CS 1331 Homework 11

Due Friday, April 12th, 2013 8:00pm

Introduction

This assignment gives you experience with writing a GUI front end from back end code that we provide.

The front end that you are going to be creating is for a rook jumping maze.

A rook jumping maze is a common puzzle game. It consists of trying to find a valid path for a chess piece under certain constraints. You are given a board with weights from 1 to any number that validly fits on the board. The piece is placed somewhere on the board, and a goal tile has its weight set to 0.

The rook moves only vertically and horizontally. It moves by a number of tiles determined by the weight of the tile it is on.

You have been provided with code that will generate valid weights for a rook jumping maze. You are also provided with a player class, a rook class (which is a child of player – if you want to mess with any other chess pieces you can use these as building blocks), and a maze class.

As with the Pacman homework, make sure that you spend some time looking through the files that we provide before you tackle writing your own parts of this code.

11.1 MazePanel.java

This Java file will contain the bulk of our program. Like we subclassed JFrame on the last homework, this week we will be subclassing JPanel.

Your panel should have a reference to a Maze object, a 2D array of JButtons, and a variable to keep track of the number of moves that the player makes (only valid moves count).

Your panel needs to look like the image provided at the end of this file; the goal button needs to be colored white; the current button needs to be colored green and labeled "Here", and the valid move buttons need to be colored yellow. When you click a valid button, the player should move to that location and the colors should update.

When the user reaches the goal, display a dialog box telling them that they won, and how many moves it took them to reach the goal. It should also display the minimum number of moves needed to solve the maze. (Java tutorial on dialogs: http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html)

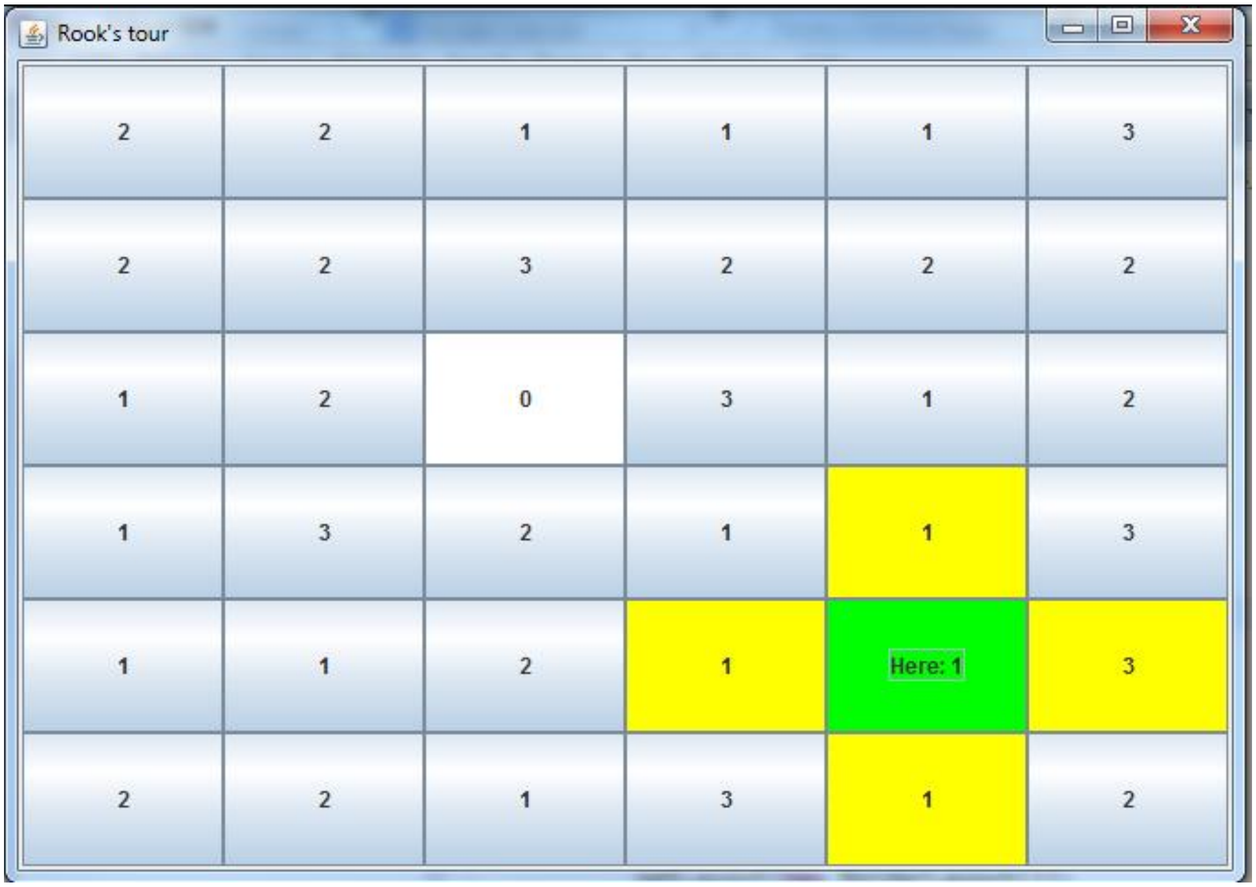
When creating your action listeners for your buttons, remember that this is a perfect situation in which to use parameterized action listeners.

Note: Occasionally you can get caught in a cycle that you cannot get out of when actually clicking around in the maze – this does not mean that your code is wrong; it is just how that maze was generated. If that happens, simply end the program and try again.

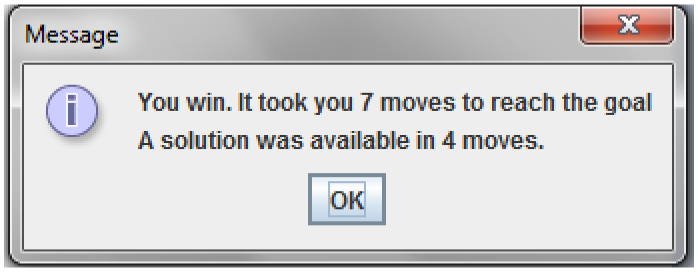
11.2 MazeDriver.java

This file will be very minimal. It should just contain a main method that creates an instance of our MazePanel and displays it to the user.

This is an example of what your program should look like:



2	2	1	1	1	3
2	2	3	2	2	2
1	2	0	3	1	2
1	3	2	1	1	3
1	1	2	1	Here: 1	3
2	2	1	3	1	2



You win. It took you 7 moves to reach the goal
A solution was available in 4 moves.

Turn-in Procedure

Turn in the following files to T-Square. When you are ready, make sure that you have actually **submitted** your files, and not just saved them as a draft.

- MazePanel.java
- MazeDriver.java

Note** Always submit .java files - never submit your .class files. Once you have submitted and received the email from T-Square, you should download your files into a fresh folder. Compile, run, and test those exact files that you turned in. It is pretty much foolproof if you use this technique. Anything less than this is a gamble. See "safe submission" info below. File issues, non-compiling files, non-source code files, etc are all problems and will cause a 0 for the HW. Also, make sure that your files are in on time; the real deadline is 8 pm. While you have until 2 am to get it turned in, we will not accept homework past 2 am for any reason. Don't wait until the last minute!

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. Read that email. Look at those filenames. We do not grade .class files. We require source code .java files.
3. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files and the fact that you submitted.
4. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
5. Recompile and test those exact files.
6. 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.)
    c. Reading the email and looking at the files you download helps prevent the turn-in of bytecode (.class) file for which you will receive no credit.
    d. Helps find last minute causes of files not compiling and/or running.