UNIVERSIDAD DE SANTIAGO DE CHILE

Informe Laboratorio #3 – Proyecto Semestral
Paradigmas de Programación

Paradigmas de Programación

Profesor: Roberto González Ibanez

Clemente Aguilar Osorio

19960801-0

Junio 2022

Introducción

El siguiente informe tiene como objetivo describir y analizar el problema planteado para el Laboratorio #3 del Proyecto Semestral de Paradigmas de Programación. A partir de esto, describir el diseño planteado para la solución escogida, diferencias respecto al paradigma abordado en el primer laboratorio y los resultados obtenidos, concluyendo finalmente respecto al trabajo realizado.

Descripción del problema

El problema propuesto consta de implementar una réplica del popular juego de mesa Dobble. Esto considera los siguientes elementos para jugar: mazo(s) de cartas, jugadores y una instancia de juego. Cada carta contiene una cantidad fija de elementos, y cada par de cartas tienen un solo elemento en común. El ciclo de juego consta en identificar el elemento en común entre las cartas mostradas. Es posible jugar distintos modos de juegos que varían en la cantidad de cartas, mazos y jugadores.

Descripción del paradigma

Para el desarrollo de este laboratorio, el problema es diseñado bajo el paradigma orientado a objetos. Esto implica que los elementos del juego (cartas, mazo, jugadores y juego) van a ser representados por clases, que a su vez son instanciadas en el programa en forma de objetos. Cada una de estas clases posee atributos y métodos. Además, ciertas clases se relacionan entre sí mediante distintas reglas.

La ventaja del paradigma orientado a objetos es que, a diferencia de los paradigmas utilizados en los laboratorios anteriores, permite una abstracción menos rígida de los elementos de juego y cómo son tratados, permitiendo una estructura cercana a cómo funcionan y se ven las cosas en la realidad.

Análisis del problema

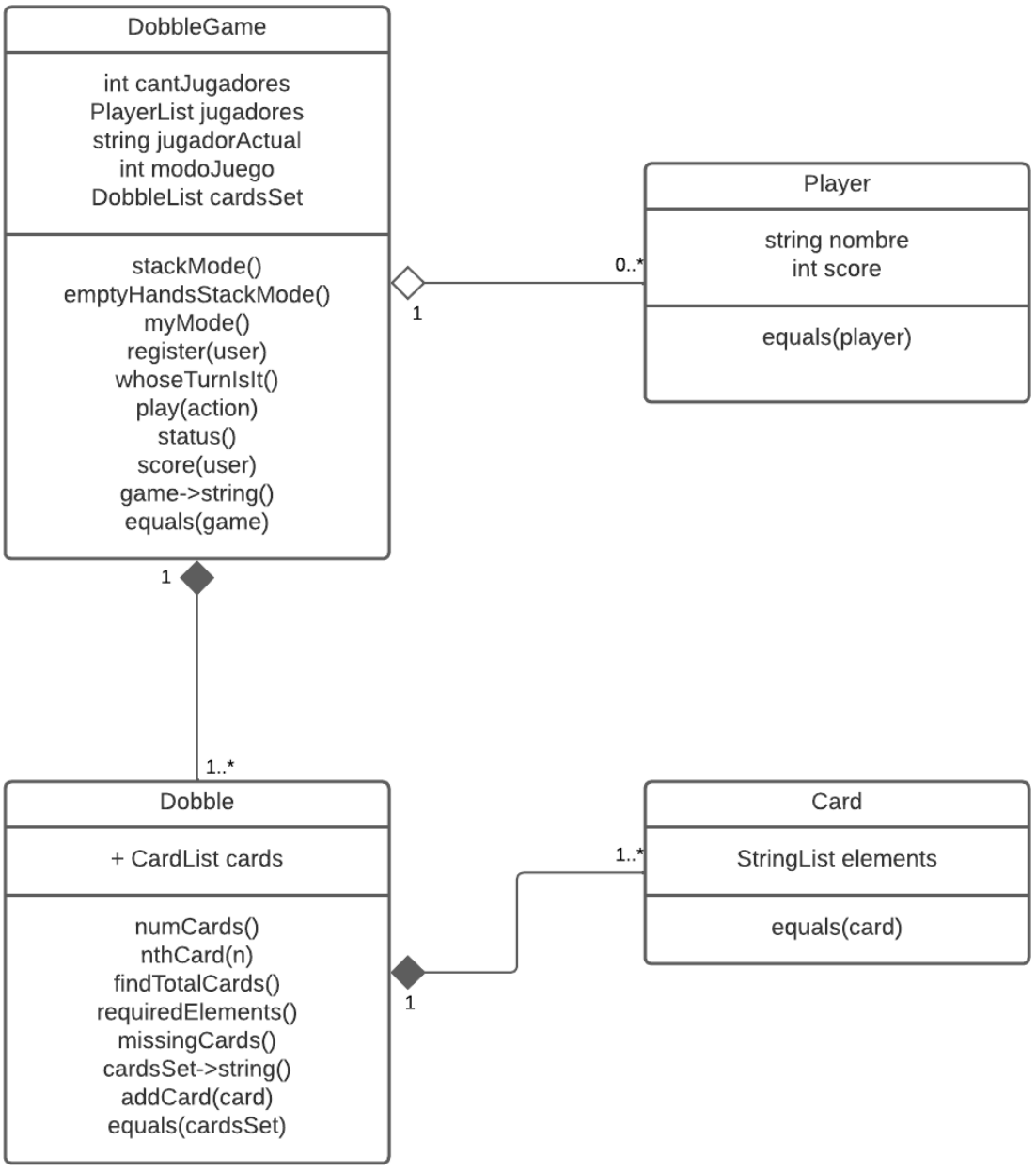
Para poder representar los elementos, acciones y pasos que componen un juego de Dobble, es necesario abstraer distintos conceptos y procesos. Estos conceptos serán descritos en un orden que va desde lo esencial a lo más complejo.

Inicialmente, al tratarse de un juego de cartas, se necesita un conjunto de funciones que den lugar a la estructura “card”. Una vez definida la estructura carta, se necesita un mazo de cartas para jugar, el cual será “Dobble”. Lo último que falta para poner un juego en marcha, son jugadores y una instancia de juego.

Para esta iteración del laboratorio, además, se apunta a simular un juego de Dobble a través de la consola de forma más interactiva.

Diseño de la solución

El primer paso en la solución corresponde a la creación del diagrama UML. Este diagrama permite visualizar las clases que componen el diseño, al igual de las relaciones entre dichas clases. Este diagrama preliminar es el Diagrama de Análisis, y será la primera aproximación a la solución planteada:



En este diagrama se pueden apreciar 4 clases:

- DobbleGame
- Player
- Dobble
- Card

DobbleGame: Esta clase representa un juego de Dobble. Sus atributos son cantidad de jugadores, lista de jugadores dentro del juego, nombre del jugador actual, modo de Juego y una lista de mazos con los que se está jugando.

Player: Esta clase modela a cada jugador. Sus atributos son nombre del jugador y puntaje actual del jugador.

Dobble: Esta clase representa al mazo de cartas del juego Dobble. Su único atributo es una lista de cartas Dobble.

Card: Esta clase modela una carta de juego. Su único atributo es una lista de elementos que contiene la carta.

Aspectos de implementación

Para la implementación del diseño, se trabaja con el ambiente de trabajo Apache NetBeans IDE 14, trabajando con el lenguaje Java, incluido en OpenJDK versión 11. Se integró al proyecto la herramienta Gradle (y Gradle Wrapper).

Resultados

Del trabajo realizado, se pudieron identificar y definir las clases a utilizar, con sus constructore,s getters, setters y funciones automáticas que entrega el entorno de trabajo.

Puntaje a asignar (denotado en paréntesis junto a cada apartado):
0: No realizado.
0.25: Implementación con problemas mayores (funciona 25% de las veces o no funciona)
0.5: Implementación con funcionamiento irregular (funciona 50% de las veces)
0.75: Implementación con problemas menores (funciona 75% de las veces)
1: Implementación completa sin problemas (funciona 100% de las veces)

1. Clases y estructuras que forman el programa (1)

1. Menú interactivo por terminal (0)

2. Constructor de juegos (0)

3. Register (0)

4. Play (0)

5. toString() (1)

6. equals(Object o) (1)

7. vs CPU Mode (0)

8. demo Mode (0)

Conclusión

Si bien no se pudo implementar la solución, se considera que las clases definidas con los atributos y métodos considerados en el diagrama de análisis permiten un buen precedente para la entrega de la solución en el comodín de este laboratorio.

Referencias

1. Dore, M. (2021, 30 diciembre). *The Dobble Algorithm - Micky Dore*. Medium. https://mickydore.medium.com/the-dobble-algorithm-b9c9018afc52
2. A. (2020, 3 junio). *The Dobble Algorithm*. 101 Computing. https://www.101computing.net/the-dobble-algorithm/