Proyecto 2

Wizard 3.0

Equipo: Los Peaky Blinders
Carlos Emilio Castañón Maldonado José Camilo García Ponce Dafne Bonilla Reyes

➢ **Explicación:**

Esta es la versión mejorada del Proyecto 1 Wizard implementado en el curso de EDD.

El proyecto de Wizard consiste en la implementación en Java del juego Wizard, el cual es un juego de cartas en el que solamente deberás acertar el número de veces que ganarás la mano en cada ronda, dependiendo del número de jugadores, habrá más o menos rondas para conseguir puntos.

Funcionamiento de las rondas

Con 6 jugadores habrán 10 rondas, con cinco serán 12 rondas, con cuatro 15 rondas y con tres jugadores 20 rondas. En cada ronda se jugarán un número de manos igual al número de cartas en tu poder, esto ayudará a decidir cuantas podemos ganar.

Objetivo:

✪ Ser el jugador con más puntuación al final del juego.

Conseguiremos puntos cuando acertemos la apuesta del número de manos que hemos ganado durante el turno, lo cual se suma como 20 puntos por acertar, más 10 por el número de manos ganadas.

Sin embargo, en el caso de no acertar el número de manos que ganaremos, nos haría perder 10 puntos por cada valor de diferencia entre nuestra predicción y la realidad.

Desarrollo de una partida:

El juego de cartas tiene 4 palos de diferente color, color el cual puede ser **rojo**, **verde**, **amarillo** o **azul**, numeradas del 1 al 13 y con un Wizard y un Joker, sumando así un total de 60 cartas. Se jugará a un número de rondas predeterminadas por el número de jugadores.

Con seis jugadores habrán 10 rondas, con cinco serán 12 rondas, con cuatro 15 rondas y con tres jugadores 20 rondas. A continuación, se muestra el proceso de desarrollo de una partida:

1 **Preparación de la ronda.**

En cada ronda repartiremos un número de cartas igual al número de ronda en la que nos encontremos.

En la primera ronda, repartiremos una carta, en la segunda ronda dos cartas y así sucesivamente, excepto en la última ronda, en la que repartiremos las 60 cartas entre todos los jugadores. Por cada ronda, se sacará una carta del mazo para que esta sea el color base o ganador de la misma después de repartir las cartas a cada jugador.

2 **Apuesta.**

Cada jugador deberá pronosticar cuantos trucos ganará, esto a partir de ver las cartas que le han tocado y analizando cuantas de ellas le ayudarán a ganar la ronda.

El valor de las cartas es el número que llevan impreso, cuánto más alto el número, más alto será el valor que tendrán.

El Wizard ganará a cualquier carta y el Joker solo servirá para desvincular el color base de la ronda. Simultáneamente, esta es la más equitativa, así los jugadores últimos no tienen ventaja sobre los primeros.

3 **Juego de las manos.**

En esta fase empieza el juego con las cartas en sí, cada carta que juegue un jugador será acompañada por una carta de los demás y esto conformará una mano.

Habrá tantas manos como cartas tenga un jugador en su poder. Empezando por el jugador inicial, jugará una de sus cartas y esperará a que los demás jueguen la suya para intentar ganar el truco.

Sabiendo que el valor de las cartas es el que nos hará ganar, iremos poniendo cartas de manera estratégica, sin olvidar que también el color de la carta del mazo ganará a cualquier otro color y valor.

✪ **Ejemplo:**

➢ **El valor del mazo es azul.**

El jugador inicial juega un 5 **verde** , los demás jugadores para ganar la mano deberán jugar una carta **verde** con un valor superior, una carta azul de cualquier valor o ser el primero en jugar un Wizard.

Quién quiera que juegue primero un Wizard, ganará la mano directamente, dará igual lo que jueguen los demás.

El jugador que juegue un Joker dejará esa mano sin valor al color de la ronda sacada del mazo.
Y pasará a ser la siguiente carta el color que marcará esa mano. Así se irán jugando las manos hasta que no queden cartas en nuestro poder, cada truco lo empezará el jugador que gane el anterior.

4 **Puntuación de la ronda.**

Ahora es el momento de puntuar. Veremos si hemos acertado con las predicciones sobre el número de trucos ganados. Si hemos acertado en la predicción ganaremos 20 puntos más 10 puntos por mano ganada.

Si no hemos acertado en nuestra predicción, perderemos 10 puntos por cada punto de diferencia entre nuestra predicción y la realidad que ha sucedido.

5 **Final del juego.**

La última ronda es la más ***difícil***. En esta ronda ya no habrá color de triunfo y todos jugaremos con muchas cartas en nuestro poder, con lo que la predicción será muy arriesgada. Nos podrá dar o quitarnos muchos puntos el acertar o errar en la predicción.

El jugador que tenga más puntos ganará la partida.

Uso del programa:

– Compilar desde `src/`

```
javac *.java cards/*.java games/*.java players/*.java views/*.java
```

– Ejecutar Servidor desde `src/`

```
java Proyecto2Servidor <#jugadores> <puerto>
```

– Ejecutar Cliente desde `src/`

```
java Proyecto2Cliente <host> <puerto>
```

– Generar la documentación desde `src/`

```
javadoc -d docs *.java cards/*.java games/*.java players/*.java views/*.java
```

Implementación:

Para este proyecto decidimos usar los siguientes patrones:

- **MVC:** Lo usamos, ya que el proyecto 2 lo pedía y para poder dividir en partes nuestro programa. El modelo son las cartas y las cosas que tienen que ver con estas, como el mazo. El controlador son las clases que tienen que ver con el funcionamiento del juego y los jugadores. Por último, la vista son las clases que se llaman vista y son para la comunicación con el usuario.
- **Strategy:** Lo usamos para poder implementar los colores y valores de las cartas, ya que estos atributos hacen las mismas cosas, pero como hacen las cosas es diferente entre los tipos de colores y valores.
- **Factory:** Lo decidimos implementar para poder facilitar la creación de las cartas y así poder ahorrarnos algo de tiempo en la creación del mazo de cartas para el juego.
- **Iterator:** Elegimos este patrón para poder mostrar las cartas de su mano al cliente, pero sin pasarle el mazo directamente (por razones de seguridad), pero debido a que se nos complicó pasar un iterador por medio de un socket, solo iteramos las cartas del lado del servidor y pasamos las cartas (en forma de texto) por el socket.
- **Proxy:** Usamos este patrón para poder darle al servidor un jugador, pero sin darle el jugador que el cliente tiene, además necesitamos este patrón para poder tener un jugador del lado del cliente y otro del lado del servidor que trabajen en conjunto.
- **Observer:** Este patrón, decidimos implementarlo para poder enviar un mensaje de lo que paso en el juego (cartas jugadas, puntuaciones, apuestas, etc.) a todos los jugadores, y así los jugadores son los observers y el juego es el subject.
- **Facade:** De esta implementación no estamos tan seguros, pero nos pareció algo útil para poder crear todo el juego y sus partes necesarias, sin tener que poner muchas líneas de código en el `main` del servidor.

Estos fueron los patrones que decidimos implementar para el proyecto, aunque es importante mencionar que hay código que no forma parte de un patrón, ya que sentimos que no era necesario o no supimos si lo era.

Advertencias de Uso:

Se recomienda escribir solo cuando se le pida al usuario, debido a que la forma en que leemos la entrada del usuario es leyendo la siguiente línea y no la última (se intentó, pero esto causó algunos problemas). En caso contrario, esto podría causar mensajes de entrada inválida o elegir opciones que no se ingresaron cuando se pidió la entrada.

Existe un caso (que encontramos, y no supimos solucionar) en el cual los clientes pueden quedarse en el limbo (se quedan esperando por un mensaje y no pasa nada). Esto ocurre cuando se cierra el servidor (`Ctrl` + `c`), mientras se le pide algo a un cliente y seguido cerrar la terminal del cliente (al que se le pide), sin poner una respuesta. Esto causa muchos problemas, por lo tanto, se recomienda no intentar replicarlas por el bien de la humanidad.