Proyecto Final

Departamento Ingeniería Informática y Ciencias de la Computación

Ingeniería Civil Informática

501250-1: Desarrollo orientado a Objetos

Integrantes:

Ignacio Contardo

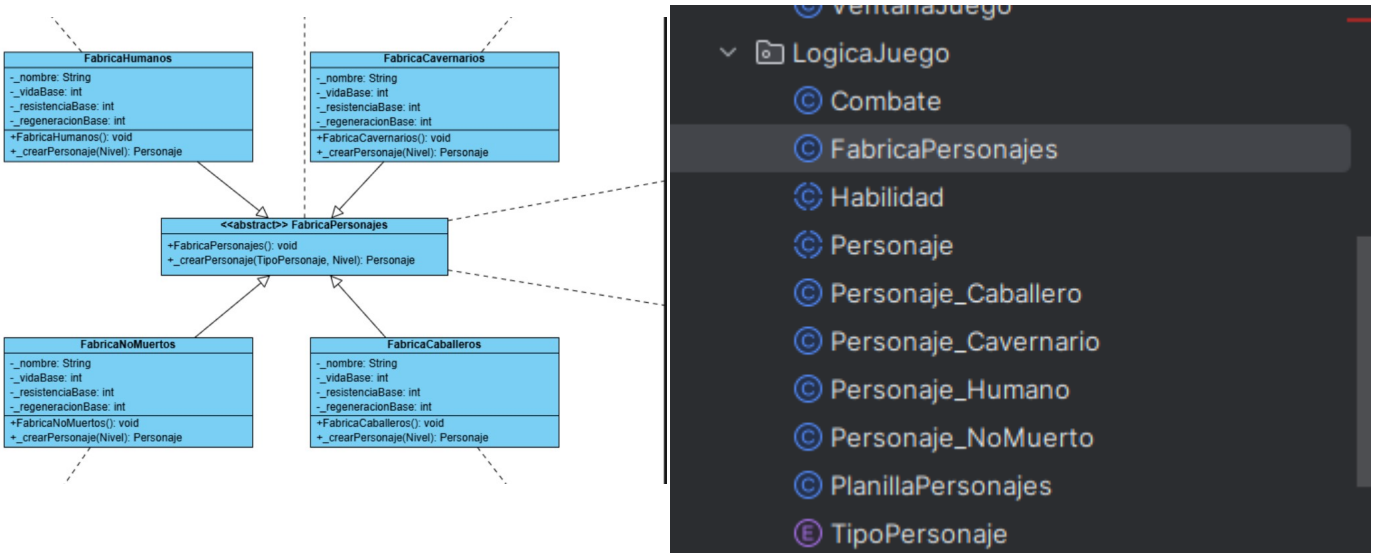
Carlos Salinas Pereira

Francisco Arentsen San Martín

Patrones de diseño utilizados:

Method Factory:

- El método *crearPersonaje* de la clase *FabricarPersonajes* utiliza el patrón Factory Method para facilitar la creación de personajes antes del inicio del juego, permitiendo instan ciar distintos tipos de personajes. Facilitando así en la creación de nuevos tipos de personajes, y aplicando así el principio de Responsabilidad única, y principo de abiero/cerrado.



Singleton:

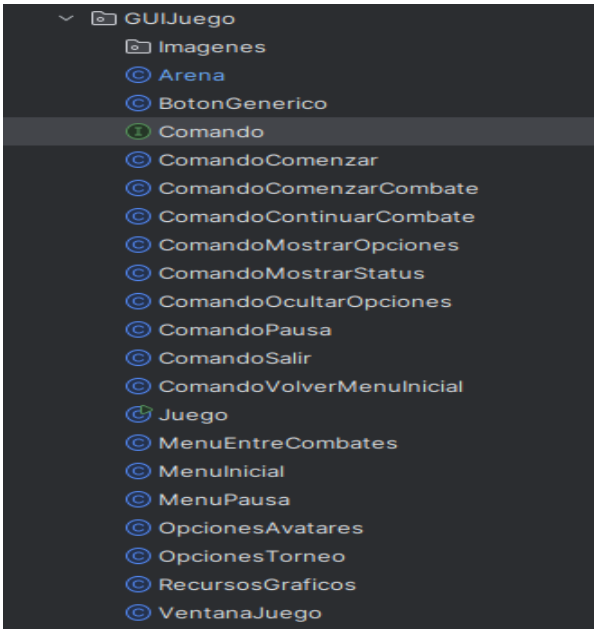
Garantiza única instancia de la ventana principal del juego a lo largo de toda la ejecución del programa, lo que asegura un control centralizado sobre la interfaz gráfica, evita inconsistencias al evitar la creación de múltiples ventanas, y poder acceder, ella por medio de los métodos estáticos.

```
private VentanaJuego(){  1 usage  Farentsens
    this.setPreferredSize(new Dimension( width: 800, height: 600));
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.pack();
    panelActual = new MenuInicial();
    this.add(panelActual);
    setExtendedState(JFrame.MAXIMIZED_BOTH);
    this.setVisible(true);
}
```

Command:

La clase principal para el desarrollo, es la clase de *BotonGenerico, p*oder separar lo de interfaz gráfica, con la lógica del juego. Por medio método de *setConmando, permite qué cada BotonGenerico reciba una acción específica y personalizada representada por un objeto de tipo Comando.*

Pensamos al comienzo utilizar el patrón strategy, pero preferimos, porque se adapta mejor al juego, al no necesitar dinamismo, y el de no utilizar algoritmos muy complejos.



Decisiones Importantes durante el proyecto:

Incorporación tardía del patrón factory:

El planteamiento original era que el usuario adquiría los personajes contra los que se enfrenta en caso de ganar, pero eso hacía innecesario el patrón factory, por eso para incorporarlo se cambio el planteamiento original, del juego.

Cortar funciones, o implementaciones que queríamos hacer, por falta de tiempo.

Antes queríamos manejar cada turno del combate aplicando los efectos de la habilidad que el personaje quisiera usar, pero por falta de tiempo decidimos que el resultado se calcule automáticamente en función de los atributos y habilidades de cada personaje.

Cosas qué se pudieron mejorar:

- Organización, y herramientas que utilizaron para agilizar los uml, muchas veces no sabíamos se crear después o antes los uml para tener una mayor claridad sobre, como estructurar el proyecto.

- Tener consideración desde el inicio a los test, y el javadoc.