Proyecto 2

Bonzi Buddy

Integrantes:

- Salazar Enrique Luis Alberto 321155405
- Gabriel Candiani Pérez 318268534
- Angel Maldonado Gerardo De Jesus 320188268

Problemática:
La empresa de videojuegos nintendogs decidió abarcar un nuevo mercado, las personas introvertidas, que tienen problemas para socializar con las personas, por lo que decidió crear un juego nuevo para poder hacer que las personas puedan conocerse más entre ellas, como también brindar una forma divertida de jugar en las fiestas. Por lo que decidió contratar al equipo de desarrollo de software Bonzi Buddy para desarrollar dicho juego, Dicho juego consiste en un juego de cartas donde en cada turno se le muestra una carta con una pregunta a una persona para que ella la responda con el propósito de poder conocerla más y poder ganar puntos, se pidió que se agregara una dinámica para poder castigar a los jugadores que no contestaran una pregunta y estos tengan como castigo responder las mismas preguntas y además tener que cumplir un reto.
Nintendogs también pidió que puedan agregar eventos al juego con el propósito que los jugadores se vean recompensados ganando puntos o puedan verse afectados si no cumplen con los eventos y sean castigados. Este juego debe de ser capaz de notificar a los jugadores durante los eventos quienes lo deben de cumplir

Patrones:

Observer:
El uso de este patrón se usa para poder llevar a cabo notificaciones a los jugadores durante la partida, nos ayuda a hacer notificaciones a todos los jugadores como también a solo una parte de ellos como lo es sólo notificar a los castigados o solo los activos. Como lo es en los eventos que solo notifica a quienes lo tienen que cumplir ya sea Castigados o Todos por mencionar un ejemplo.

State:
Los jugadores pueden tener dos estados los cuales son Activos y Castigados, el uso de este patrón facilitara poder manejar las diferentes acciones que pueden realizar los jugadores en base a el estado como lo es los retos y preguntas al mismo tiempo que solo le salen a los Castigados y el manejo de que pasa si cumplen o no su turno, ya que, los regulares no contestan se convierten en castigados y los castigados no pueden avanzar en su objetivo de volver a ser regulares, por eso state nos ayuda a solucionar esta problemática de cambiar su comportamiento en tiempo real de los jugadores.

Iterator:
El juego utilizará distintas estructuras de datos para poder abordar el problema, como tener una cola para poder hacer el orden en que los jugadores pasaran y se les dará una carta, Uso de listas para poder usar observer en la forma en que se notifican a los jugadores sobre actualizaciones del juego como finalmente Arreglos para las barajas de cartas. El uso de este facilitara el manejo de todos estas estructuras de datos como también esconde como se manejan las estructuras que el programa usa.

Prototype:

Se usa para la creación de las Tarjetas de los mazos, al tener que crear muchas cartas podemos usar este patrón para hacer una clonación de objetos en las Tarjetas y poder hacer de manera eficiente la creación de todas las cartas de todos los mazos. De esta forma hacemos que sea menos costoso esta parte del codigo.

MVC:

El uso de este patrón nos ayuda a seccionar el codigo para poder separarlo segun su funcionamiento,
Tenemos model que nos ayuda a mantener toda la información de los Jugadores, poder llevar los datos de estos de mejor forma, como también tiene comunicación con nuestra base de datos en txt.
Tenemos controller que nos ayuda a poder verla lógica en base a las acciones que los jugadores llevan durante la partida y este mande la información a model para que haga dichos cambios como también tener información con vista y mostrar la información correcta a los jugadores.
Vista se encarga de los mensajes que se me mostraran al usuario y como también llevar la comunión con el usuario acerca de las entradas que este puede dar a lo largo de la partida como recibir nombre de los jugadores o si cumplieron con lo que se les pidió.
Hacer el proyecto de esta forma nos ayuda a tener un sistema flexible ante cambios que puedan surgir, así como facilitar el mantenimiento del mismo.

Clases involucradas con cada Patron

Prototype: Tarjeta, Baraja, Clonable
State: Jugador, Castigado, Regular, EstadoJugador, Estado
Observer: Jugador, Observer, Sujeto, Model
Iterator: Iterator, ListIterator, QueueIterator, ArrayIterator, Baraja, ListaJugadores, Model, LectorPreguntas

MVC: Model, ModelInterface, Vista, VistaInterface, Controller, ControllerInterface y todas las demás son parte de Controller aunque los jugadores se almacenan en Model

¿Cómo ejecutar el programa?
Desde la terminal que estés ocupando en el momento (de Windows o Linux) deberás estar en la ruta donde está el proyecto ej. "C: \User\Desktop\Proyecto\Proyecto2_BonziBuddy", de ahí deberás pasar a la carpeta "src". Tu ruta será algo así:
"C: \User\Desktop\Proyecto\Proyecto2_BonziBuddy\src" para este punto ya podrás ejecutar el programa con los siguientes comandos:
1ro) "javac *.java"
2do) "java Main"
Esos son los pasos suficientes para correr el programa.

Modo de uso:
El programa está diseñado para que le diga al usuario vea que entradas debe de elegir por lo que es sencillo usar el juego.
Al principio pide nombres de los jugadores, posteriormente se tendrán que ingresar entradas de tipo si/no para avanzar a través del juego.

Versiones de java usadas:
21.0.7
23.0.2

Notas del equipo:

1. El diagrama de casos de uso solo muestra una persona,el juego puede iniciarse con una sola persona, por este motivo se ve una sola persona solo se ve un jugador en el diagrama, pero la idea es jugar varias personas.
2. El programa está diseñado para que los txt puedan ser modificados y asi poder cambiar las preguntas, retos o eventos y el programa usará estos nuevos archivos txt, para hacerlo tienes que poner por cada línea una pregunta/reto/evento, eviten usar ñ por favor