CS 2410 – Fall 2016
Assignment #8 - Minesweeperish

Introduction

You will be creating a Minesweeper-like game called Minesweeprish. If you don't know how this game works then just google it up and play a few rounds. Specific implementation details for our version are below.

This assignment builds on much of your knowledge gained throughout the semester. There are many approaches to building the program, and you should use your best judgement on how to create it. That being said, here are a few concepts to keep in mind as you build your program:

- Property Binding
- CSS
- Inheritance
- Recursion
- Concurrency (Thread/Timer)
- Lambda Expressions
- Method References

General requirements

In order to receive full credit your code must follow appropriate class conventions

Gameboard

1. (15 pts) There should be a score area at the top that displays the time elapsed, number of bombs left, and a start button. The main area is a grid of cells. You can create just a single mode of 20x20 cells and 100 bombs (That's 25% of the grid).
2. (5 pts) The game window is not resizeable

Gameplay

Minesweeperish is a game of logic. (If you haven't played Minesweeper, then Google it up and play for a bit. It'll be easy to understand.) Players use information about a given cell to determine which cells have bombs, and which do not. The player must clear all cells that do not have bombs in order to win the game. The game works as follows:

1. Player selects a cell (left mouse click) from the game board grid.
    a. (5 pts) If it is the first cell selected then the timer starts.
    b. (15 pts) The timer works properly and doesn't interfere with gameplay
    c. (15 pts) If the cell does not contain a bomb, then it displays the number of neighbors (including diagonally) that contain a bomb.
    d. (15 pts) If no neighboring cells contain a bomb, then each neighboring cell is cleared as if they had each been selected by the user.
    e. (10 pts) If the cell contains a bomb the game is over.
2. (15 pts) A player may right click on a non-cleared cell to mark it.
    a. Cycle through the following markings in this order
        i. Bomb (mark with flag image)
        ii. Possible Bomb (mark with question image)
        iii. No marking
    b. Cells marked as bombs or possible bombs may not be selected (with left click)
3. (10 pts) Player continues to select cells until all cells are cleared that do not have bombs
4. Game over
    a. (15 pts) The timer is stopped when the game is over
    b. (20 pts) If player successfully clears board, then all bomb locations are revealed with an image of a bomb in each cell with a green background.

c. (15 pts) If the player clicks a bomb, then all bomb locations are revealed with an image of a bomb in each cell.
    i. Successfully marked bomb cells have a green background
    ii. Unmarked cells with a bomb have a red background
    iii. Marked cells without a bomb have a yellow background
    iv. All other cells are left unchanged
d. (10 pts) Provide appropriate message when game is over
    i. Use timer value in message if game is completed successfully
e. (10 pts) User should be able to click a reset button and play again
    i. Everything should reset properly
    ii. Unusual behavior in gameplay after the first game will result in lost points.

More Requirements

In addition to meeting all of the gameplay and gameboard requirements, you must adhere to the following:

1. (10 pts) packages & classes
    a. one for view with AT LEAST the following
        i. one class for scoreboard
        ii. one class for cell (each button is a cell)
            1. inherit Button (or some other Node of your choosing) in this class
    b. package for controller
        i. at least one class to control gameplay
    c. Note: Just do your best at separating code properly. It will force you to think about your code more, which will streamline things and probably save you a good amount of time.
2. (15 pts) Use CSS to control how the cells (Buttons) look at different stages

Did you know? (If not, don't worry. We'll be discussing)

- An ArrayList is part of the Java Collections Framework. That means you can use it with the following static method: Collections.shuffle(<Collection>)
- An Iterator is also part of the Java Collections Framework. You can create an Iterator object with the iterator() method. Iterators can kind of be used like Scanner object with hasNext() and next() methods.

What/How To Turn In (READ THIS)

Submit your files on Canvas according to class conventions

Due: December 9, 2016