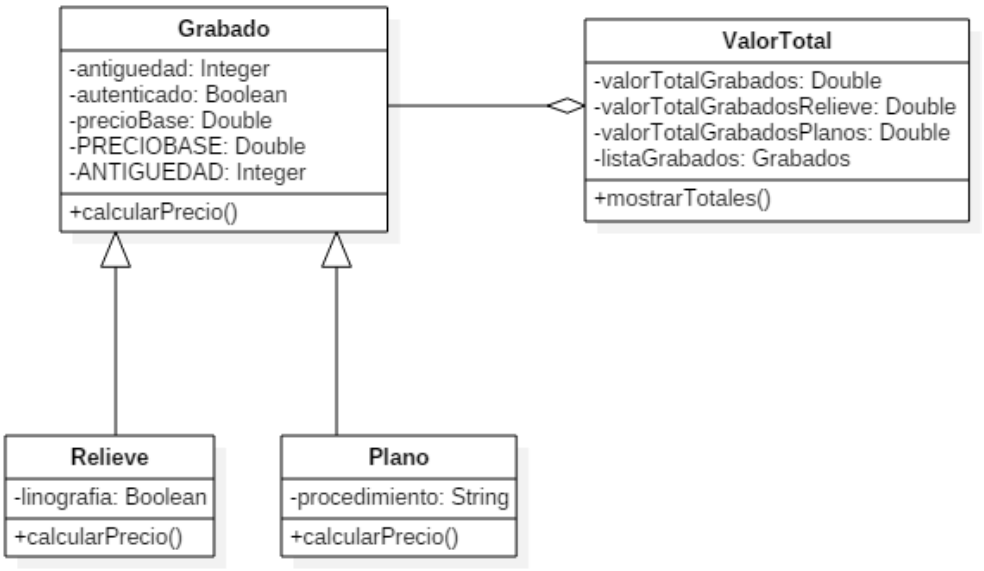
Ciclo 2 Fundamentos de programación

Reto 2

Descripción del problema:

El museo de arte ha decidido adquirir una colección de grabados desarrollados en técnicas de relieve, plano y otras genéricas. Con el fin de asegurarse una buena negociación necesita determinar el valor total de la colección, así como el valor total por tipo de grabados. Para esto se ha contratado su compañía, en donde se ha determinado que el modelo de clases mediante el cual se resolverá el problema es el siguiente:



Todos los grabados comparten los atributos `antiguedad` y `autenticado`, los cuales modifican su precio final. Se cuenta también con el atributo `precioBase` (en dólares), el cual representa el precio del grabado antes de sumar los respectivos valores según la evaluación de los atributos antigüedad y autenticado. El método `calcularPrecio()` permite obtener el precio final para un grabado de la siguiente forma:

Si el cuadro tiene entre 50 y 70 años (incluidos) de antigüedad al precio base del grabado aumenta un 10% de su precio base, si tiene mas de 70 años hasta 120 años (incluido) de antigüedad el precio base del grabado aumenta un 25% de su precio base, en caso de tener mas de 120 años el precio base del grabado aumenta un 40% de su precio base, en caso de tener menos de 50 años no aumenta su valor. Para el caso del atributo de autenticado si este es verdadero el cuadro aumenta su precio base en USD$800.

Los grabados del tipo relieve cuentan con el atributo adicional denominado `linografía` el cual en caso de ser verdadero agrega un valor adicional al precio del relieve de USD$700.

Los grabados del tipo plano cuentan con el atributo adicional denominado `procedimiento` el cual en caso de ser SERIGRAFIA agrega un valor adicional al precio del relieve de USD$700, en caso de ser MONOPATIA agrega un valor adicional al precio del relieve de USD$300 y si es LITOGRAFIA agrega un valor adicional al precio del relieve de USD$100.

En la colección existen grabados que no son de tipo `Relieve`, ni de tipo `Plano`, estos serán tratados como grabados de tipo `Grabado` que son genéricos y su precio será calculado por el método `calcularPrecio()` de la clase Grabado.

En caso de no ser suministrado se conoce que el mínimo precio base de cualquier grabado es de USD$350 y que la antigüedad mínima supuesta de las obras es de 50 años y que por ende no se encuentran autenticados. Estos valores deberán ser definidos como constantes de la clase `Grabado` y se deben utilizar como valor por defecto para calcular el precio final del grabado en caso de que no se envíen estos valores al constructor de la clase. Por lo anterior debes implementar 2 constructores en la clase principal, uno en el que contemples los casos en que se envían todos los parámetros y otro en el que no se envía ningún parámetro.

Los atributos de la clase `ValorTotal` son: `valorTotalGrabados, valorTotalGrabadosRelieve, valorTotalGrabadosPlanos y listaGrabados`. Este último atributo contiene todos los grabados de la colección, los cuales son almacenados en un array (tipo `Grabado`) y son entregados al constructor de la clase `ValorTotal` en el método `main()`, desde donde se llama al método `mostrarTotales()`, el cual debe imprimir en consola:

El precio total de los grabados que son genéricos.
El precio total de los grabados tipo relieves
El precio total de los grabados tipo planos.
El precio total de la colección.

Notas:

- Los precios presentados deben ser redondeados con Math.round().
- Cada una de las clases deben ser codificadas en una clase (archivo independiente) y las pruebas son ejecutadas desde en una clase Main.java .
- Las clases del reto, exceptuando Main.java, se deben cargar juntas en un solo archivo en la plataforma de pruebas iMaster.

Ejemplo:

Pruebas	Salida

`public class Main {`
`  public static void main(String[] args) {`
`  Grabado[] grabados = new Grabado[5];`
`  grabados[0] = new Grabado(60, true, 1000.0);`
`  grabados[1] = new Relieve(65, true, 1000.0, true);`
`  grabados[2] = new Plano(66, true, 80.0, "SERIGRAFIA");`
`  grabados[3] = new Plano(90, true, 180.0, "MONOPATIA");`
`  grabados[4] = new Grabado();`
`  ValorTotal respuesta = new ValorTotal(grabados);`
`  respuesta.mostrarTotales();`
 	
`  }`
`}`
	2285
	2600
	2913
	7798

`public class Main {`
`  public static void main(String[] args) {`
`  Grabado[] grabados = new Grabado[4];`
`  grabados[0] = new Grabado();`
`  grabados[1] = new Grabado(80, false, 3000.0);`
`  grabados[2] = new Relieve(90, false, 2000.0, false);`
`  grabados[3] = new Plano(35, true, 90.0, "LITOGRAFIA");`
	4135
	2500
	990
	7625

```
        ValorTotal respuesta = new ValorTotal(grabados);
        respuesta.mostrarTotales();

    }
}
```

Esqueleto:

```
// Esta clase Main no debe ser subida a imaster solo es para que realice las pruebas
public class Main {
    public static void main(String[] args) {
        Grabado[] grabados = new Grabado[5];
        grabados[0] = new Grabado(60, true, 1000.0);
        grabados[1] = new Relieve(65, true, 1000.0, true);
        grabados[2] = new Plano(66, true, 80.0, "SERIGRAFIA");
        grabados[3] = new Plano(90, true, 180.0, "MONOPATIA");
        grabados[4] = new Grabado();
        ValorTotal respuesta = new ValorTotal(grabados);
        respuesta.mostrarTotales();

    }

}

class Grabado {
    // Constantes
    private final static Double PRECIOBASE = 350.0;
    private final static Integer ANTIGUEDAD = 50;
    private final static Boolean AUTENTICADO = false;

    // Atributos
    private Integer antiguedad;
    private Boolean autenticado;
    private Double precioBase;

    // Constructores

    // Métodos
    public double calcularPrecio() {
        //
        return precioFinal;
```

```
    }

}

class Plano extends Grabado {
    // Atributos


    // Constructores

    // Metodos
    @Override
    public double calcularPrecio() {
        return precioFinal;
    }
}

class Relieve extends Grabado {
    // Atributos


    // Constructores

    // Metodos
    @Override
    public double calcularPrecio() {

        return precioFinal;
    }

}

class ValorTotal {
    // Atributos


    // Constructores

    // Metodos
    public void mostrarTotales() {
        // cálculos totales

        System.out.println(Math.round(valorTotalGrabados));
        System.out.println(Math.round(valorTotalGrabadosRelieve));
        System.out.println(Math.round(valorTotalGrabadosPlano));
```

```
        System.out.println(Math.round(totalColeccion));
    }


}
```