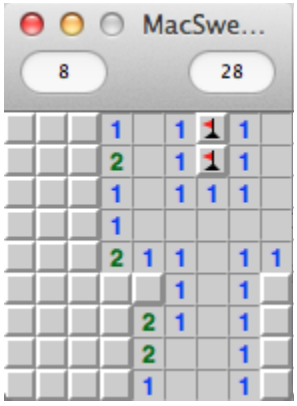
CS1331 HW10A

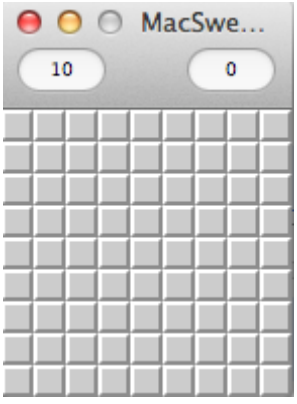
Minesweeper

Due Friday November 30th

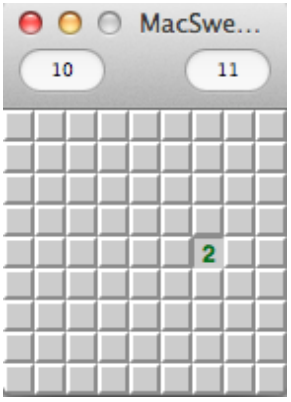


In this homework you will be recreating the classic game, minesweeper. The rules are simple:

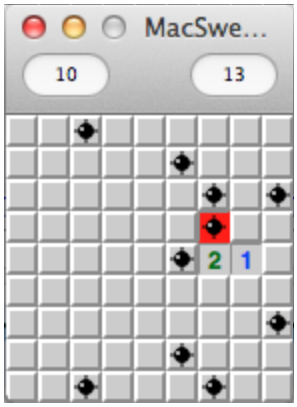
Start with a blank field. All the tiles are hidden, and some of them contain mines.



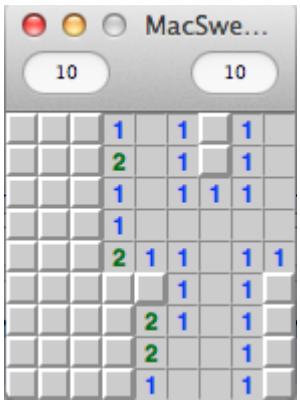
Left- click somewhere on the field. Whenever you left-click on a tile and there is no mine there, a number is revealed that specifies the number of mines in the surrounding 8 tiles.



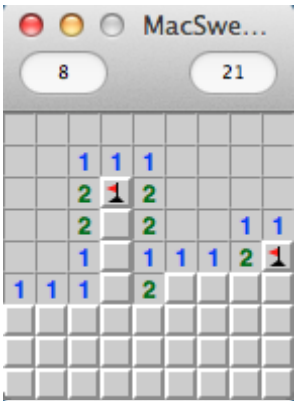
You have to be careful, though. If you click on a tile where there is a mine, you lose the game! Upon losing, all the mines are revealed.



If you left-click on a tile and there are no surrounding mines, then the entire mine-free area it is connected to is revealed. For example, in the image below, I clicked in the top-right corner. This behavior is a bit tricky to code, so we've provided the code for you.



The user can make educated guesses as to where mine could be. They can do this by right-clicking on a tile they think a mine is located. This places a flag on that tile. Flags don't have any effect on the game, they just serve to help the player remember where mines are. As you can see in the image below, I've placed flags where I **know** a mine is (due to the "1" tiles beside it).



The game is won when every non-mine tile is revealed.

Requirements

You will be implementing minesweeper in Java and Swing, as per the requirements above. You are free to implement the code any way you like, given that your design follows object oriented design principles and is well organized. Additional requirements:

1. Whenever a new game is started, you need to ask the user for the number of x/y tiles, and the number of mines to place on the field.

2. Do not place the mines until after the user has clicked once. This prevents the first click from being on a mine.
3. You will need to have a button or menu option to start a new game.
4. You must have a “hint” button or menu option that tells the user where a single, un-flagged mine is.
5. You may not use command-line to take in input from the user or tell the user information.
6. The user must be notified when they win or lose, and then given the option to play a new game.

You need to include a README with your submission that tells the TA how to run your assignment.

Provided Code

We have provided for you utility code to find connected regions of tiles (see the section above). You can write your own if you like. Assuming you have written a **Tile** class that implements the provided interface **MineTile.java**: the method **expandTile** takes in a single **Tile** to expand from, and then a board of **Tile**s to search through. The method returns an **ArrayList** of **Tile**s that are all part of one connected region that you should then open.

```
ArrayList<Tile> tilesToOpen = MinesweeperUtils.expandTile(tile, board);
```

It’s very important that you format the call this way, as the method makes use of something called ***generics***, a topic which isn’t covered in this course. This means that the method will work perfectly as long as your **Tile** implements **MineTile.java**.

Additional Info

If you decide to use JButtons for tiles, you will need to use:

```
setMargin(new Insets(0, 0, 0, 0));
```

On the button to make the text/image fully visible.

Extra Credit

- (+10) High Score Table is saved/loaded from hard drive. When the player completes a field, the time they took to finish it is recorded and saved, along with their name. The high score list should be sorted by score, and grouped by number of tiles. That means that a clear time of 30 seconds on a 10x10 grid is a higher score than a clear time of 5 seconds on a 4x4 grid.

Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- Java files needed to run your program.
- Informative README, telling the TA how to run your assignment.

All .java files should have a descriptive javadoc comment.

Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**

(If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

a. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect.