UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio 1

Paradigma Funcional

Paradigmas de Programación (sección 13310-0-A-1)

Nombre: Sahid Nazar Jeria

Profesor: Gonzalo Martinez

Sección: A-1

Fecha: 28/04/2025

Indice

1. Introducción

Este informé detalla la evolución de la creación del proyecto Capitalia, una recreación de un juego de mesa inspirado en Monopoly desarrollado en Racket utilizando el enfoque funcional como base principal. El objetivo de esta experiencia práctica era poner en práctica los fundamentos del paradigma funcional para representar las distintas entidades del juego tales como jugadores y propiedades mediante tableros y cartas específicas. Mediante un enfoque modular se desarrollaron diversos TDAs que se interrelacionan para simular situaciones como desplazarse por el tablero de juego y adquirir propiedades; usar cartas de suerte y comunidad; estar en la cárcel y declararse en bancarrota.

2. Descripción del Paradigma

El enfoque funcional se basa en la utilización de funciones puras y la inmutabilidad de los datos, además del empleó amplio de recursión en lugar de las estructuras iterativas tradicionales. En esta iniciativa se pusieron en práctica los aspectos fundamentales del paradigma funcional.

- Lista de definiciones de TDAs.
- Funciones que devuelven valores y argumentos en su uso.
- Utilización de la recursividad para el manejo de listas.
- División modular del código utilizando require´ y provide´.

Estos elementos facilitaron la creación de un diseño limpio que seguía las mejores prácticas del paradigma sin generar efectos secundarios no deseables.

3. Análisis del Problema

Para poder llevar a cabo una simulación efectiva del juego Capitalia fue esencial abordar varios requisitos específicos relacionados tanto a la lógica de juego como a la representación del estado y la modularidad del código. Se analizaron detenidamente los siguientes aspectos fundamentales para resolver el problema:

Para mantener la representación del estado del juego sin variables que cambien constantemente y modelar cada transición como una nueva estructura de datos siguiendo los principios del paradigma funcional.

Cada participante tenía la capacidad de arrojar dados, desplazarse por el tablero de juego y adquirir propiedades, asimismo debían abonar tarifas de alquileres y edificar viviendas y hoteles según fuera necesario, también tenían la posibilidad de ser encarcelados o liberados y emplear cartas especiales.

Se necesitaba una función principal llamada “juego-jugar-turno” que manejara por completo el turno según la información proporcionada y devolviera un estado de juego actualizado.

La simulación determinista requería que las semillas de los dados fueran establecidas de antemano para reproducir resultados previsibles y facilitar la validación.

Los eventos especiales como caer en la cárcel o sacar una carta de bancarrota debían ser gestionados sin modificar directamente el estado actual del juego sino utilizando funciones puras.

Además de eso, la tarea implicaba la creación de scripts de prueba claros para demostrar cómo funciona el sistema, mostrando ejemplos de juegos para 1, 2 y 3 jugadores, tal como lo indicaban las instrucciones del laboratorio.

4. Diseño de la Solución

La solución se diseñó basándose en TDAs independientes que representan cada aspecto primordial del juego.

- Jugador: Es una entidad que constaba de una lista que incluía su identificador único, nombre, dinero, posesiones, posición en el tablero, situación carcelaria y cantidad de cartas restantes para salir de la cárcel.
- Propiedad: Se describe como una lista que incluye campos para identificación (ID), nombre del propietario y precio de alquiler o venta de la misma, también se registra el dueño actual junto al número de propiedades en posesión y su respectivo estado (si es una casa o un hotel), así mismo si está o no hipotecada.
- Tablero: Contiene un listado de propiedades ubicadas en diferentes posiciones e incluye cartas y casillas especiales como la cárcel y la comunidad.
- Juego: Combina todos los elementos mencionados anteriormente junto a la mecánica de turnos y gestión de dinero del banco y ajustes generales.

El juego avanza mediante la función “juego-jugar-turno” que se ejecuta de la siguiente manera:

1. Verifica si el jugador se encuentra en prisión y toma las medidas correspondientes.
2. Evalúa la casilla obtenida para determinar si es una propiedad y verifica si se trata de una compra o alquiler o si se construyó.
3. Si se encuentra en una casilla especial durante el juego, puedes simplemente saltar tu turno.

Se crearon también funciones secundarias como “juego-actualizar-propiedad”, “jugador-pagar-renta”, “jugador-construir-casa” y “jugador-esta-en-bancarrota” para organizar la lógica y hacerla más fácil de usar en diferentes situaciones del juego.

5. Aspectos de Implementación

Para llevar a cabo de manera efectiva el desarrollo del proyecto Capitalía en Racket se tuvo que considerar la forma de representar cada elemento del juego utilizando las estructuras apropiadas del paradigma funcional. En esta situación específica se decidió emplear listas como la estructura principal de cada tipo abstracto de datos (TDA).

Por ejemplo, una persona que juega se muestra como una lista que incluye su identificador, nombre, cantidad de dinero, propiedades, posición en el tablero, si está o no encarcelado y cuántas cartas para salir de la cárcel posee.

El enfoque funcional sugiere la creación de respuestas mediante la combinación y utilización de funciones puras en lugar de métodos tradicionales basados en objetos o estados mutables. Por tal motivo el software fue dividido en secciones independientes donde cada una cuenta con sus propios métodos para crear, seleccionar y actuar.

En numerosas ocasiones se empleó la recursión natural para recorrer listas de propiedades o jugadores. Asimismo, se hicieron uso de funciones auxiliares y selectores específicos para acceder a los campos de los TDAs como lo harían los métodos tradicionales getters.

La función principal juego-jugar-turno hace mucho uso de la composición de funciones al aplicar subprocesos como “juego-manejar-carcel”, “jugador-mover”, “jugador-comprar-propiedad”, entre otros, para retornar un nuevo estado del juego según los eventos acontecidos en el turno.

6. Instrucciones de Uso

Para ejecutar los TDA junto a los script, se deben seguir los siguientes pasos:

1. Colocar todos los archivos `.rkt` y `README.md` en la misma carpeta.

2. Abrir DrRacket y cargar el archivo `script_base_20879921_Sahid_NazarJeria.rkt`.

3. Ejecutar el archivo para iniciar el juego.

4. Verificar en consola los turnos, posiciones y eventos impresos.

5. El juego termina cuando un jugador entra en bancarrota.

En la sección de anexos se mostrará el resultado obtenido por consola al ejecutar el script base, entre otras funciones.

7. Resultados y Autoevaluación

En los requisitos funcionales se consiguió la completa implementación del sistema requerido; esto incluyó la creación de las diferentes entidades como jugadores y propiedades del tablero de juego junto al manejo de la lógica central mediante la función "juego-jugar-turno".

Se crearon tres scripts distintos, uno base que es el proporcionado en las instrucciones del laboratorio y dos de prueba para 1 jugador y 3 jugadores respectivamente, todos ellos fueron validados tanto con los ejemplos provistos como con casos adicionales creados para evaluar el comportamiento del juego.

En el proceso de validación se confirmó que las características principales funcionan como se espera, los jugadores se desplazan según los resultados de los dados lanzados, pueden adquirir propiedades al igual que construir casas y hoteles, por otra parte se reconocieron correctamente las situaciones de cárcel y quiebra del juego. La utilidad de la función "jugador-esta-en-bancarrota" resultó clave para señalar el fin del juego.

Se llevaron a cabo pruebas manuales utilizando las semillas de dados descritas en las instrucciones para supervisar los movimientos en el tablero y validar manualmente los cambios en dinero y propiedades de cada jugador al final de cada turno. Todos los scripts se ejecutaron sin problemas y produjeron los resultados previstos, no se detectaron errores en la consola en ninguno de los script ejecutados. Por lo tanto, podemos decir que el desarrollo fue exitoso desde el punto de vista funcional.

8. Conclusiones

El uso del enfoque funcional resultó eficiente al modelar un juego de mesa como sistema complejo. La no variabilidad y la pureza de las funciones simplificaron la detección de fallos y garantizaron una coherencia en el proceso.

A pesar de que este enfoque impide ciertas operaciones como cambios directos o control de estados compartidos, la estructura modular hizo posible una solución comprensible. El proyecto Capitalia ilustró cómo una actividad de este tipo puede ser un excelente ejercicio para practicar la programación funcional y para el entendimiento del paradigma.

9. Referencias

HtDP: How to Design Programs. (2023). MIT Press. Disponible en: https://htdp.org

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S. (2018). The Racket Guide (v8.8). Racket Documentation. https://docs.racket-lang.org/

Universidad de Santiago de Chile (2025). Laboratorio 1 - Paradigmas de Programación. Departamento de Ingeniería Informática.

Anexos

Anexo 1

Resultados por consola del script base

```
===== CAPITALIA =====

TURNO 1: Carlos
Valor Dado 1: 5
Valor Dado 2: 6
Turno 1 completado.
TURNO 2: Ana
Valor Dado 1: 2
Valor Dado 2: 3
Turno 2 completado.
TURNO 3: Carlos
Valor Dado 1: 3
Valor Dado 2: 4
Turno 3 completado.
TURNO 4: Ana
Valor Dado 1: 5
Valor Dado 2: 6
Turno 4 completado.
TURNO 5: Carlos
Valor Dado 1: 1
Valor Dado 2: 2
Turno 5 completado.

¿Jugador 1 en bancarrota?: #f
¿Jugador 2 en bancarrota?: #f
>
```

Anexo 2

Funciones Relevantes

La función juego-jugar-turno ejecuta todo el ciclo de turno del juego, desde lanzar los dados hasta comprar propiedades, pagar renta, etc.

- (define (juego-jugar-turno juego semillas comprarPropiedad_or_construirCasa construirHotel pagarMultaSalirCarcel usarTarjetaSalirCarcel)

La función lanzar-dados simula el lanzamiento de dos dados usando semillas predefinidas. Retorna un par de valores entre 1 y 6.

- (define (lanzar-dados semilla1 semilla2)