Report For HW2

Author: Abdullah Mesut Güler

Number: 32815522378

Purpose of the Project: This project is built to have necessary methods in order to find a path from 1 to 32 inside a matrix and print that found path. If there is not a path of 1 to 32 the program will print "There is not a path". The matrix will be read from a txt file.

Documentation for the Software:

- To make this project, 'Apache Netbeans IDE 15' and Java version 'jdk19' are used.
- Firstly the **HW3_1Interface.java** is created with the following method declerations. These methods are implemented in **HW3.java class**.
- public void **read_file**(String **filepath**);
- public String **find_path**();
- public void **print_path**(String **mypath**);
- print **path_to_file**(String **filepath**);
- In addition to these methods, following private class and method are implemented in **HW3.java class**.
- private static IntArrayStack **look_around(**IntArrayStack **theStack);**
- **private static class IntArrayStack;**
- The **HW3.java class** has the following fields.
- **private static int numberOfRows;**
- private static int **numberOfColumns;**
- private int **theMatrix[][] =** new int**[numberOfRows][numberOfColumns];**
- private static int **paddedMatrix[][] =** new int**[numberOfRows+2][numberOfColumns +2];**
- private static final int **dest = 32;**
- private static IntArrayStack **rstStack;**
- private static String **resultedPath;**
- Following libraies are used in **HW3.java class**.
- **import java.io.BufferedReader;**
- **import java.io.IOException;**
- **import java.util.EmptyStackException;**
- **import java.io.BufferedWriter;**
- **import java.io.FileReader;**
- **import java.io.FileWriter;**

! IMPORTANT NOTE

- Since an ordinary int array stack is used, the x and y coordinates of a matrix cannot be stored inside one cell of the stack as a tuple or list. So when an index of a point is going to be pushed into the stack it is pushed as (m*10 + n) where m = row and n = column. And when the index is going to be popped out from the stack m is restored as (popped / 10) and similarly n is restored as (popped % 10) . This restoration is done by **theRow()** and **theColumn()** methods which are defined inside **IntArrayStack.java** class.

- public void **read_file**(String **filepath**)
- The matrix written inside a txt file is read line by line the buffered reader. While reading the number of rows and columns are also counted. Then these number written into **numberOfRows** and **numberOfColumns** fields, respectively. Also the matrix inside the txt file is stored in initMatrix and this reference is written into **theMatrix** field. This field of matrix is initialized with **numberOfRows** and **numberOfColumns**.
- This method takes the path of the file that is to be read as a parameter and does not return anything, which is why those references written into the fields, to be able to use those inside other methods.

- public String **find_path()**
- Inside this method, firstly, all of the 4 egdes of the matrix is framed by -1's to be able to check edge points. The resulting padded matrix is written into **paddedMatrix** field. Then from **IntArrayStack** class "myStack" object is created. Then possible starting point of 1's inside the matrix are found and pushed into the "myStack". After this point with an if statement it's checked whether there is a 1 inside the matrix or not. If not then "There is no path" will be returned. If there is/are 1's then the myStack is send to **look_around** as an input. This method will return the the stack that is containing found path or if there isn't any path the returned stack will be empty. If returned stack has the path then the path will be popped out from this stack and written into a string, "thePath". This "thePath" will be written into **resultedPath** field. If the returned stack is empty then "There is no path" message will be written into bot "thePath" and **resultedPath**.
- This method will return "thePath" which either contains the found path or error message.
- private static IntArrayStack **look_around(**IntArrayStack **theStack)**
- This method is used to move inside the matrix to find the path from 1 to 32. It will take the stack that contains found 1s inside the matrix. And at the end it returns the final stack.
- Inside a while loop the path will be tried to found. At the begining of the loop, the current point we are on inside matrix will be "peeked" from the stack, the row and the column indexes. "cond" variable is forced to be "true" in the beginning of the loop. With the following 8 different if statements, each for 8 different neighboors of the current point. Each will be checked and if that neighboor satisfies the condition of being the next point of the path its index will be pushed into the stack. 8 if statements work separetely if there are more than 1 possible next points both will be pushed into the stack. Inside these each if statements "cond" is forced to be "false". After these 8 if blocks, with another if block if the 32 is reached or not, if reached it will break out from the loop. With the following if block, if the "cond" is true, which means we couldn't find any appropriate neighboors and the current point is a deadend. This point is marked "-1" and it will be popped out from the stack and loop continues.
- At the beginning of the loop "condOfFail" will forced to be "true", which means the stack is now empty and there is no path.
- After the while loop, if "condOfFail" is "true" then an empty stack will be returned.
- Now, if condOfFail is false then our path is inside the stack but upside down. And also there is a possibility of repating same points. To eliminate these, this stack will be dumped into "finalStack" while removing same points. And this stack is returned.
- public void **print_path(**String **mypath)**
- This method will take the found path as an input and it will simply just print out either the path or the "There is no path" message.

- public void **print_path_to_file**(String **filepath**)

- This method will take the path of the file which we want to write the found path or failure message inside as an input. It will write the string stored inside the file using buffered writer.

! IMPORTANT NOTE

- If you try to write the resulting path to the txt file that you have read the matrix from, the matrix inside the txt file will be deleted and the path will be written into that file.

END OF THE REPORT