Simulacro de examen escrito Curso 2024-2025

Curso Académico (*Academic year*): 2023-2024
Fecha (*Date*):

Fecha/Date:

Nombre y Apellidos (*Name and Surname*): *DNI:*
Titulación (Degree**):**
Curso, grupo (Course, group**):**
Asignatura (*Subject*):

Distribución de los puntos por ejercicio y tipo

Puntos	*Tipo*			
Ejercicio	**Práctico**	**Teórico**	**Suma total**	**Tiempo recomendado**
1		1	1	10 minutos
2		0,5	0,5	5 minutos
3	2	1	3	40 minutos
~~4~~	~~3~~	~~1~~	~~4~~	~~50 minutos~~
5	0,5	1	1,5	15 minutos
Suma total	~~**5,5**~~ **2,5**	~~**4,5**~~ **3,5**	~~**10**~~ **6**	~~**120 minutos**~~ **70 minutos**

Introducción. Dualidad Onda-Partícula

Nos encontramos en un famoso centro de estudios de partículas y nos han pedido generar un sistema de información, basado en patrones de diseño con Java, con el objeto de simular el comportamiento dual denominado Onda-Partícula. En este caso, los ejercicios están enfocados a simular el comportamiento de dos tipos de Luz, Luz visible y Luz infrarroja y la capacidad de los observadores a poder 'capturar' ambos comportamientos.

Ejercicios

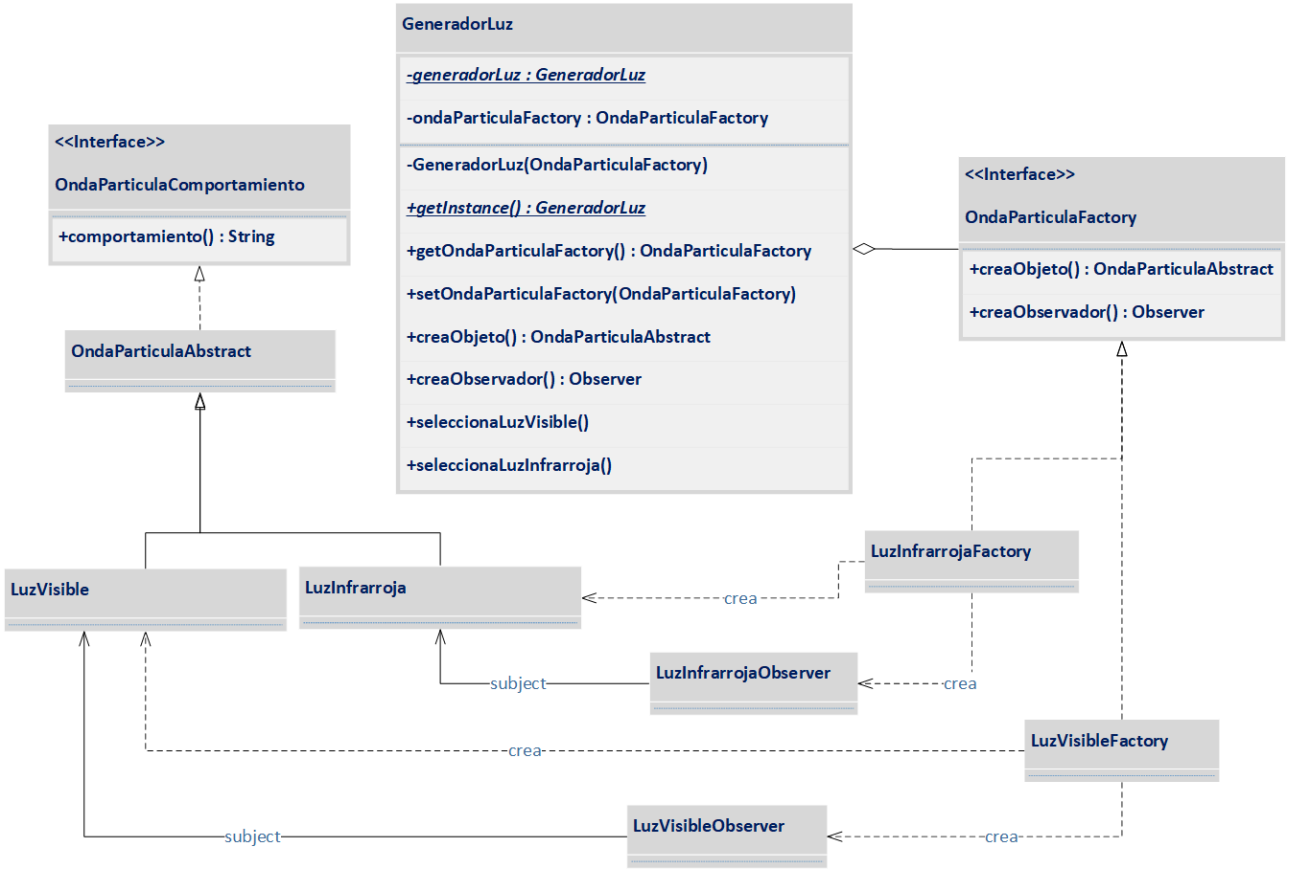
Ejercicio 1 (1 Puntos)
Clasifica los 10 patrones vistos en la asignatura bajo los siguientes tres tipos de patrones, de comportamiento, estructurales y creacionales.

De comportamiento	Creacionales	Estructurales

Ejercicio 2 (0,5 Puntos)
Explica en qué se diferencia el patrón Adaptador del patrón Decorador

Ejercicio 3 (3 Puntos)

El siguiente diagrama representa el diseño de un subsistema que permite gestionar la creación de objetos Luz, y también de los observadores especializados que son capaces de interaccionar con cada tipo de objeto.



- Hay un objeto único creador, **GeneradorLuz**, que gestiona la creación de objetos concretos de Luz, **LuzVisible** y **LuzInfrarroja**. También se encarga de la creación de los observadores concretos para cada tipo de objeto Luz. Un objeto LuzVisible resuelve el método toString con el mensaje "Soy Luz visible" y un objeto LuzInfrarroja resuelve el método toString con el mensaje "Soy Luz infrarroja". Los objetos concretos de Luz, cumplen el comportamiento de la interface OndaParticulaComportamiento.

- La clase **GeneradorLuz** principalmente tiene los siguientes métodos:

    - **creaObjeto**(), cada vez que se invoque este método se creará un objeto concreto de Luz de tipo **OndaParticulaAbstract**
    - **creaObservador(),** cada vez que se invoque este método se creará un un objeto concreto observador de tipo Observer. El objeto concreto dependerá del tipo de Luz que esté seleccionada para la creación de los objetos, entonces, devolverá un objeto LuzVisibleObserver si se están creando objetos LuzVisible, y un objeto LuzInfrarrojaObserver si se están creando objetos LuzInfrarroja.
    - **seleccionaLuzVisible**() , este método establece que todos los objetos concretos que se creen serán de tipo **LuzVisible** y los observadores de tipo **LuzVisibleObserver**.
    - **seleccionaLuzInfrarroja**() , este método establece que todos los objetos concretos que se creen serán de tipo **LuzInfrarroja** y los observadores de tipo **LuzInfrarrojaObserver**.

Se pide:

- **Apartado A (1 punto)**
  Identificar los patrones utilizados e indicar los componentes de cada patrón.

- **Apartado B (2 puntos)**
  Escribe el código de la clase **GeneradorLuz** de tal forma que por defecto se creen objetos de luz visible indicando, mediante un breve comentario, el tipo de delegación que se debe producir en el cuerpo de los métodos **creaObjeto**() y **creaObservador**().

    - Cabecera **(0,1 puntos)**
    - Atributos **(0,25 puntos)**
    - Constructor **(0,25 puntos)**
    - Método getInstance **(0,25 puntos)**
    - Métodos getters y setters **(0,25 puntos)**
    - Método de servicio, creaObjeto **(0,20 puntos)**
    - Método de servicio, creaObservador **(0,20 puntos)**
    - Método selector de objetos, seleccionaLuzVisible **(0,25 puntos)**
    - Método selector de objetos, seleccionaLuzInfrarroja **(0,25 puntos)**

  public class GeneradorLuz Cabecera {
  Atributos
  Constructor
  Método getInstance
  Métodos getter y setter
  Método de servicio, creaObjeto
  Método selector de objetos, seleccionaLuzVisible
  Método selector de objetos, seleccionaLuzInfrarroja
  }

Ejercicio 4 (4 Puntos)

No incluido en el simulacro.

Ejercicio 5 (1,5 Puntos)

La clase **Simulador** permite poner en marcha el sistema de forma sencilla, el método **aplicaTest** se encarga de todo lo necesario para poder observar los objetos Onda/Partícula.

```
public class Simulador {
  private GeneradorLuz                generadorLuz;
  private OndaParticulaAbstract       ondaParticulaLuzVisible;
  private OndaParticulaAbstract       ondaParticulaLuzInfrarroja;
  private Observer                    luzVisibleObserver;
  private Observer                    luzInfrarrojaObserver;
  public Simulador() {
        this.generadorLuz = GeneradorLuz.getInstance();
        this.configuracionUnObservadorPorObjeto();
  }
  private void configuracionUnObservadorPorObjeto() {
        this.creaObjetos();  this.aplicaObservadores();
  }
  private void creaObjetos() {
        this.generadorLuz.seleccionaLuzVisible(); //Crea objetos para luz visible
        this.ondaParticulaLuzVisible = this.generadorLuz.creaObjeto();
        this.luzVisibleObserver = this.generadorLuz.creaObservador();
        this.generadorLuz.seleccionaLuzInfrarroja(); //Crea objetos para luz infrarroja
        this.ondaParticulaLuzInfrarroja = this.generadorLuz.creaObjeto();
        this.luzInfrarrojaObserver = this.generadorLuz.creaObservador();
  }

  private void aplicaObservadores() {
        //Solo un observador para el objeto Luz visible
        this.ondaParticulaLuzVisible.attach(this.luzVisibleObserver);
        //Solo un observador para el objeto Luz Infrarroja
        this.ondaParticulaLuzInfrarroja.attach(his.luzInfrarrojaObserver);
  }
  public void aplicaTest() {
          //Se muestran Resultados
          System.out.println("Resultados");
          System.out.println(this.luzVisibleObserver);
          System.out.println(this.luzInfrarrojaObserver);
  }
 public static void main(String[] args) {
        Simulador simulador = new Simulador();
        simulador.aplicaTest();
  }
}
```

Estos son los resultados por consola:

```
Resultados
Soy un Observador de Luz Visible
He capturado el objeto Soy Luz visible
y esta es mi observación inicial Soy una Onda/Particula y me comporto como una Partícula
y esta es mi observación final Soy una Onda/Particula y me comporto como una Onda

Soy un Observador de Luz Infrarroja
He capturado el objeto Soy Luz infrarroja
y esta es mi observación inicial Soy una Onda/Particula y me comporto como una Partícula
y esta es mi observación final Soy una Onda/Particula y me comporto como una Onda
```

Se pide:

- **Apartado A. (0,25 puntos)**
  Identifica el patrón utilizado, justifica tu respuesta.

- **Apartado B (1,25 puntos)**
  Contesta y desarrolla
  - **(0,25 puntos)** ¿Es posible refactorizar el método **creaObjetos**? Justifica tu respuesta.
  - **(0,25 puntos)** Indica qué bad code smells pudiera mostrar.
  - **(0,25 puntos)** Indica qué técnica de refactorización habría que aplicar.
  - **(0,50 puntos)** Refactoriza el método **creaObjetos** según la técnica de refactorización elegida.