Minesweeper Read Me

Implementation:
This minesweeper is represented as a 2D grid of MinsweeperCells. This way the MinsweeperCell fields can hold all important information to the logic of Minesweeper such as: isBomb, isFlagged, number of adjacent bombs, etc. The MinesweeperCell field can be changed as the game is played. The model also holds a list of high scores for the standard minesweeper size grid. And this is updated and moved, but will only keep track of the lowest 10 completion times.

Wow Factor:
In normal Minesweeper, it is possible to have to guess when selection safe tile vs bomb time even with clues. We have implemented a solver to ensure that this case never happens. While the bombs are being placed, run an internal "solver" to ensure bomb placement will not cause a situation where you would have to guess.

Starting the game:
When starting up Minesweeper for the first time. It will look for a file to load(saved_game.dat).
- if no file found, then it will start game at standard size and standard bomb count
    - standard size and bomb count is 10.
- if file found, it will use the file to construct the model, with information from previous save including: previous flagged cells, revealed cells, time and high score array.

Playing the game:
1. Once the game is started it is guaranteed that first right click is never a bomb.
    - since bombs are not put into the board until after the first click.
    - after first right click is made bombs are randomly generated into the model.
        - this is done by generating a random int value for rows and cols in the range of 0 and the number for rows and cols for the model.
            - if the two values generated are same location as first click it will find new random values and try again.

2. All subsequent right clicks are making moves.
    a. If location is a bomb, game is over and all bombs are revealed.
        - alerted that game is over and you lost, and displays high scores for standard game size.

    b. If location is adjacent to a bomb, the number of adjacent bombs are revealed
        - bomb adjacency is in 8 directions from the cell (NW,N,NE,E,SE,S,SW,W)

    c. If location has no adjacent bombs, location is revealed and all contagious zero adjacent cells are revealed.

    d. If location is last unrevealed, non-bomb cell, then game is over, and game is won.
        - alerted that game is over and you won, and displayed high scores for standard game size.

3. If left click is made, it marks a cell as flagged.
    - flagged cells are potential bombs.
    - flagged cells can not be revealed, with a right click event.
        - this is to prevent the player from accidentally ending the game if they believe that the cell is a bomb.

- if cell is flagged and left click is made again it removes the flagged status.

4. If while playing you close the window the game is then saved including: time so far, the flagged cells, and the revealed and unrevealed cells.
    - this creates the saved_game.dat file that is called to load when the game is first started.
    - if file already exists, it is deleted and a new file with the new information replaces it.
    - if the game is over when the window is closed, it will not save the game state.

5. There is a drop down menu where players can start a new game of varying dimensions.
    a. If standard option 1 is selected, the model is reset with standard size and standard bomb counts.

    b. If 15X15 game option 2 is selected, model is reset to a grid of 15x15 with 15 bombs.

    c. If 20x20 game option 3 is selected, model is reset to a grid of 20x20 with bombs.

    d. If custom game option 4 is selected. A series of TextInputDialog windows will pop up and prompt player for integer values. The TextInputDialog will continue to prompt a user for int value of custom grid.
        - it will not take on numeric input.
        - it will not allow for bomb count entry to be over NxM.