Assignment 4 Specification

SFWR ENG 2AA4

Conway ADT Module

Template Module

Conway

Uses

N/A

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Conway	input_file : string	Conway	invalid_argument
get_matrix		Conway	
get_value	$\mathbb{N}$, $\mathbb{N}$	$\mathbb{B}$	out_of_range
set_value	$\mathbb{N}$, $\mathbb{N}$		out_of_range
remove_value	$\mathbb{N}$, $\mathbb{N}$		out_of_range
next			
output_graphics			invalid_argument

Semantics

Environment Variables

input : File of game with given input for a 7 by 7 board

State Variables

matrix: Seq of Seq of $\mathbb{B}$

State Invariant

$|matrix| = 7$
size of each row is 7

Assumptions & Design Decisions

- The Conway constructor is called before any other access routine is called on that instance. Once a Conway has been created, the constructor will not be called on it again.
- the vector class is used to represent a sequence.
- The value in the vector is only boolean type meaning only true or false (true means there is a point in the board and false means there is not a point on the board).
- The board is a 7 by 7 board.
- For the input file the board is represented by o's and x's (o's meaning there is no point and x's indicating points)
- For better scalability, this module is specified as an Abstract Data Type (ADT) instead of an Abstract Object. This would allow multiple games to be created and tracked at once by a client.
- Gettter and setter functions are provided for testing and gives users more flexibility in terms of creating the board.
- The rows and columns are in range 0 to 6 (not 1 to 7).
- My version of Conway's game of life works exactly like this version https://academo.org/demos/conways-game-of-life/

Access Routine Semantics

Conway($input\_file$):

- transition: reads data from the file input that is associated with the string input_file. After it uses this file to create a sequence of sequences of booleans based on what is on the file (x's become true and o's become false). Next the state variable matrix is assigned this specific sequence and the variable is called matrix since this sequence acts like a matrix.

  An example of a correct file would be:

  o o o o x x o
  o o o o x x o
  o o o o x x o
  o o o o x x o
  o x x x o x o
  o o o o o x o
  o o o o o o o

- exception: if the file cannot be found the exception called invalid_argument will be raised

get_matrix():

- output: $out := matrix$
- exception: None

get_value($row, column$):

- output: $out := matrix[row][column]$
- exception: if row or column is less than 0 or greater than 6 it will raise an out_of_range error.

set_value($row, column$):

- transition: $matrix[row][column] := true$
- exception: if row or column is less than 0 or greater than 6 it will raise an out_of_range error.

remove_value($row, column$):

- transition: $matrix[row][column] := false$
- exception: if row or column is less than 0 or greater than 6 it will raise an out_of_range error.

next():

- transition: Loop through sequence (index i) and all sequences inside the sequence (index j). have a counter to count how many points around the desired point are equal to true. Next for each iteration if counter is greater than 3 or less than 2 and matrix[i][j] is equal to true than matrix[i][j] becomes false.

ouput_graphic():

- transition: goes through the matrix and creates a textfile called output.txt which is board similar to the input textfile. If matrix[row][column] = false then it would become o in the textfile and if it is true then it would become x in the textfile.
- exception: if file cannot be written an invalid_argument will be raised.

Critique of Design

Consistency

One way I tried to make this consistent is by only using a 7 by 7 matrix, since it would make it easier for me to design and would also help avoid unusual user inputs such as having a large/infinite matrix and also having a very small matrix such as 1 by 1 or 0 by 0. Ways I implemented this rule is for the constructor it only reads the first 49 inputs in the text file leading to only be 7 by 7 and in the getter and setter methods if you input a row or column that is not in this range it will throw an exception to prevent this from happening.

Essentiality

In terms of essentiality everything in this specification is essential except for get_value. The function get_value is not essential since you can get the value by using get matrix (Ex. object.get_matrix[0][0]). The reason why I added the function get_value is because some individuals may not to think to use get_matrix to obtain the value and get_value makes it much easier to get the specific value the user wants.

Generality

When looking at generality I believe my design is not that general because everything is specific to the game. One example is that an individual can only use a 7 by 7 matrix which gives them less variety. Also in terms of the next function I just followed this version https://academo.org/demos/conways-game-of-life/, so there not much ambiguity for edge cases such as what do for points in the corners etc...

Minimality

In terms of minimality almost every function is minimal (Ex. for the set_value it only sets the value to true and doesn't return anything) but one function that is no minimal is the constructor. The constructor is not minimal since it does two things, reads the file and makes the matrix. The reason why I decided to not make a read function was I only read a text file in the constructor and no where else.

Cohesion

In terms of each function every function has a relationship with the matrix meaning it would have high cohesion. For the get_value and get_matrix it shows the matrix or value

to the user and for the other functions it involves some form of updating the matrix. And for output_graphic it converts the matrix and puts the matrix into the text file.

Information Hiding

Finally for information hiding the user can only call the available functions in the specification and cannot see any of the code inside each method. One thing lacking is there are no local functions but this is because there isn't that much need for local functions.