    }
```

6. Implementar la clase Ticket a partir de la siguiente interfaz:

```
public class Ticket {

      /* post : el Ticket se inicializa con importe 0.
      */
      public Ticket ();

      /* pre : cantidad y precio son mayores a cero . El ticket está
      * abierto .
      * post : suma al Ticket un item a partir de la cantidad de
      * de productos y su precio unitario .
      */
      public void agregarItem ( int cantidad , double precioUnitario );

      /* pre : el Ticket está abierto y no se ha aplicado un descuento
      * previamente .
      * post : aplica un descuento sobre el total del importe .
      */
      public void aplicarDescuento ( double porcentaje );

      /* post : devuelve el importe acumulado hasta el momento sin cerrar
      * el Ticket .
      */
      double calcularSubtotal ();

      /* post : cierra el Ticket y devuelve el importe total .
      */
      public double calcularTotal ();

      /* post : devuelve la cantidad total de productos .
      */
      int contarProductos ();
}
```

```java
import java.util.ArrayList;
import java.util.List;

public class Ticket {  no usages

    private double importe;  6 usages
    private boolean estaAbierto;  2 usages
    int cantidadTotalDeProductos;  3 usages
    List<String> items;  2 usages

    /* post : el Ticket se inicializa con importe 0.
     */
    public Ticket() {  no usages
        importe = 0;
        estaAbierto = true;
        cantidadTotalDeProductos = 0;
        items = new ArrayList<>();
    }

    /* pre : cantidad y precio son mayores a cero . El ticket está abierto .
     * post : suma al Ticket un item a partir de la cantidad de
     * de productos y su precio unitario .
     */
    public void agregarItem(int cantidad ,double precioUnitario) {  no usages
        items.add(String.valueOf( obj: "Cantidad: " +cantidad + "Precio: " + precioUnitario));
        cantidadTotalDeProductos+=cantidad;
        importe += precioUnitario*cantidad;
    }

    /* pre : el Ticket está abierto y no se ha aplicado un descuento previamente .
     * post : aplica un descuento sobre el total del importe .
     */
    public void aplicarDescuento(double porcentaje) {  no usages
            importe -= importe * porcentaje;
    }

    /* post : devuelve el importe acumulado hasta el momento sin cerrar el Ticket .
     */
    double calcularSubtotal() {  no usages
        return importe;
    }

    /* post : cierra el Ticket y devuelve el importe total .
     */
    public double calcularTotal() {  no usages
        estaAbierto = false;
        return importe;
    }

    /* post : devuelve la cantidad total de productos .
     */
    int contarProductos () {  no usages
        return cantidadTotalDeProductos;
    }
```