Universidad de Santiago de Chile

Facultad de Ingeniería

Departamento de Ingeniería Informática

Curso: Paradigmas de Programación

Proyecto semestral de laboratorio (Primer semestre 2025)

Implementación de Juego de Mesa Estilo Monopoly
Desarrollado en Racket utilizando Programación Funcional

Estudiante: Fernando Ríos
Profesore: Gonzalo Martínez
Fecha: 07-05-2025

Tabla de Contenidos

1. INTRODUCCIÓN

Descripción breve del problema

Este proyecto está basado en el desarrollo de un juego de mesa similar a Monopoly, que simula la compra de propiedades, el pago de impuestos, y la construcción de casas y hoteles. El juego permite a dos jugadores moverse por un tablero, realizar acciones como comprar propiedades y pagar multas, y competir por la bancarrota. Cada jugador maneja su dinero, propiedades y se enfrenta a eventos que pueden alterar el curso del juego.

Descripción breve del paradigma y los conceptos aplicados

El proyecto utiliza el paradigma de programación funcional con el lenguaje Racket. A lo largo del desarrollo, se utilizan conceptos clave como los Tipos de Dato Abstractos (TDA) para modelar entidades como jugadores, propiedades y el tablero de juego. Además, se implementan funciones y recursividad para simular los turnos, el lanzamiento de dados, y las interacciones entre los jugadores y las propiedades en el tablero. La elección de este paradigma se debe a la capacidad de Racket para gestionar estados inmutables y manejar estructuras de datos complejas de manera eficiente.

2. ANÁLISIS DEL PROBLEMA

Análisis respecto de los requisitos específicos

El juego debe cumplir con varios requisitos funcionales, entre ellos:

1. **Simulación de turnos**: Cada jugador debe poder lanzar dados, moverse por el tablero y realizar acciones basadas en la posición en la que caen.
2. **Compra de propiedades**: Los jugadores pueden adquirir propiedades si tienen suficiente dinero y estas no están ya en posesión de otro jugador.
3. **Construcción de casas y hoteles**: Los jugadores deben tener la posibilidad de construir casas y hoteles si poseen propiedades y han cumplido con las condiciones necesarias.
4. **Bancarrota**: Un jugador pierde el juego si su saldo llega a cero, y el sistema debe detectar este estado y finalizar el juego.
5. **Cárcel y tarjetas de salida**: Los jugadores deben poder caer en la cárcel y, si lo desean, usar una carta de salida para salir de allí.

Además, se incluyen requisitos no funcionales como la eficiencia en la simulación de turnos y la claridad del código.

3. DISEÑO DE LA SOLUCIÓN

Enfoque de solución y técnicas utilizadas

La solución está organizada en varios módulos (TDAs) que encapsulan la lógica relacionada con cada aspecto del juego: jugadores, propiedades, tablero, y cartas. La descomposición en módulos facilita el mantenimiento y la extensión del juego. El flujo del juego sigue los siguientes pasos:

1. **Creación de los objetos jugadores y propiedades**: Cada jugador tiene atributos como nombre, dinero, y propiedades. Las propiedades se crean con un precio, renta y estado de hipoteca.
2. **Simulación de turnos**: Cada turno consiste en lanzar los dados, mover al jugador por el tablero y realizar las acciones correspondientes (comprar, construir, pagar multas).
3. **Manejo de la bancarrota y cárcel**: Si un jugador se queda sin dinero, el juego termina. Si un jugador cae en la cárcel, pierde su turno, pero puede pagar una multa o usar una carta para salir.

Diagrama de flujo del juego

- Los jugadores lanzan los dados.
- Se mueve al jugador según el resultado de los dados.
- Se verifica si la casilla corresponde a una propiedad.
- Si es una propiedad disponible, el jugador decide si comprarla o pasar.
- Si se cumplen las condiciones, puede construir casas y hoteles.
- Se revisa si un jugador está en bancarrota al final de su turno.

4. CONSIDERACIONES DE IMPLEMENTACIÓN

Aspectos de implementación

El proyecto está implementado en Racket, un lenguaje funcional que facilita la manipulación de listas y la creación de funciones puras. Se ha optado por Racket debido a su capacidad para manejar estructuras inmutables y su soporte para la creación de funciones de alto nivel.

El código está organizado en varios archivos que definen los TDAs de los jugadores, las propiedades, el tablero y las cartas. Se utilizaron bibliotecas estándar de Racket para la manipulación de listas y la generación de números aleatorios.

5. INSTRUCCIONES DE USO

Instrucciones de uso

1. **Abrir el archivo:** Abrir el archivo script_base_182384798_FERNANDO_RIOSVEGA.rkt desde el IDE DrRacket en una version 8.16 o superior:
2. **Correr el programa**: Ejecutar el programa con el botón Run desde DrRacket
3. **Visualizar la simulación**: Visualizar la simulación de un juego de monopoly, los turnos, movimientos de jugador, tiro de dados, entre otros, en la consola de interacción de DrRacket
4. **Identificar la finalización del juego**: El juego finaliza cuando un jugador entra en bancarrota. La función jugador-esta-en-bancarrota verifica si esto ocurre.
5. **Testeo de nuevas simulaciones:** Puede modificar el contenido del archivo script_base_182384798_FERNANDO_RIOSVEGA.rkt para evaluar nuevos escenarios de juego:

   Ejemplo de modificación:

A) ; Carlos construye casa en una propiedad que ya posee
   (define g3.5 (juego-jugar-turno g3 (juego-lanzar-dados 0 0) #t #f #f #f))
   g3.5
B) Utilice (juego-lanzar-dados 0 0) para simular un turno sin movimiento
C) #t activar la construcción de casa si se cumplen las condiciones

6. RESULTADOS OBTENIDOS

Resumen de resultados

El juego ha sido completado y probado con varios escenarios, incluyendo:

- Compra de propiedades y construcción de casas y hoteles.
- Pagos de impuestos y multas.
- Manejo correcto de la bancarrota y la cárcel.

Se realizaron pruebas unitarias sobre las funciones clave, y los resultados fueron en su mayoría exitosos. Se encontró que algunas funciones relacionadas con la hipoteca no fueron completamente implementadas.

7. EVALUACIÓN COMPLETA

Conclusiones

El desarrollo de este proyecto fue una buena oportunidad para aplicar conceptos de programación funcional a un caso concreto y entretenido como lo es un juego de mesa estilo Monopoly. A través del uso de Racket y estructuras como los TDAs, se logró construir un sistema que simula de forma clara y ordenada el flujo de turnos, las decisiones de los jugadores y los eventos del juego.

Uno de los mayores aprendizajes fue cómo dividir correctamente el problema en componentes manejables, lo que facilitó tanto la implementación como las pruebas. El lenguaje Racket, aunque desafiante al principio, demostró ser adecuado para este tipo de simulaciones, sobre todo por su enfoque en funciones puras y estructuras inmutables.

Si bien algunas funcionalidades más específicas (como las hipotecas) no fueron completamente desarrolladas, el sistema general funciona correctamente y cubre los aspectos esenciales del juego. En resumen, este proyecto permitió reforzar tanto el uso del paradigma funcional como habilidades de diseño y organización de código, dentro de un contexto más práctico y lúdico.

8. REFERENCIAS:

- Documentación Racket: https://docs.racket-lang.org/
- Gonzalo Martinez – Repositorio Github “USACH Programming Paradigms”: https://github.com/USACH-GM-Programming-Paradigms/
- Gonzalo Martinez – Drive de Apuntes y Respaldos de Clases de ramo Paradigmas de Programación: https://docs.google.com/document/d/1BtHmBewGpxjRiFPa9fIDzWdhjDyDUPpmNvZQ9icaGh0/edit?tab=t.0
- 2025_01 Laboratorio (General): https://docs.google.com/document/d/1cj6CsEN2ThY_L2Wk880z7Y03Kk1vxpoqY9LVmUQPcuQ/edit?tab=t.0
- 2025_01 Laboratorio 1: https://docs.google.com/document/d/1c3Foi2RuNZimvkPhHDyuI8NUgG3TljqGEgl-ZQZQXLY/edit?tab=t.0#heading=h.xcu38lrz6h81