



CS 101

Lab 8 - Minesweeper

Due: December 4/5/6/7, 11:55 PM

Pre-lab Preparation

Before coming to lab, you are expected to have:

- Read Bruce chapters 15

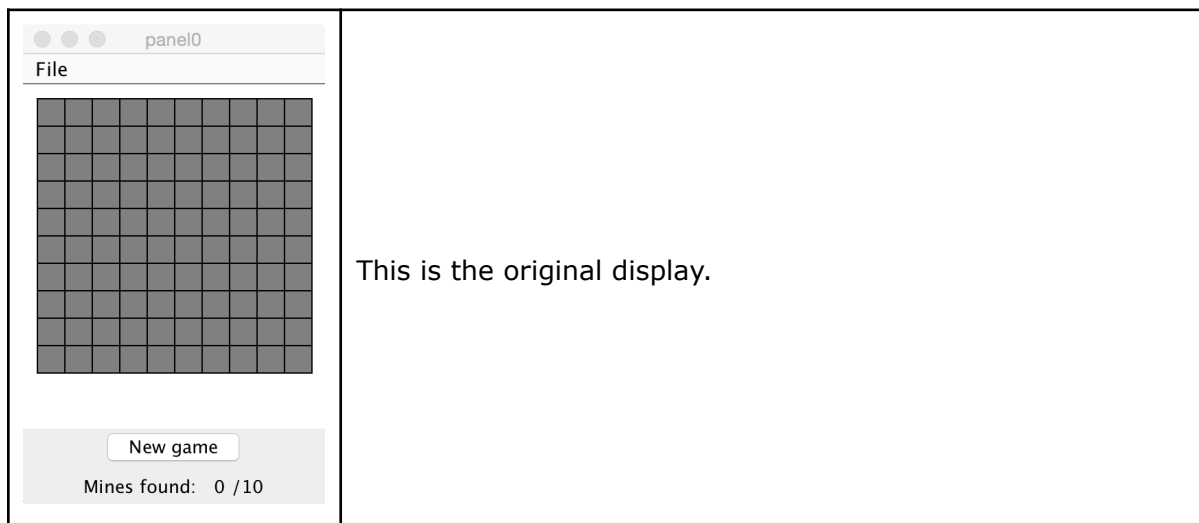
Goals:

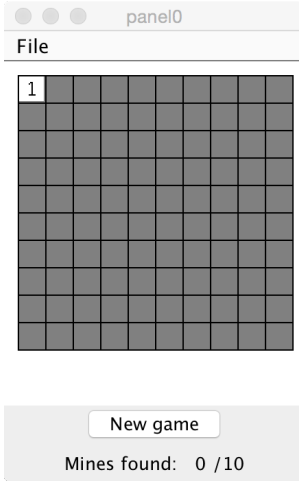
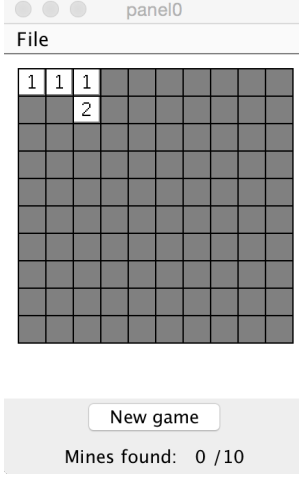
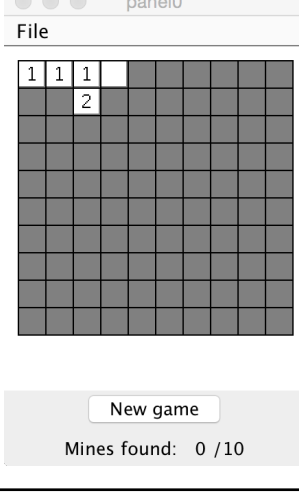
- To work with 2D arrays

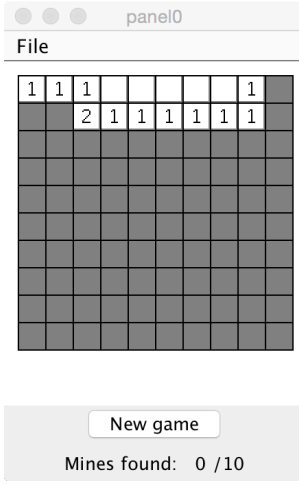
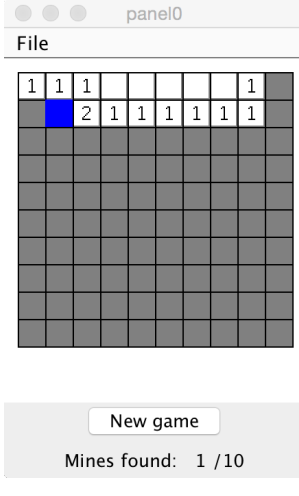
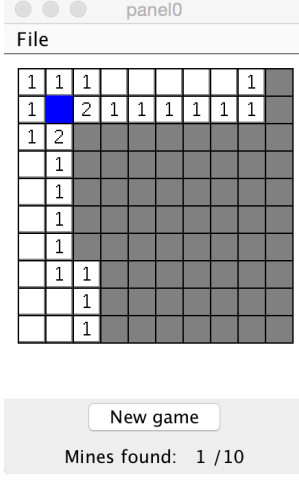
Introduction to the Assignment

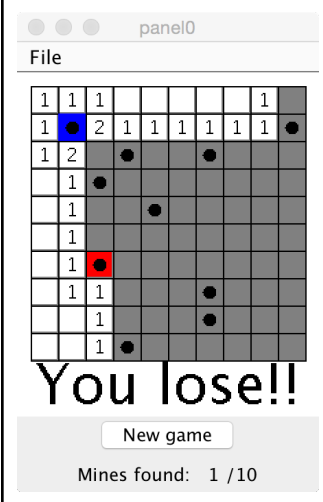
Minesweeper (along with its partner, Solitaire) has been responsible for countless hours of procrastination and lost productivity since it was included in Microsoft's Windows 3.1. The game seems simple enough. The board is divided into a grid and "mines" are hidden under some grid cells. The player's task is to determine where all the mines are without causing them to explode. If the player clicks on a grid cell and it contains a mine, the mine explodes, ending the game. To make the game remotely winnable, when a grid cell is clicked on and there is no mine, the cell is labeled with the number of neighboring cells that do have a mine. From this, the player can often learn more about the location of mines, and place a flag on the locations where mines are believed to exist. The goal of the game is to find all the mines without causing any of them to explode.

Here are some snapshots from this program:



	<p>When the user clicks on a cell that does not contain a mine, it displays the number of its neighbors that have mines. A neighbor is up, down, left, right, or diagonally in any direction. In this case, one of the 3 neighboring cells contains a mine.</p>
	<p>The player has now clicked on 4 cells without stepping on a mine. Notice that one of the cells has 2 neighbors with mines.</p>
	<p>If the user clicks on a cell with no neighbors with mines, the cell is displayed as blank, rather than with a 0 in it.</p>

	<p>When the user finds a blank cell, the user will know that it is safe to click on all of its neighbors. After clicking on safe cells in this way, we may end up with this screen.</p>
	<p>Once the game is in a state like the one above, the user will know that there must be a mine where the blue square appears here. The user can mark the square by right-clicking on it. The mark is drawn as a blue square, remembering where the user believes there is a mine.</p> <p>Notice also that the information at the bottom now indicates that the user has found 1 mine. This is just a count of the marked squares. The user might be wrong.</p>
	<p>After more clicks the user is in this state.</p>



if the user clicks on a cell that contains a mine, it explodes and the game is over. This is how the screen should look. The exploded mine is drawn with a red background. Marked mines are drawn with a blue background. Mines not yet located are drawn with the default, unexplored background.

The user can win by flagging all the mines and exploring all the remaining squares. In that case, a You win! message should appear.

The user should be able to click New game to start over at any time.

Playing the Game

The grid should be 10 x 10 and have 10 mines hidden at random locations. When the game starts, all the cells are unexposed. The game proceeds by the user clicking on cells, where mouse clicks have the following effect.

On a click:

- If the cell contains a mine, the player loses.
- If the cell contains a mark (but no mine), nothing happens.
- If the cell is unexposed (and does not contain a mine), it is exposed, revealing the number of its neighbors that contain mines.

On a right-click (or control-click):

- If the cell is unexposed, it is marked.
- If the cell is marked, the mark is removed.
- If the cell is exposed, nothing happens.
- Nothing special happens if the cell contains a mine.

The Program Design

This program will have 3 classes: Minesweeper, Grid and GridCell.

The Minesweeper class extends WindowController. It is responsible for defining the begin method and the event handlers.

- begin method - This method should create and layout the Swing components in the user interface. It should also call the Grid constructor.
- actionPerformed method - This method should handle the click on the New Game button.
- mousePressed method - This method will deal with the clicks on the Grid. You will not be able to use the usual onMouseClicked method because it is not possible to distinguish between a click and a right-click of the mouse. We will provide the code that makes this distinction. (More details below.)

The Grid class encapsulates the two-dimensional array that represents the 2D space of the game. Its responsibilities include:

- Constructing the individual GridCells
- Converting from x, y pixel coordinates to row, col coordinates
- Randomly placing the mines in the grid
- Calculating the number of neighbors of a cell that contain a mine
- Providing the logic about what should happen when the user clicks or right-clicks on a cell
- Determining when the game is over

The GridCell class remembers information about an individual cell:

- Whether the cell contains a mine
- The number of neighbors that contain mines
- Whether the cell is flagged
- Whether the contents of the cell has been revealed to the user
- Functionality to draw the cell according to its current state

Writing the Program

Create a new BlueJ project called Lab8.

Step 1: Create the display

First, layout the Swing GUI components and see that you can get the screen design correct.

Next, add the code to draw an empty grid on the canvas. The grid should have 10 rows and 10 columns.

Step 2: Add the mines

Add 10 mines to your grid. Initially, do not hide the mines so you can be sure the mines are being placed appropriately. You will find it convenient to write a method called `showAllMines` that simply draws all the mines. Until your program is working and you want to actually play the game, it would make sense to always display the mines on the grid for debugging purposes. When everything works, you can remove the call to `showAllMines`.

Step 3: Calculate the neighbor counts

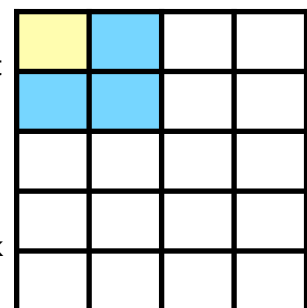
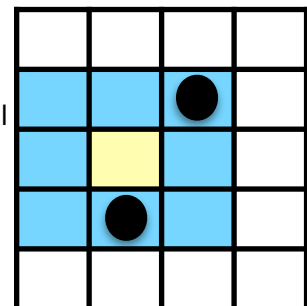
Write the code to determine the neighbor count of a cell. A neighbor is either directly to the left, right, above, below, any of the 4 diagonal squares. For example, this picture shows a yellow cell with its 8 neighbors in blue. The black circles represent mines. So, the yellow square in this case would have a neighbor count of 2.

You will need to be careful to consider the location of the cell. For example, a cell in the corner of the maze has only 3 neighbors as shown in the second example.

Test this code by displaying the neighbor count in all cells that do not have a mine. Display cells with a neighbor count of 0 as empty white rectangles.

Step 4: Translate x,y locations into row,col positions

Now hide the mines and the neighbor counts. Translate a mouse click on the canvas to a click on a grid cell. Display the contents of a cell when it is clicked on (either a mine or the neighbor count if there is



no mine).

Step 5: Explode a mine when it is clicked on

Update the code that handles mouse clicks so that if the user clicks on a cell containing a mine, that cell's background becomes red, all the mines are displayed and the user is told that he/she lost.

Step 6: Add flags

Allow an unexposed grid cell to be marked as a suspected mine location when it is "control-clicked".

Allow a marked cell to become unexposed by a "control-click". (That is, remove the mark.)

Update the code to handle a normal mouse click so that it does nothing if the cell is marked.

Update the label showing the number of mines found.

Step 7: Report a win

Detect and report a win when the last unexposed cell that does not contain a mine is exposed.

Step 8: New game

Add the ability to start a new game at any time.

Extra credit

1. Timer: You can create a timer that shows how long the user has been playing the game, measured in seconds. You can write a timer with a class that extends `ActiveObject`. The timer value should be displayed in a label. The timer should stop when the game ends, either with a win or loss.
2. Multiple levels: Add a `JComboBox` that allows the user to select different difficulty levels. Harder levels should have both more rows and columns and more mines.

Grading

10	Step 1: Create the display
10	Step 2: Add the mines
5	Step 3: Calculate the neighbor counts
5	Step 4: Translate x,y locations into row, col
5	Step 5: Explode a mine when it is clicked on
5	Step 6: Add flags
5	Step 7: Report a win
5	Step 8: New game
10	Comments

5	Constant declarations
5	Indentation
10	Variable names
5	Instance variables / local variables / parameters
5	If statements / loops
5	Good use of private methods
5	Arrays and for loops
100	Total
5	Extra credit: Timer
5	Extra credit: Multiple levels

Turning in Your Work

Your work will be automatically collected from your dev/cs101/Lab8 folder at the time that it is due. Please be sure your files are in the right place.

Over the course of the semester, you may have up to 5 late days total. Use them wisely! To request a late day, fill out the Google form on the course website so that we do not collect your assignment when it is due.