Assignment 4, Design Specification

SFWRENG 2AA4

April 13, 2021

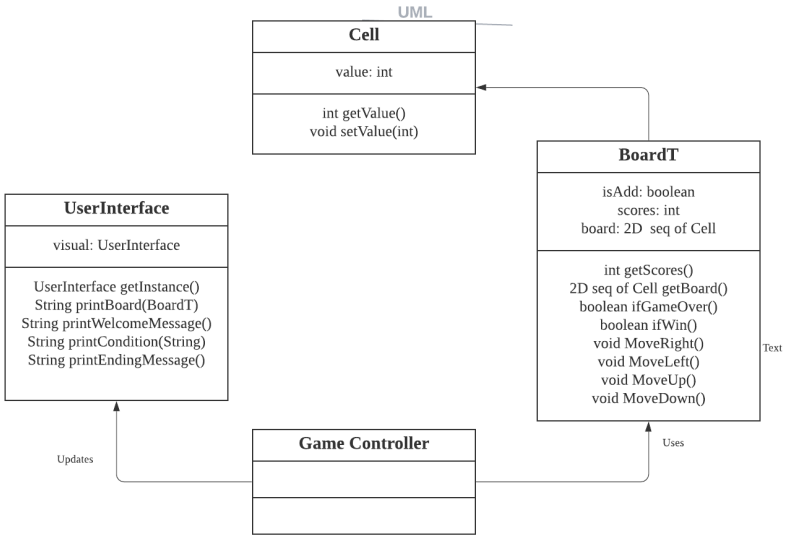
This Module Interface Specification (MIS) document contains modules, types and methods for implementing the game *2048*. 2048 is often played on a plain 4×4 grid, with numbered tiles that slide when a player moves them using the four arrow keys. Every turn, a new tile randomly appears in an empty spot on the board with a value of either 2 or 4. Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move. The game can be launched and play by typing `make demo` in terminal.

reference:https://en.wikipedia.org/wiki/2048$_{(video_game)}$

1 Overview of the design

This design applies Module View Specification (MVC) design pattern, Strategy design pattern and Singleton design pattern. The MVC components are *GameController* (controller module), *BoardT* (model module), and *UserInterface* (view module).

An UML diagram is provided below for visualizing the structure of this software architecture



The MVC design pattern is specified and implemented in the following way: the module *BoardT* stores the state of the game board and the status of the game. A view module *UserInterface* can display the state of the game board and game using a text-based graphics. The controller *GameController* is responsible for handling input actions.

For *GameController* and *UserInterface*, use the getInstance() method to obtain the abstract object.

Likely Changes my design considers:

- Data structure used for storing the game board
- The visual representation of the game such as UI layout.
- Change in peripheral devices for taking user input.
- Change in game ending conditions to adjust the difficulty of the game.
- Design more modes to have more fun such as set a timer.

Cell Module

Module

Cell

Uses

N/A

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Cell	$\mathbb{N}$	Cell	IllegalArgumentException
getValue		$\mathbb{N}$	
setValue	$\mathbb{N}$	Cell	IllegalArgumentException

Semantics

State Variables

value: $\mathbb{N}$

State Invariant

None

Access Routine Semantics

Cell(v):

- transition: value := v

- output: $out := self$
- exception: $exc := (\neg(v = 0||v = 2||v = 4||v = 8||v = 16||v = 32||v = 64||v = 128||v = 256||v = 512||v = 1024||v = 2048) \Rightarrow IllegalArgumentException)$

getValue():

- transition: None
- output: $out :=$ value
- exception: None

setValue():

- transition: value $:= v$
- output: None
- exception: $exc := (\neg(v = 0||v = 2||v = 4||v = 8||v = 16||v = 32||v = 64||v = 128||v = 256||v = 512||v = 1024||v = 2048) \Rightarrow IllegalArgumentException)$

Board ADT Module

Template Module inherits EndCondition

BoardT

Uses

Cell

Syntax

Exported Types

None

Exported Constant

None

Exported Access Programs

Routine name	In	Out	Exceptions
BoardT		BoardT	
getScores		$\mathbb{N}$	
getBoard		seq of (seq of Cell)	
ifGameOver		$\mathbb{B}$	
ifWin		$\mathbb{B}$	
MoveRight		BoardT	
MoveLeft		BoardT	
MoveUp		BoardT	
MoveDown		BoardT	

Semantics

State Variables

board: sequence of (sequence of Cell)
isAdd: $\mathbb{B}$
scores: $\mathbb{N}$

State Invariant

None

Assumptions

- The constructor BoardT is called for each object instance before any other access routine is called for that object.

Access Routine Semantics

BoardT():

- transition:

$$\text{board} := (\langle \begin{matrix} \langle \text{Cell}(0)_0, ..., \text{Cell}(0)_3 \rangle \\ \langle \text{Cell}(0)_0, ..., \text{Cell}(0)_3 \rangle \\ \langle \text{Cell}(0)_0, ..., \text{Cell}(0)_3 \rangle \end{matrix} \rangle \Rightarrow createNew() \Rightarrow createNew())$$

  $isAdd, scores{=}true, 0output$:out := self

- exception: none

getScores():

- transition: none
- output: *out* := scores
- exception: none

getBoard():

- transition: none
- output: *out* := board
- exception: none

ifGameOver():

- transition: none
- output: true if values for all Cell in board are not zero and there is no cell whose value is as same as its left Cell or its right Cell or its unpper Cell or its lower Cell.

- exception: none

ifWin(x, y):

- output: $out := \exists(s : \text{Cell} | s \in board : s.getValue() = 2048$
- exception: none

MoveRight():

- transition: If there are Cells whose value is 0, all the Cells on their left whose value is not 0 will be moved one position to the right. For any Cell in the board, if the value of its left most Cell whose value is not 0 is as same as the Cell itself, these values for the two Cells would be added to the right Cell. If a Cell whose value is 0 appears, steps repeat.
- output: None
- exception: None

MoveLeft():

- transition: If there are Cells whose value is 0, all the Cells on their right whose value is not 0 will be moved one position to the left. For any Cell in the board, if the value of its right most Cell whose value is not 0 is as same as the Cell itself, these values for the two Cells would be added to the left Cell. If a Cell whose value is 0 appears, steps repeat.
- output: None
- exception: None

MoveUp():

- transition: If there are Cells whose value is 0, all the lower Cells whose value is not 0 will be moved one position up. For any Cell in the board, if the value of its lower most Cell whose value is not 0 is as same as the Cell itself, these values for the two Cells would be added to the upper Cell. If a Cell whose value is 0 appears, steps repeat.
- output: None
- exception: None

MoveDown():

- transition: If there are Cells whose value is 0, all the upper Cells whose value is not 0 will be moved one position down. For any Cell in the board, if the value of its upper most Cell whose value is not 0 is as same as the Cell itself, these values for the two Cells would be added to the lower Cell. If a Cell whose value is 0 appears, steps repeat.
- output: None
- exception: None

Local Functions

getEmpty: $seqof(seqofCell) \rightarrow seqofCell$

getEmpty(board) $\equiv (s : \text{Cell} | s \in board \wedge s.value = 0 \Rightarrow s \in list$

createNew: board $\Rightarrow board$

createNew: randomly make the value for one Cell whose value is 0 in the board 2 or 4. Chance for appearance of 2 is 25%.Chance for appearance of 4 is 75%

UserInterface Module

UserInterface Module

Uses

None

Syntax

Exported Types

None

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
getInstance		UserInterface	
printBoard	BoardT	String	
printWelcomeMessage			
printCondition	String		
printEndingMessage			

Semantics

Environment Variables

window: A portion of computer screen to display the game and messages

State Variables

visual: UserInterface

State Invariant

None