Assignment Due Date and Time

17 November 2024 (Sun) 11:55 p.m.

Instructions to Students

1. This assignment is weighted 40% of the overall End of Assessment of this module.
2. This assignment is an individual assignment and should be done by you only. **Plagiarism will be treated seriously**. Any submitted assignment is found that involved wholly or partly in plagiarism (no matter the assignments are from the original authors or from the plagiarists) **will be scored Zero mark** and the students involved will be received discipline penalty set by the institute accordingly.
3. Only Java programming language is allowed to develop any required program.
4. Your programs must be well structured. A comment should be included in each class and method to state its main function. Explanation of each variable is also required. The source code must be properly indented. The first few lines in a source file must be a comment stating the name of the source file, your name, class, student number as well as the description of the purposes of the program. Marks will be deducted if any of the above mentioned comment is not included.
5. Grading of your programs will be based on correctness, quality, style, efficiency and the advanced features.
6. You are required to hand in a softcopy of the program source codes uploaded to Moodle.
7. Remember to backup all your programs.
8. Late submission will **NOT** be accepted.

Assignment Specification

You are asked to write a Gomoku game in Java.

Information on the Gomoko Game

Gomoku, also called **Five in a Row**, is an abstract strategy board game. It is traditionally played with Go pieces (black and white stones) on a 15×15 Go board. Because pieces are typically not moved or removed from the board, gomoku may also be played as a paper-and-pencil game. The game is known in several countries under different names.

Rules

Players alternate turns placing a stone of their color on an empty intersection. Black plays first. The winner is the first player to form an unbroken line of five stones of their color horizontally, vertically, or diagonally. If the board is completely filled and no one has made a line of 5 stones, then the game ends in a draw.
Gomoku is a very popular game and it is available on many online website. To understand the game rules you can play it on:

https://www.mathsisfun.com/games/gomoku.html

Requirements of the Assignment

The Gomoku game will be run in console mode. For simplicity, we use the letter ‘1’ and '2' to represent the pieces of player1 (black) and player2 (white) respectively, and the letter '0' is used to represent the empty space of the grid.

Furthermore, to make game different to the normal one, the board size is changed to **10X10** and the **Five in a Row change to Four in a Row**.

The program then asks player1 to input a pair of integers (row and column indices). The program will redraw the grid on the console with the new piece and the changes.

```
0 | 0 0 0 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 1's turn.
Enter row and column (e.g., 0 1): 1 3
```

User inputs 1 and 3

```
0 | 0 0 0 0 0 0 0 0 0 0
1 | 0 0 0 1 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 2's turn.
Enter row and column (e.g., 0 1):
```

Then it will ask player2 to input another cell position to place its piece. After that, the program will show the updated grid on the console. It then asks player1 to input a position again. The process will be continued until the first player to form an unbroken line of **Four** stones of their color horizontally, vertically, or diagonally. If the board is completely filled and no one has made a line of 5 stones, then the game ends in a draw.

When the game ends, the program will show the result and who is the winner or a draw game.

```
Player 2's turn.
Enter row and column (e.g., 0 1): 0 3
0 | 0 1 2 2 0 0 0 0 0 0
1 | 0 1 2 0 0 0 0 0 0 0
2 | 0 2 2 1 0 0 0 0 0 0
3 | 2 1 1 1 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
```

```
Player 2 wins!
```

Your programme should also require to handle following errors:

```
0 | 0 0 0 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 1's turn.
Enter row and column (e.g., 0 1): -1 0
Out of range! Input again.
Player 1's turn.
Enter row and column (e.g., 0 1): 0 10
Out of range! Input again.
Player 1's turn.
Enter row and column (e.g., 0 1): 0 1
0 | 0 1 0 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 2's turn.
Enter row and column (e.g., 0 1): 0 1
Invalid move. Try again.
Player 2's turn.
Enter row and column (e.g., 0 1):
```

Testing

You can ease the testing by using '**Copy and Paste**' rather than inputting data manually. Prepare a text file, which includes all user inputs in a game run. The following input is an example.

By using Copy and Paste, you can automatically input data in the command prompt window and then get the result automatically. **Note:** The input data will not be echoed.

Test Case

```
-1 0
0 10
0 1
0 1
0 2
1 1
1 2
2 3
2 2
3 1
2 1
3 2
3 0
3 3
0 3
```

Expected Output

```
0 | 0 0 0 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 1's turn.
Enter row and column (e.g., 0 1): -1 0
0 10
0 1
0 1
0 2
1 1
1 2
2 3
2 2
3 1
2 1
3 2
Out of range! Try again.
Player 1's turn.
Enter row and column (e.g., 0 1): Out of range! Try again.
Player 1's turn.
Enter row and column (e.g., 0 1):
0 | 0 1 0 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 2's turn.
Enter row and column (e.g., 0 1): Invalid move. Try again.
Player 2's turn.
Enter row and column (e.g., 0 1):
0 | 0 1 2 0 0 0 0 0 0 0
1 | 0 0 0 0 0 0 0 0 0 0
2 | 0 0 0 0 0 0 0 0 0 0
3 | 0 0 0 0 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
....... (SKIP the OUTPUT) .......
Player 1's turn.
Enter row and column (e.g., 0 1):
0 | 0 1 2 0 0 0 0 0 0 0
1 | 0 1 2 0 0 0 0 0 0 0
```

```
2 | 0 2 2 1 0 0 0 0 0 0
3 | 2 1 1 1 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 2's turn.
Enter row and column (e.g., 0 1):
0 | 0 1 2 2 0 0 0 0 0 0
1 | 0 1 2 0 0 0 0 0 0 0
2 | 0 2 2 1 0 0 0 0 0 0
3 | 2 1 1 1 0 0 0 0 0 0
4 | 0 0 0 0 0 0 0 0 0 0
5 | 0 0 0 0 0 0 0 0 0 0
6 | 0 0 0 0 0 0 0 0 0 0
7 | 0 0 0 0 0 0 0 0 0 0
8 | 0 0 0 0 0 0 0 0 0 0
9 | 0 0 0 0 0 0 0 0 0 0
  +--------------------
    0 1 2 3 4 5 6 7 8 9
Player 2 wins!
```

Example codes for Scanner usage

Following is a **WRONG** example program to use a Scanner to do the input:

```
// create new Scanner objects in loop
do {
      Scanner sc = new Scanner( System.in);
      choice = sc.nextInt();
} while (choice != 1);
```

Following is another **WRONG** example program to use a Scanner to do the input:

```
import java.util.Scanner;
public class WrongTest {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int x;
    System.out.print("Enter x:");
    x = sc.nextInt();
  }

  public static void method2() {
    Scanner sc = new Scanner(System.in); //second Scanner objects
    int y;
    System.out.print("Enter y:");
    y = sc.nextInt();
  }
}
```

You are NOT allowed to use GUI such as JOptionPane in your program.
Following is an example program to use a Global Scanner to do the input:

```
import java.util.Scanner;

public class Test {
  //Global declaration for Scanner
  public static Scanner sc = new Scanner(System.in);

  public static void main(String args[]) {
    int x;
    System.out.print("Enter x:");
    x = sc.nextInt();
  }

  public static void method2() {
    int y;
    System.out.print("Enter y:");
    y = sc.nextInt();
  }

}
```

Marking Scheme

Functions

Display the game board (Initial 5%, Subsequent 5%)	10%
Players play alternatively	10%
Correctly place a piece	10%
Error Checking	
Range	5%
Invalid move	5%
Game finishes appropriately	20%
Game show Winner/Draw	20%
Program structure and using appropriate methods.	10%
Style	5%
Documentation (Comments)	5%

Deduction

Cannot pass the Java compiler	-100
Do not follow the requirements in "***Requirements of the Assignment***"	-30
Cannot do the Test Case in the "***Testing***"	-30
More than one Scanner objects in your codes	-20
Do not use Scanner as the one and only one input method in your codes	-50
Cannot perform a demonstration to your lecturer	-50