**Final Project – Alex Dorodko – 200454517**

**Overview:** The program has four classes. One handles all of the user input (UserInput.java), one handles everything related to the board (Board.java), the third one handles the two core functions (CoreFunctions.java) and the final one is the main method which executes the application (TicTacToe.java).  
  
The game itself is a game of Tic-tac-toe which includes basic visual graphics, designed for two players. Player X always goes first, so the two players must decide among themselves who that will be. The users will keep marking letters on the board until all of the spaces are taken, or one of the users win. At the end of the game, the user is asked if they would like to play again, if the answer is yes, the game starts up again.

**TicTacToe.java (Main Method)**

/\*\* Application Purpose: This class handles all of the interactions with the user.

\* Author: Alex Dorodko

\* Date: 05/DEC/2020

\* Time: 12:50 PM

\*/

import java.util.\*;

import java.util.Scanner;

public class TicTacToe

{

public static void main(String[] args)

{

//Initiating the userinput and board objects.

UserInput input = new UserInput();

Board board = new Board();

//Starting a try/catch block to catch any exceptions, in addition to the one in the coreCode funtion.

try

{

//Introduction

System.out.println("Hello, welcome to a game of Tic Tac Toe. Here, you can play the game with two players, each taking turns. \nThere will be player X, and player O. Player X goes first, so you can choose between each other who that is.\n");

//Executing the main function which runs the game, and if at the end they would like to play again, the while loop executes again.

do

{

CoreFunctions.coreCode();

input.playAgain();

}

while (input.playAgainValue.equals("Y"));

}

//If the user enters incorrect array index, prevents the program from crashing.

catch (ArrayIndexOutOfBoundsException e)

{

System.out.println("You have entered an invalid number for rows or collumns. Starting the game over.");

CoreFunctions.coreCode();

}

//If the user inputs a char or string instead of an int, it also catches the exception.

catch (InputMismatchException e)

{

System.out.println("You have entered a character instead of a number for row or collumns. Starting the game over.");

CoreFunctions.coreCode();

}

}

}

**UserInput.java (Handles user input)**

/\*\* Application Purpose: This class shandles all of the interactions with the user.

\* Author: Alex Dorodko

\* Date: 06/DEC/2020

\* Time: 12:35 AM

\*/

import java.util.Scanner;

public class UserInput

{

//This is all of my private variables.

private Scanner sc = new Scanner(System.in);

private Board board = new Board();

private int round = 1;

private int newRow;

private int newCollumn;

//Static varible so i can retrieve the data directly in a different class.

public static String playAgainValue = " ";

//Method for asking for the row.

public int rowRq()

{

System.out.println("\nEnter the row where you would like to place the letter: ");

return sc.nextInt();

}

//Method for asking for the collumn.

public int collumnRq()

{

System.out.println("Enter the collumn where you would like to place the letter: ");

return sc.nextInt();

}

//Method for checking if the row value is valid (between 1-3), if not, ask for it again.

public int rowValueCheck(int row)

{

newRow = row;

while (newRow > 3 || 0 >= newRow)

{

System.out.println("This number is not betweeen 1-3. Enter the row where you would like to place the letter: ");

newRow = sc.nextInt();

}

return newRow;

}

//Method for checking if the collumn value is valid (between 1-3), if not, ask for it again.

public int collumnValueCheck(int collumn)

{

newCollumn = collumn;

while (newCollumn > 3 || 0 >= newCollumn)

{

System.out.println("This number is not betweeen 1-3. Enter the collumn where you would like to place the letter: ");

newCollumn = sc.nextInt();

}

return newCollumn;

}

//This is a method to prevent the user from being able to place a letter o nthe board where a letter is already applied.

public void positionCheck(int row,int collumn, String board[][])

{

while (board[row - 1][collumn - 1] != " ")

{

System.out.println("A letter is already placed here, try another location. Enter the row where you would like to place the letter: ");

row = sc.nextInt();

System.out.println("Enter the collumn where you would like to place the letter: ");

collumn = sc.nextInt();

}

}

//This is a method to ask the user if he/she would like to play again.

public void playAgain()

{

System.out.println("\nWould you like to play again? (Y/N)");

playAgainValue = sc.nextLine().toUpperCase();

while (playAgainValue.equals("Y") != true && playAgainValue.equals("N") != true)

{

System.out.println("Invalid Answer. Would you like to play again? (Y/N)");

playAgainValue = sc.nextLine().toUpperCase();

}

if (playAgainValue.equals("Y"))

{

round += 1;

System.out.println("----- Round " + round +" -----");

}

else if (playAgainValue.equals("N"))

{

System.out.println("\nHope you had a good time, goodbye!");

}

}

}

**CoreFunctions.java**

/\*\* Application Purpose: This class stores the core functions of the TicTacToe

\* Author: Alex Dorodko

\* Date: 05/DEC/2020

\* Time: 03:00 PM

\*/

import java.util.\*;

public class CoreFunctions

{

//This checks if the provided user won the game, or if its a tie.

public static boolean winCheck(String board[][], boolean win, String ltr)

{

//bottom horizontal 1

if (board[2][0] == ltr && board[2][1] == ltr && board[2][2] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//middle horizontal 2

else if (board[1][0] == ltr && board[1][1] == ltr && board[1][2] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//top horizontal 3

else if (board[0][0] == ltr && board[0][1] == ltr && board[0][2] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//vertical left 4

else if (board[0][0] == ltr && board[1][0] == ltr && board[2][0] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//vertical middle 5

else if (board[0][1] == ltr && board[1][1] == ltr && board[2][1] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//vertical right 6

else if (board[0][2] == ltr && board[1][2] == ltr && board[2][2] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//top left to bottom right 7

else if (board[0][0] == ltr && board[1][1] == ltr && board[2][2] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//top right to bottom left 8

else if (board[0][2] == ltr && board[1][1] == ltr && board[2][0] == ltr)

{

System.out.println("Player " + ltr + " won!");

win = true;

}

//Checking if all the slots are filled up, and if there is no winner, it is a tie.

int counter = 0;

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

if (board[i][j] != " ")

{

counter += 1;

}

}

}

if (counter == 9)

{

System.out.println("Draw!");

win = true;

}

return win;

}

public static void coreCode()

{

int row;

int collumn;

boolean win = false;

Board board = new Board();

UserInput input = new UserInput();

try

{

while (win == false)

{

//Outputting the board

board.getBoard();

//Stating the current player's turn.

System.out.println("Current Turn: Player X");

//requesting the coordinates where to put the letter

//Asking for the row, and making sure the value is valid.

row = input.rowRq();

row = input.rowValueCheck(row);

collumn = input.collumnRq();

collumn = input.collumnValueCheck(collumn);

//Checking if a letter is already positioned there

input.positionCheck(row, collumn, board.getBoardValues());

//Setting the new letter in the array of values

board.setLetter(row, collumn, "X");

//Outputting the board

board.getBoard();

//Checking if the person won

win = CoreFunctions.winCheck(board.getBoardValues(), win, "X");

if (win == true)

{

break;

}

System.out.println("Current Turn: Player O");

//requesting the coordinates where to put the letter

//Asking for the row, and making sure the value is valid.

row = input.rowRq();

row = input.rowValueCheck(row);

collumn = input.collumnRq();

collumn = input.collumnValueCheck(collumn);

//Checking if a letter is already positioned there

input.positionCheck(row, collumn, board.getBoardValues());

//Setting the new letter in the array of values

board.setLetter(row, collumn, "O");

//Outputting the board

board.getBoard();

//Checking if a person won. If one did, end the loop.

win = CoreFunctions.winCheck(board.getBoardValues(), win, "O");

if (win == true)

{

break;

}

}

}

//If the user enters incorrect array index, prevents the program from crashing.

catch (ArrayIndexOutOfBoundsException e)

{

System.out.println("You have entered an invalid number for rows or collumns. Starting the game over.");

CoreFunctions.coreCode();

}

//If the user inputs a char or string instead of an int, it also catches the exception.

catch (InputMismatchException e)

{

System.out.println("You have entered a character instead of a number for row or collumns. Starting the game over.");

CoreFunctions.coreCode();

}

}

}

**Board.java**

/\*\* Application Purpose: This class stores all varibales and methods related to the TicTacToe board.

\* Author: Alex Dorodko

\* Date: 06/DEC/2020

\* Time: 10:49 PM

\*/

import java.util.Scanner;

public class Board

{

//Creating the board array

private String[][] board = { {" ", " ", " "},

{" ", " ", " "},

{" ", " ", " "} };

//This is the method to retrieve the board, formatted with values.

public void getBoard()

{

System.out.println(" 1 2 3" +

"\n1 " + board[0][0] + " | " + board[0][1] + " | " + board[0][2] + " " +

"\n -----|-----|-----" +

"\n2 " + board[1][0] + " | " + board[1][1] + " | " + board[1][2] + " " +

"\n -----|-----|-----" +

"\n3 " + board[2][0] + " | " + board[2][1] + " | " + board[2][2] + " \n");

}

//This is a method to set a value of the board.

public void setLetter(int row, int collumn, String ltr)

{

this.board[row - 1][collumn - 1] = ltr;

}

//This is the method to get the board values raw, without formatting.

public String[][] getBoardValues()

{

return board;

}

}

------------------------------------------------------------------------------------------------------------

***Declaration of understanding***

*I understand that plagiarised answers will receive a grade of zero and an academic misconduct.*

*I understand that an academic misconduct will result in an automatic zero in the assignment in the event that this is my first offense and an automatic grade of zero in the course if this is my second offense.*

*I understand that all answer material, must be original material.*

*I understand that plagiarism means to present someone else’s material as my own.*

*I understand that to copy material from the internet, text books, course slides, or any other source, is a form of plagiarism.*

*I understand that to copy material from the internet, text books, course slides or any other source, and then to change a few variable names etc. in that material is a form of plagiarism.*

*I understand that to copy material from the internet, text books, course slides or any other source, and then to slightly modify that material is a form of plagiarism.*

*I understand that the appropriate way to complete this assignment is to write and test all of the code myself while referencing other sources for general guidance.*

*I have read and clearly understand each one of these statements, and accept the responsibility and penalty for any actions that I take which may contravene any one of these statements.*

*Name (type)* ***Alex Dorodko***

*Student Number:* ***200454517***

*Signature: (draw with mouse or tablet pen): *

------------------------------------------------------------------------------------------------------------