Assignment 1 report

Tic-Tac-Toe

Introduction

The purpose of this assignment is to write a java code for a tic-tac-toe game. The code should generate this game as a console application. Game should be (6 rows)*(7 columns). In this report, the description and algorithms of the code attached with all methods included shall be discussed.

Discussion

In the beginning, a java class was created under the name of “TTT” (abbreviation for tic-tac-toe). Two integer constants were declared for the rows “ROWS” and columns “COLUMNS”. ROWS constant was set to 6 and the COLUMNS constant was set to 7. Then a 2d string called “boxes” was declared with size of (rows*columns). This “boxes” string holds all the positions of the game where a player can play(the string elements are either “ X “ or “ O ” or ” “ where there is 1 space before and after each symbol to make the game look better) .

The first methods created was **public void emptyBoxes()** which initializes all the elements of the boxes string to an empty string “ ”. The function works by looping in all rows and columns and setting the value of all elements to “ ”(three space characters).

Then a method created has a prototype **public void printBoxes()**. This method starts by creating a new string called “layout”. Then it loops in all rows and columns. It concatenates an element of the boxes string starting from first element followed by a “|” character (to separate columns) unless it is the last row element (it then concatenates the element only without the “|” character) . After reaching the end of the row, the string "\n---+---+---+---+---+---+---+---\n" is concatenated to the layout string to create a separator between each row and the other except for the final row such that the grid of the game is opened from all directions. Then at the end of the method, the “layout” string is printed.

The third method is **public boolean playTurn(int i, int j, String input)**. This method accepts 3 parameters. First 2 parameters are integers representing the row and column respectively of the element to be placed (position of “X” or “O” to be played). The third parameter is a string which is the character to be used depending on whose turn it is. (either “X” or “O”) . Inside the method, an if condition is used to check if the position of the element to be entered is empty or occupied . if empty, it inserts the character to be played between two space characters all in the entered position(i and j). If the player inputs in a

position which was already occupied, the function will print to him a message telling them that the box is already occupied and returns false. If the position is empty the function will work normally and return true.

The next method is used to check whether the game has ended or not. It has the prototype **public boolean isGameOver(String input)**. It takes one parameter which is a string containing either " X " or " O " depending on which of them was played in the last turn. It checks rows, columns, main diagonals and other diagonals for 3 consecutive " X "s or " O "s and returns true if found. It checks rows by making 2 loops. An outer loop that loops all rows and an inner loop that loops all columns. By these loops each row is checked for 3 consecutive characters then the next row is also checked till the final row is reached. Columns are checked by also making 2 loops. An outer loop that loops from column to column and an inner loop that checks each row in the column. For the main diagonals, the outer loop loops rows from the first row till the fourth row as the fourth row is the last possible row to get 3 consecutive symbols in the main diagonal. The inner loop loops columns from the first column to the fifth column. As for the other diagonal, it loops the rows starting from the third row which is the beginning of possible consecutive 3 inputs in the not-main diagonal. Inside the loop, there is an inner loop that loops all columns in each row till the fifth column which is the last possible column to have 3 consecutive diagonal inputs.

The last method is the **public boolean isDraw()** method. It loops around all elements of the "boxes" string, if it finds an empty box(" "), then the game is still going and there can still be a winner. In this case it returns false. If all boxes are occupied, then there is no more possible moves and game is a draw and it returns true.

In the test.java class, the main method is called to run the code. Inside the main method, the java.util.Scanner class is imported as input shall be received from the user and a Scanner object “sc” is created. Then a “TTT” object is created and given the variable name “t”. Then the string “input” is declared and set to “X” as “X” is the first to start the game. Then the boxes inside the “t” object are all initialized to empty strings by calling the method **t.emptyBoxes()** The game then prints a greeting to the players and prints a message that “X” will start the game.

Then the code enters an infinite while loop. Inside the loop t.printBoxes() is called to print the layout of the tic-tac-toe game. Then two messages are printed to the user to ask them for the number of row and the number of column they wish to tick or input. Then an integer is scanned for each of the two numbers under variable names of (row) and (col). Both integers are decremented by 1 as the index of the boxes 2d string starts from 0 and not 1. Then the function **t.playTurn(row, col, input)** is called where the input of the user is placed in the scanned position that they entered. The returned Boolean of the function is stored inside a Boolean variable called “isInputValid”.

After that the **t.isGameOver(input)** function is called as the condition of an if statement to check whether the game has ended by someone winning or not. If true, the boxes are printed and a message stating who the winner is printed and the while loop breaks. Where the winner is the current value of the “input” string which was initialized to “X”.

By the same way, the function **t.isDraw()** is called as the condition of an if statement. If true, a message is printed stating that the game ends in a draw and the while loop breaks.

At the end of the while loop, an if statement checks that if “input” is “X” and that “isInputValid” are both true (It means that the current player was “X” and that he entered a valid input such that he didn’t input in an occupied position because in this case the “input” variable will remain the same for the player to repeat his turn), then the value of “input” changes to “O” such that the loop continues again with the same steps but for the second player. And an else if statement having the same conditions but with “O” instead of “X”.

Conclusion

The two java classes used to create the tic-tac-toe game along with all the methods, constants, variables and loops included were discussed and explained as well as the algorithm used for the implementation of all the mentioned methods.

Group members

Yahia Walid Mohamed Ahmed Zaky ID : 7137 G2 Section 1 Lab 1

Bahaa Eldin Mohamed Gamal Eldin Ahmed Hassan ID : 6726 G2 Section 1 Lab 1