Informe de Laboratorio 1: Implementación de juego «Dobble» en Racket

Hans Villarroel

18/04/2022

Capítulo

Índice general

Capítulo 1

Introducción

Este informe corresponde al 1er laboratorio de la asignatura *Paradigmas de Programación*. En este primer laboratorio se empleara el Paradigma Funcional, usando el lenguage de programacion `Racket`, un descendiente del lengua ge `Scheme`. Este se usara a través de el IDE [Integrated Development Environment] DrRacket.

1.1. Descripción del Problema

Se pide crear una implementacion de Dobble. Este juego consiste en un mazo de cartas con 55 cartas (57 originalmente) donde dos cartas cualquieras comparten **solo** un elemento entre si. Para jugar Dobble existen distintos tipos modos. El modo mas conocido consiste en que los jugadores deben tomar dos cartas del mazo y el primero que identifique cual simbolo se encuentre en ambas cartas toma la primera carta y la guarda. Despues, se toma otra carta y se repite el proceso hasta que se acabe el mazo. Una vez que se acaben las cartas, el jugador que tenga la mayor cantidad de cartas ganará. Para representar esto en Racket hay que tener en cuenta que la generacion de cartas debera respetar la condicion de que solo se repita el simbolo una vez entre cualquier par de cartas. Ademas hay que tomar en cuenta que tipo de modo de juego seleccionara el usuario.

1.2. Descripción del Paradigma

Para crear la representacion de «Dobble» se usara la Programacion Funcional, Un paradigma de la programacion con un gran enfoque en las funciones, evitando datos mutables. Las funciones se podran usar como argumento y estas podran regresar otra funcion (funciones de alto nivel). Este paradigma esta basado sobre el cálculo lambda, el cual sigue una simple regla de sustituir variables y un esquema simple para definir las funciones. Las principales caracteristicas de este paradigma son:

- **Funciones Puras**: Siempre produciran la misma salida con los mismos argumentos sin importar otros factores
- **Recursion**: No se hace uso de ciclos `for` o ciclos `while`, y en cambio se hara uso de la recursion, donde se llamara la funcion una y otra vez hasta que se llegue a un caso base.
- **Funciones son de primera clase**: En el paradigma funcional, las funciones se podran usar como argumentos y se podran regresar desde otras funciones. A las

funciones que toman otras funciones como argumentos y/o regresen otras funciones seran llamadas **funciones de alto nivel**.

- **Inmutabilidad**: Las variables seran inmutables, osea que no se podran modificar una vez sean creadas.

Estas caracteristicas tendran sus ventajas y desventajas. Las ventajas de estas son que haran que las funciones puras sean mas facil de entender ya que no cambiaran de estado y solo dependeran de la entrada que reciban. Pero tendran la desventaja que podrian requerir de mas rendimiento en caso de mal utilizar la recursion. Además, usualmente se esta acostumbrado a usar loops para programar, lo cual hara pasar a solamente usar recursion algo dificil.

Capítulo 2

Desarrollo

2.1. Analisis del Problema

Para este proyecto se tendra que tener en cuenta los aspectos fundamentales del juego «Dobble». Este juego, como ya ha sido mencionado, tendra un mazo de cartas con 57 cartas, donde 2 cartas cualquieras tendran siempre un elemento en comun. Para este hay que tener en cuenta las bases matematicas del juego, pero en este caso se nos entrega un algoritmo escrito en `Javascript` el cual facilitara la creacion del algoritmo en `Racket`. El mazo creado sera representado como una lista de listas, donde cada lista sera una carta diferente. El usuario podra pedir diferentes acciones respecto al mazo o a una carta:

- Verificar si un mazo es valido [dobble?]
- Ver el numero de cartas en el mazo [numCards]
- Buscar que carta hay en x posición [nthCard]
- Encontrar cuantas cartas necesita para un mazo valido a partir de una carta [findTotalCards]
- Encontrar el numero de elementos que necesita para armar un mazo valido a partir de una carta [requiredElemets]
- Ver que faltas cartan de un mazo para armar uno valido [missingCards]
- Transformar el mazo a una representación en strings. [cardsSet→string]

Para permitir al usuario poder jugar el juego hay que tener en cuenta que este podra:

- Ingresar cantidad de jugadores [register y numPlayers]
- Elegir modo de juego [mode, stackMode, myMode y emptyHandsStackMode]
- Elegir que hacer en su turno [play y pass]
- Terminar el juego cuando quiera
- Ver de quien es el turno y el puntaje [whoseTurnIsIt? y status]