Homework 07 – Minesweeper

Authors: Vince, Ian, Connor, Nicolas

Purpose

For Homework 07, you will be creating the Graphical User Interface (GUI) for the game Minesweeper (https://en.wikipedia.org/wiki/Minesweeper_(video_game)) using your **JavaFX and Event-Driven Programming** skills. We will provide you the program containing the logic of the game. You will be responsible for creating the interface for the user to play minesweeper.

Solution Description

We have provided `MinesweeperGame.java` which allows you to start and play a game of minesweeper through method calls. In addition, we have provided you with `Tile.java` and `Difficulty.java` which are used by `MinesweeperGame.java`. Check out the Homework 7 Appendix file for helpful details about these provided classes.

You will write `MinesweeperView.java`, which will display the board and allow users to use clicks to play the game, as well as show a start and end screen. You will also have to adhere to proper Checkstyle including Javadocs.

`MinesweeperView.java`

This file contains the `MinesweeperView` which will be the base for your JavaFX application. It will display the start screen where a user can start a game, the board where the user can play the game, and an end screen that displays the final result of the game. Please **do not create multiple stages** to display the different screens (this means only one window should be displayed throughout the entire game play, not including alerts). Your window must be reasonably sized. Your application must have the following properties:

- The title of the window must be "`Minesweeper`"
- The user, when first opening the application, should be greeted by the start screen. This should contain the following elements:
    - A clearly labelled dropdown menu to select the difficulty: Easy, Medium, or Hard
    - A text input for the user to input their name. The text input must be clearly designated for the name.
    - A clearly labelled start button. When this button is clicked:
        - If a value of the dropdown is selected AND a name is entered, proceed to the game screen.
        - Otherwise, inform the user that both fields are required and remain on the start screen.
        - You should use a **lambda expression** to accomplish this functionality.
- Once the user enters the game screen, create a new `MinesweeperGame` object. The game screen should have the following properties:
    - The `MinesweeperGame` constructor should be called, and the difficulty selected on the start screen should be passed in. A `Difficulty.java` file containing an enum of the different difficulties has been provided.
    - A 15x15 grid of buttons that represent the `Tiles` in the `MinesweeperGame` should be displayed. Each should be empty at first.
    - When a user clicks any of the tile buttons do the following. This handler should be implemented with an **anonymous inner class**:
        - Call the `check(int y, int x)` method of the `MinesweeperGame` that takes in the x and y index on the board (not the pixels) – note the top left corner of the board is 0, 0.

        - ◊ `check()` will return an array of `Tile` objects that the move revealed.
            - ♦ If the `Tile` clicked contained a mine, the array will be of length 1 and the `Tile` 's `isMine()` method will return true.
            - ♦ Otherwise, each `Tile` object in the returned array will contain the x and y values of a `Tile` to reveal, and the number of surrounding mines for that `Tile`. These fields can be accessed using the `getX()`, `getY()`, and `getBorderingMines()` methods respectively. This represents, in the game, when the `Tile` clicked on is not a mine and there are a number of cells that can be revealed (not just the cell that was clicked).
            - ♦ **Note:** The returned array will always contain the `Tile` at the location that was clicked – it will be the only `Tile` in the array when the clicked location is a mine and one of the `Tile` objects in the array when the clicked location is not a mine.
    - ▪ If the clicked location contained a mine, the game should end. A new screen should be displayed that reads "`You Lost, [Name of User]`". This screen should also contain a clearly labeled New Game button returning them to the **Start screen** of the application. Use a **named inner class** to achieve the new game button functionality.
    - ▪ If the clicked location was not a mine, you should reveal each location on the grid that check returned, these should now display an integer value representing the number of surrounding mines.
    - ▪ Check if the user won the game, with `MinesweeperGame`'s `isWon()` method. If the user won, a new screen should be displayed that reads "`Congratulations, [Name of User]`". This screen should also contain a clearly labeled New Game button returning them to the start screen of the application. Use a **named inner class** to achieve the new game button functionality. This inner class should be the *same* one used when the game is lost.

- **[Optional]** Extra credit will be awarded for the following features:
    - o Make a non-trivial addition to the program's graphics that clearly extends its look (e.g. Make the visible tiles appear visually different from the tiles that have yet to be revealed or add animations to the ending screens).
    - o Make a non-trivial addition to the program's controls or functionality that clearly extends its capabilities (e.g. Add a timer to the game or allow the user to add flags to the tiles they believe to be mines).
    - o You can choose to try for one, both, or neither of the above extra credit options. If you choose to go for the extra credit, attach a file named `extra_credit.txt` to your submission that explains exactly what additional features you have added.
    - o **Note:** Extra credit additions should NOT affect any of the required functionality!

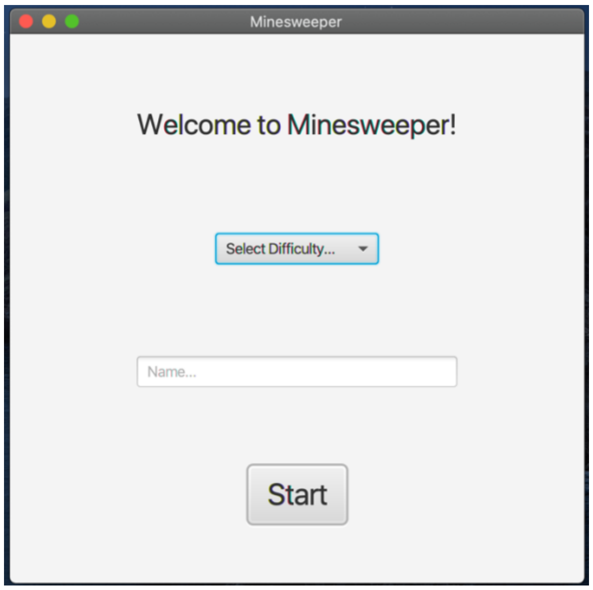
You will find it useful to create helper methods to help you structure the logic of your program. Feel free to create as many helper methods as you would like but this is one way you can tackle the problem:

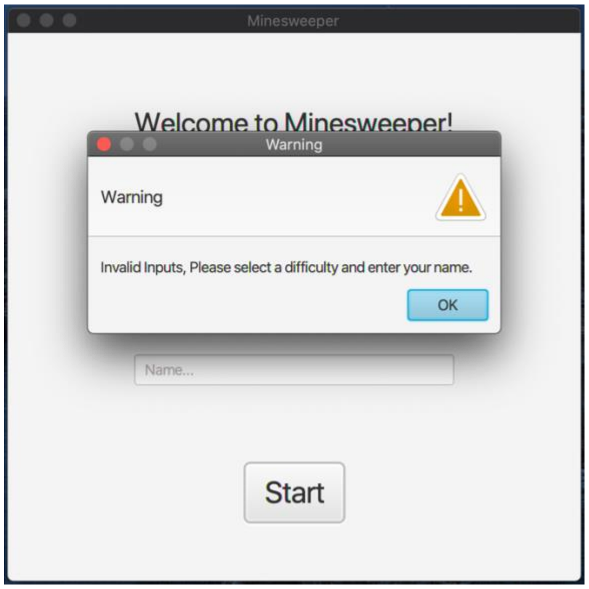
- o `start(Stage primaryStage)`
    - ▪ Overrides the `start()` method from `Application`.
    - ▪ Creates a `Scene` for start screen and shows it on the `Stage`.
    - ▪ The start button's lambda expression calls `playMinesweeper()` with the appropriate parameters if a name and difficulty are properly entered.

- `playMinesweeper(Stage primaryStage, Difficulty gameDifficulty, String name)`
  - Creates another `Scene` where the user can play minesweeper and shows it on the passed in `primaryStage`.
  - You should construct a `MinesweeperGame` here with the passed in `gameDifficulty`.
  - Create a grid of buttons (each of which represents a `Tile`), loop through each one, and use an anonymous inner class to set the handler for when it is clicked.
  - When the user wins or loses the game, create a new `Scene` to display the appropriate end screen using a private inner class and show it on the passed in `primaryStage`.
    - ◊ **Note:** The private inner class that starts a new game cannot directly access the parameters of the `playMinesweeper()` method--you might find it helpful to write a constructor.

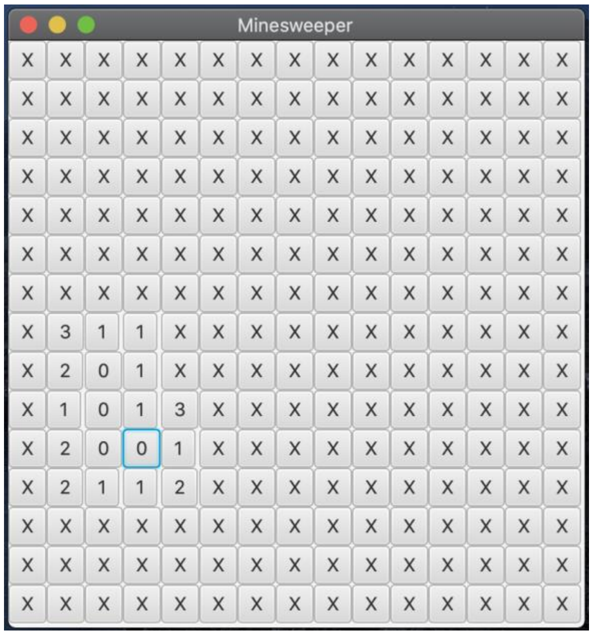
Clarifications and Example Output

Please refer to the pinned Piazza thread for HW07 Clarifications for any necessary clarifications about the requirements outlined in this pdf. Some examples of the different screens of the game are shown below:

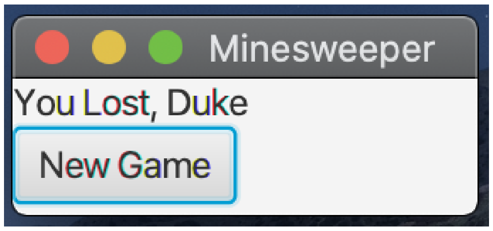




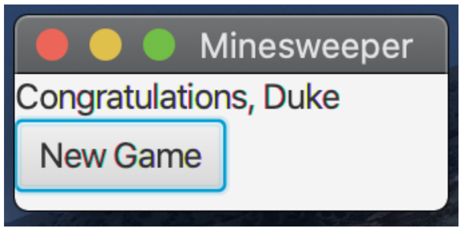
Invalid Inputs, Please select a difficulty and enter your name.



X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	3	1	1	X	X	X	X	X	X	X	X	X	X	X
X	2	0	1	X	X	X	X	X	X	X	X	X	X	X
X	1	0	1	3	X	X	X	X	X	X	X	X	X	X
X	2	0	0	1	X	X	X	X	X	X	X	X	X	X
X	2	1	1	2	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X



You Lost, Duke



Congratulations, Duke

Allowed Imports:

You are allowed to import anything from the javafx library for this assignment.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`
- `Scene Builder`

Checkstyle and Javadocs

You will need to write Javadocs for this assignment and all following homeworks as well as run Checkstyle for Javadocs.

You must run Checkstyle on your submission. The Checkstyle cap for this assignment is **15 points**. If you don't have Checkstyle yet, download it from Canvas -> Modules -> Checkstyle Resources (all sections) Place it in the same folder as the files you want Checkstyled. Run checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off (limited by the Checkstyle cap mentioned above). The Java source files we provide contain no Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Additionally, you must Javadoc your code.

Run the following to only check your Javadocs:

```
$ java -jar checkstyle-8.28.jar -j yourFileName.java
```

Run the following to check both Javadocs and Checkstyle:

```
$ java -jar checkstyle-8.28.jar -a yourFileName.java
```

For additional help with Checkstyle see the CS 1331 Style Guide.

Collaboration

No collaboration is allowed on this assignment. See syllabus for more details.

In addition, note that it is not allowed to upload your code to any sort of public repository or website. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- MinesweeperView.java
- extra_credit.txt (if you choose to try for the extra credit)
- Any image files you may want to use for the extra credit

Make sure you see the message stating "HW07 submitted successfully". From this point, Gradescope will run basic compilation tests on your submission. Except for compilation tests, this assignment will be ***entirely manually graded*** due to the fact that it is a GUI program.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Run Checkstyle on your code to avoid losing points
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Feature Restrictions" to avoid losing points
- Check on Piazza for a note containing all official clarifications