OO2 - Parcial 12/07/2025

Patrones

Se desea implementar un sistema para representar documentos de texto (tipo Markdown). Estos documentos tienen un título, un autor y una sección raíz, que representa su contenido principal. Una **sección** es un elemento con un **título** y una colección de elementos que puede incluir:

- **Párrafos**, que contienen texto (String).
- **Listas ordenadas**, que contienen una secuencia de ítems de texto (String).
- **Otras secciones**.

COMPOSITE

Los documentos deben responder la siguiente funcionalidad:

toString(): debe retornar un string con el título, el autor y la representación de los elementos que contiene, siguiendo estas reglas:

- Un párrafo retorna únicamente el texto que contiene finalizado con "\n".
- Para la lista ordenada retorna cada ítem precedido por su posición, finalizando cada uno con "\n".
- Una sección retorna su título (precedido por "###" y seguido por el separador de línea) y cada uno de los elementos que contiene.

A continuación se muestra la salida de toString() de un documento de Pedro con título "Plan de estudio" con una sección raíz cuyo título es "Orientación a objetos" la cual contiene los siguientes elementos: un párrafo "Temas de la materia:", una lista con los ítems de texto "Patrones de diseño" y "Refactoring de código" y por último, una sección con título "Arquitectura de servicios" la cual contiene un párrafo con el texto "Arranca el semestre que viene"

```
Plan de estudio - Pedro
### Orientación a objetos
Temas de la materia:
1. Patrones de diseño.
2. Refactoring de codigo.
### Arquitectura de servicios
Arranca el semestre que viene
```

buscar(String texto): retorna un booleano si el documento contiene el texto o no. La búsqueda se realiza en la sección raíz del documento y recorre todos sus elementos: el texto de los párrafos, los ítems de las listas y los títulos de las secciones.

traducir(): Debe retornar un nuevo documento en el que **tanto el título como todo el contenido estén en otro idioma**. Para ello, debe construirse una **nueva sección raíz** con su título traducido, y cada uno de los elementos contenidos en ella debe ser representado por una **nueva instancia del mismo tipo** (párrafo, lista u otra sección), pero con el texto traducido. Para obtener la traducción de un texto, utilice el siguiente código:

```
Translator.translate(String texto)
```

Tareas (debe realizar los cuatro ítems para aprobar el tema):

1. Realice un diagrama UML de clases para su solución al problema planteado.
2. Indique claramente en el diagrama UML el o los patrones de diseño que utiliza en el modelo y el rol que cada clase cumple en cada uno.
3. Implemente su solución en Java.
4. Implemente un test para verificar la funcionalidad de imprimir con el ejemplo detallado anteriormente.

Refactoring

OO2 - Parcial 12/07/2025

La plataforma "StremiOO" ofrece contenido multimedia y notifica a los usuarios sobre nuevos lanzamientos. Su notificador funciona de la siguiente manera:

```
1  public class Notificacion {
2      private String destinatario;
3      public Notificacion(String destinatario) { this.destinatario = destinatario; }
4      public void enviar(String tipo, String mensaje) {
5          if (tipo.equalsIgnoreCase("VIDEO")) {
6              System.out.println("Video: " + mensaje + " a " + destinatario);
7          } else if (tipo.equalsIgnoreCase("MUSICA")) {
8              System.out.println("Música: " + mensaje + " a " + destinatario);
9          }
10     }
11 }
12 //Ejemplo de uso:
13 Notificacion notificacion = new Notificacion("usuario@ejemplo.com");
14 notificacion.enviar("VIDEO", "Nuevo lanzamiento disponible");
```

Lo siguiente es el código resultante luego de aplicar modificaciones al código anterior:

```
1  public abstract class Notificacion {
2      protected String destinatario;
3      public Notificacion(String destinatario) { this.destinatario = destinatario; }
4      public abstract void enviar(String mensaje);
5  }
6  public class NotificacionVideo extends Notificacion {
7      public NotificacionVideo(String destinatario) { super(destinatario); }
8      public void enviar(String mensaje) {
9          System.out.println("Video: " + mensaje + " a " + destinatario);
10     }
11 }
12 public class NotificacionMusica extends Notificacion {
13     public NotificacionMusica(String destinatario) { super(destinatario); }
14     public void enviar(String mensaje) {
15         System.out.println("Música: " + mensaje + " a " + destinatario);
16     }
17 }
18 public class NotificacionPodcast extends Notificacion {
19     public NotificacionPodcast(String destinatario) { super(destinatario); }
20     public void enviar(String mensaje) {
21         System.out.println("Podcast: " + mensaje + " a " + destinatario);
22     }
23 }
24 //Ejemplo de uso:
25 Notificacion notificacion = new NotificacionVideo("usuario@ejemplo.com");
26 notificacion.enviar("Nuevo lanzamiento disponible");
```

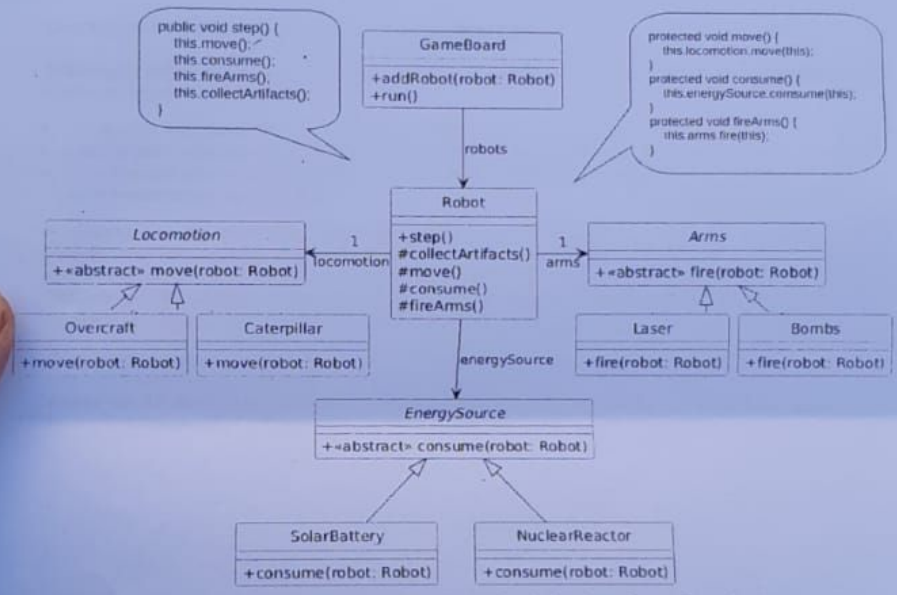
Tareas:

1. Realice el diagrama de clases UML del código luego de los cambios realizados.
2. Analizando el código modificado, ¿Considera que todas las modificaciones realizadas corresponden a refactorizaciones? Fundamente brevemente su respuesta.
3. Liste el/los code smells presentes en el código original que se solucionaron en el modificado.
4. Identifique y nombre los refactorings aplicados. Para cada uno, indique qué code smell soluciona del punto anterior. Explique brevemente los pasos que se realizaron para su aplicación, detallando qué clases se modificaron, qué métodos se crearon o cómo se aplicó.

Frameworks

OO2 - Parcial 12/07/2025

Consideremos el framework que permite generar juegos de luchas de Robots visto en la materia. Los juegos generados por este framework consisten en robots que se ubican en un tablero y realizan una serie de acciones cuando les llega su turno, siempre siguiendo este orden: se mueven en el tablero, consumen energía, disparan su armamento y recolectan recursos. El comportamiento de los robots varía de acuerdo a sus componentes, diferenciándose en tres aspectos: **sistema de locomoción** (Caterpillar u Overcraft), **fuente de energía** (SolarBattery o NuclearReactor) y **armamento** (Laser o Bombs). El siguiente diagrama muestra cómo está conformado este framework.



Para utilizar este framework, se debe crear una instancia de la clase **GameBoard** y luego agregarle instancias de **Robot**. Para crear un robot, se crea una instancia de esa clase y se indica mediante su constructor, que componentes va a utilizar el robot. Luego, se envía el mensaje *run()* a GameBoard y eso produce que se ponga en funcionamiento el juego. Cada vez que se ejecuta una ronda del juego, GameBoard se encargará de llamar al mensaje *step()* de cada Robot que se haya agregado al tablero.

Tareas (debe realizar los cuatro ítems para aprobar el tema):

1. ¿Cuáles son los frozen spot del diseño del framework presentado?
2. ¿Cuáles son los hot spot de este framework? Para utilizarlos, ¿se debe emplear herencia o composición? ¿Hay algún patrón de diseño presente? Indique cual.
3. Se quiere tener un robot que tenga como arma un Láser, como batería un NuclearReactor y como sistema de locomoción Overcraft. Escriba el código para crear un tablero con el robot mencionado. ¿Esta forma de usar el framework es una extensión o una instanciación? ¿Su uso es de caja blanca o de caja negra?
4. Se quiere agregar un nuevo tipo de robot con dos sistemas de armas: uno principal y otro secundario. Estos robots siempre disparan primero el armamento principal y luego el secundario. Explique qué cambios considera necesarios para satisfacer este nuevo requerimiento. ¿Considera que los cambios son una extensión o una instanciación?