EEE 210: Sofware Engineering (Spring 2019)
Final Project Description

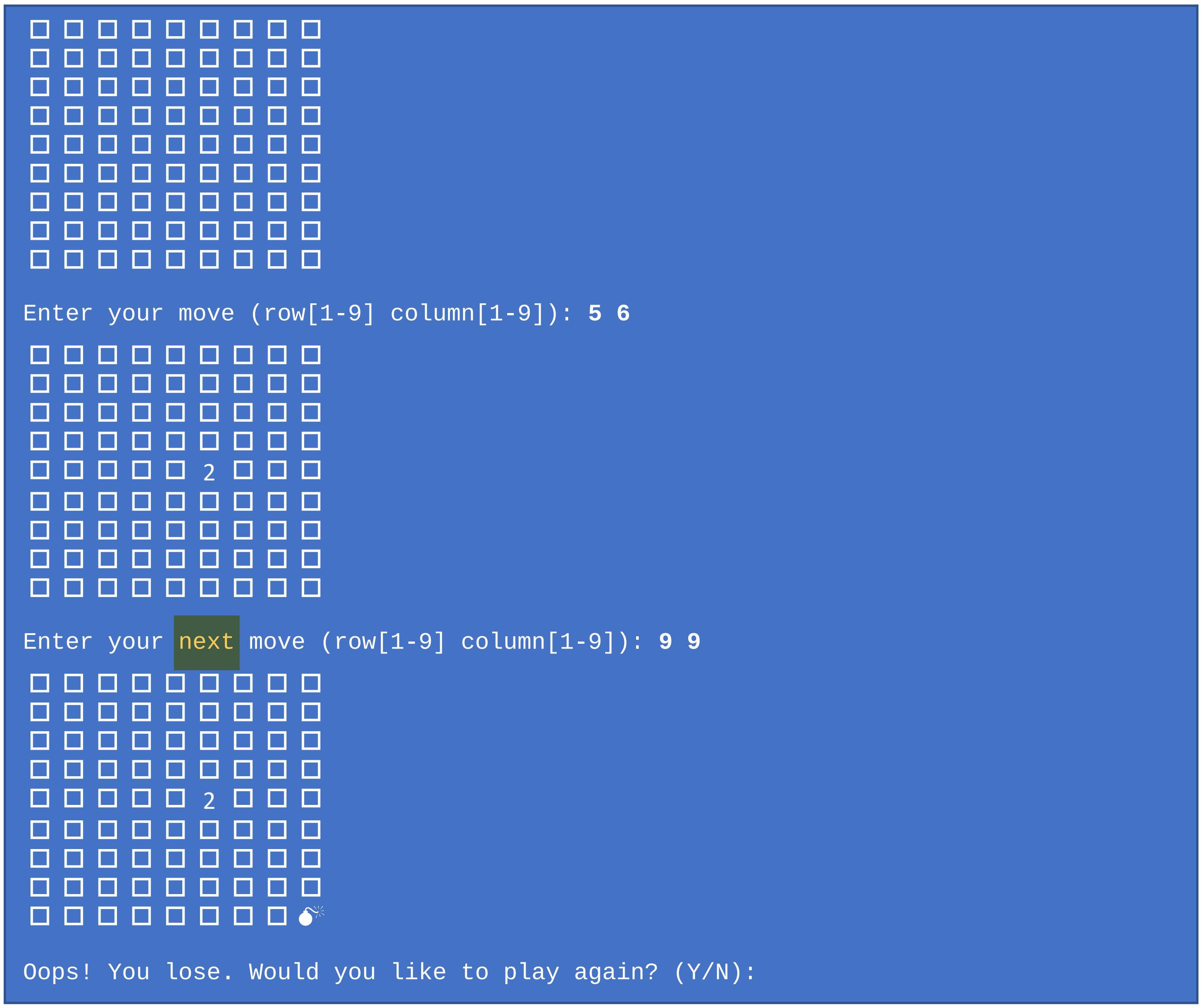
Title: Design and Development of a Multi-Player Minesweeper Game

Online submission deadline: 21st April (Sunday) before mid-night

The main objective of this programming project is to use OOP in designing and implementing a simple two-player game using JavaFX or Swing. To understand the rules of the game, you can wiki “Mine Sweeper”. The total project grade is distributed over multiple programming levels defined below. To obtain the grade for each level, you should implement the task given in that level. Groups consisting of **at most 2** students are required to work only on all Levels A, B, C and D.

Level A: Non-graphical Non-OOP Single Player Minesweeper:

Implement a non-OOP game for a single player in a 9 × 9 grid game using the console of the IDE (without any graphical features). As shown in the example below, the player begins by placing an X in position (5,6) of the grid and pressing enter. It is then followed by the revealing the flipped element at (5,6). If successful, then the player is asked to enter the next element co-ordinate. Otherwise, the player loses and is asked if he/she desires to play a new game or not.



```
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □

Enter your move (row[1-9] column[1-9]): 5 6
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ 2 □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □

Enter your next move (row[1-9] column[1-9]): 9 9
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ 2 □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ 💣

Oops! You lose. Would you like to play again? (Y/N):
```

If the player attempts to place a sign in a non-empty or undefined position, the program should prompt the player with an error message as shown in the following example:

```
Enter your next move (row[1-9] column[1-9]): 9 11
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ 2 □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □

Invalid move. Try again:
```

When the player wins the game, the following message should be displayed on the console:

```
You win. Congrats!
```

Level B: Graphical OOP Single Player Minesweeper:

Convert the earlier non-OOP version of the game to a more complex object-oriented. You need to implement two classes namely, Position and Grid. The class Position is used to define the element position on the grid and the class Grid is used for designing the 9 × 9 game board. Finally, write a main class called MineSweeperMain to pull all the pieces together. The player should be able to interact with the game by clicking on the element to be flipped using the mouse.

Level C: Graphical OOP Two-Player Minesweeper:

In this level, you need to extend the game designed in Level B to include two players playing the game in turns. When the game is executed, the GUI should ask each player his/her name. The game should then display the names of the two players (assume Jack and Jill) along with their accumulated score. The score of a player is defined as the number of empty elements he/she flips by clicking on an element. Once Jack clicks an element, a message at the bottom of the GUI should read "It is Jill's turn.". The game will come to an end only if (i) either one of the players clicks on the last safe element or (ii) either one clicks on the element containing the bomb symbol.

Level D: Graphical OOP Two-Player Minesweeper with Arbitrary Grid Dimensions:

As the final step, you are required to prompt the user for creating the game grid based on the input. For instance, if the user enters 6 8, you should then create a game with dimensions (6,8). After doing so, your game should function as implemented in Level C.

Please Note:

1. Your report should be prepared using the sample word format provided on Moodle.
2. You should include only ONE detailed UML class diagram for the complete project. The details shown in the UML diagram should actually be implemented in your code and it will be checked.
3. Any queries regarding the project should be clarified with the instructor before Week 12.