Informe TP1 - Monopoly-morphic

Grupo 12

En este trabajo se presentaba el problema de realizar un juego estilo monopoly utilizando buenas prácticas de programación y el paradigma orientado a objetos.

Para la visualización de los casilleros utilizamos JavaFx.

Algunas hipótesis que tomamos fueron:

- Las rentas de todas las propiedades son iguales
- Las loterias dan todas el mismo premio
- Los casillero de multa cobran todos lo mismo
- Las casas y hoteles valen lo mismo en todos los terrenos
- Los transportes solo se pueden comprar y vender (no se pueden hipotecar)
- Si el jugador compra un hotel, rompe la regla de cantidad de casas por casilla

Uno de los desafíos fue manejar correctamente los distintos estados de entidades del juego (jugador, la disponibilidad de las propiedades).
Para solucionar el manejo de estado, utilizamos el patrón State que se encarga de delegar la responsabilidad de cambiar de estado a una clase concreta.

Luego, para manejar las distintas operaciones que puede realizar un jugador, utilizamos “Opcionales” que enmascaran la funcionalidad que tienen dentro y según el momento del juego, le pasamos al jugador una instancia concreta para que ejecuten la lógica que contienen.

Por último el manejo de las propiedades lo delegamos en una clase auxiliar llamada “AdministradorDePropiedades”. En esta clase, está contenida la lógica para habilitar la construcción de casas en los terrenos. Esta lógica supuso un desafío, ya que no podíamos preguntarle a cada terreno cuántas casas tiene (pues se rompería el principio de Tell Don’t Ask). Entonces se nos ocurrió implementar una clase auxiliar “Semáforo” que cuando está en verde habilita la opción de construir una casa en ese Terreno. A medida que a cada Terreno de un grupo se le va construyendo una casa, se van cambiando a rojo los semáforos de ese Terreno para así no sobrepasar la diferencia de 1 entre cantidad de casas en un mismo grupo.
Lo mismo sucede para las casas, pero en el sentido inverso. Para vender una casa el semáforo debe estar en rojo y una vez vendida el semáforo se pone en verde y no podes vender en esa propiedad hasta que vuelva a rojo.

Como conclusión, hacer uso de los patrones de diseño que vimos y forzarnos a respetar los principios de diseño, permitió desarrollar un programa escalable, robusto, mantenible, flexible.