Diseño de Software Orientado a Objetos

Trabajo Práctico Obligatorio 2

Integrantes:
- Lara Combina
- Ignacio Dias Gundin

Captura de Requisitos

- El juego de la vida consiste en un “juego” sin jugadores, donde la única entrada es el estado inicial de las células en un tablero.
- El tablero consiste en una matriz de N x M celdas, donde en cada casilla existe una célula con dos posibles estados: viva o muerta.
- A partir de un conjunto de reglas aplicadas sobre las células se determina el próximo estado del tablero. Las reglas son las siguientes:
    - Si una célula está viva y tiene dos o tres vecinos a su alrededor se mantiene viva.
    - Si una célula está muerta y tiene menos de tres o más de tres vecinos a su alrededor se mantiene muerta.
    - Si una célula está viva y tiene menos de dos vecinos, o más de tres, se muere.
    - Si una célula está muerta y tiene tres vecinos, esta pasa a estar viva.

-El tablero generado deberá ser mostrado en consola.

Patrones Utilizados

Observer (Salida en consola)

En esta primera versión nuestro juego tiene una única forma de mostrar la salida, que es a través de la pantalla. Creemos que esto a futuro podría cambiar, pidiéndonos que mostremos una salida mediante una GUI, o que además de la salida se nos requiera recopilar estadísticas del juego, por eso es que decidimos aplicar el patrón Observer de manera temprana.
Este patrón utilizará a Game como clase observada (“subject”) y la interfaz IOutput como observadora (“observer”). La clase ConsoleDisplay implementará IOuput y una interfaz IObserver, siendo el motivo por el cual tiene los métodos show() y update().

Command (Reglas del Juego)

A cada célula se le aplica un conjunto de reglas para modificarlas, las cuales además podrían verse modificadas a futuro. Para implementar esto, la idea consiste en que el tablero tenga como atributo un conjunto de reglas, que pueden cambiarse dinámicamente.

Clases

- **Cell** (implementa **ICell**). Las celulas tienen un atributo CellType type. Este **CellType** es un enumerado con los valores DEAD y ALIVE.
- **Board** (implementa **IBoard**). La posición de las células se representa con una matriz estática de N x M. El tablero debe ser inicializado con un estado inicial, en nuestro caso donde todas las células tienen el mismo estado pasado como parámetro. Además, el tablero guarda todas las reglas a aplicar sobre las células, basada en la cantidad de vecinos que tiene. Los métodos disponibles del tablero serán los siguientes:
    - putCell(int row, int col):void
    - getCell(int row, int col):ICell
    - nextState():IBoard (retorna el proximo tablero en donde se ha aplicado las reglas a todas las células).
    - numberOfNeighbours: int
- **Game**. Esta clase será observada por la consola. Los métodos públicos son:
    - Initialize(board)
    - play(int numberOfGenerations): void (se ejecuta n veces la regla de avanzar el tablero)
    - notifyObservers(): este método se comunicará con las posibles salidas, informando el estado del tablero actual.
- **ConsoleOutput** (implementa **GameObserver**): mostrará por pantalla los resultados.
- I**Rule.** Las reglas abarcan todas las posibilidades de estados o transformaciones, por lo que a toda célula se le debe aplicar exactamente una regla. En nuestro caso las clases de reglas son **RuleStayAlive, RuleStayDead, RuleBorn** (cambia de muerta a viva)**,** y **RuleDie** (cambia de viva a muerta).