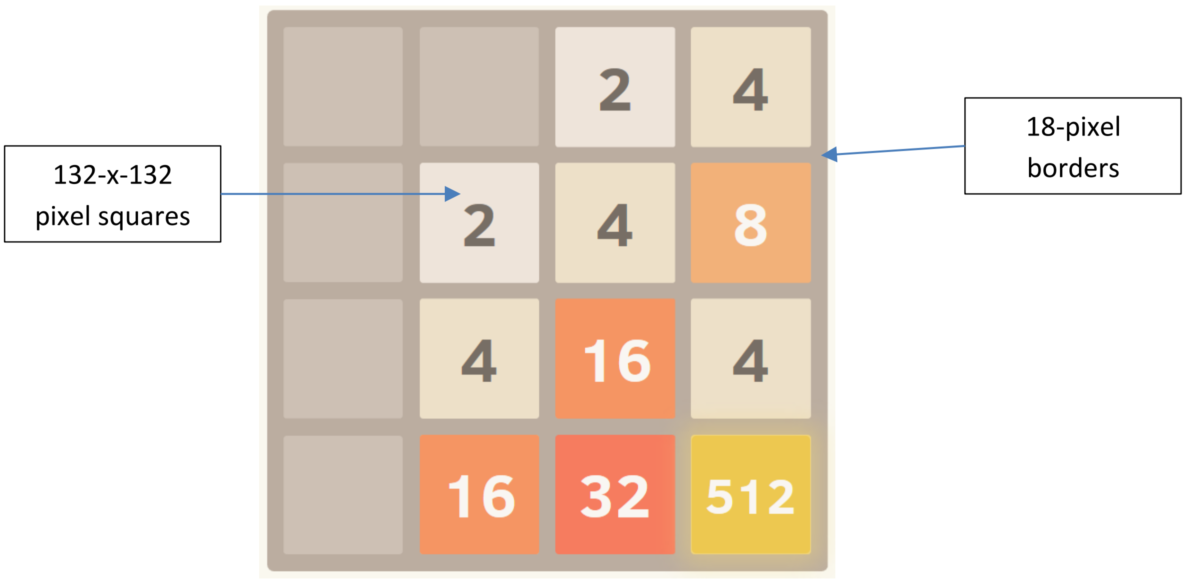
EECS 1510 – Introduction to Object-Oriented Programming in Java

Spring 2015 – Programming Assignment 2, Due 05/05/2015 at 11:59 PM

Your last major programming assignment will be to re-implement the 2048 game from the first assignment using a GUI, rather than the console I/O-based operations you used in the first one.

Below is a sample screen shot from the game:



Your individual tiles should be 132 pixel squares (with rounded corners), with 18-pixel-wide borders. Your cells should mimic the color scheme of the game from the internet.

The program should respond to only the up, down, left, and right *arrow keys* on the keyboard for movement (U/u/8, D/d/2, L/l/4, R/r/6 should not be allowed)

When the user moves, you are to animate the movement of the tiles with a PathTransition, using a straight line as the controlling shape (note: the zero-containing cells don't move).

When two cells merge, following the animation to put the two tiles "on top of each other", replace the existing tile with a new one containing the appropriate value, and then use a ScaleTransition animation to have the resulting merged tile "bloom" slightly and then return to its original 132-x-132 size. For more information on this particular animation, see the Oracle documentation for JavaFX 8, or, more specifically, https://docs.oracle.com/javafx/2/api/javafx/animation/ScaleTransition.html

With every turn, you are to display the number of turns taken and the current score, computed as before.

You should respond to CTRL+Z (the "universal" undo key sequence), to allow the user to un-do the last ***ten*** moves.

Your program should also have two buttons, which will respond to ALT+S and ALT+L to Save and Load the game. The "Save" operation is to save the current state of the board (and the ten undo buffers), the score, and move count in a file called 2048.dat. The "Load" function is to open that file, read back in the status of the game, and continue from where it was saved.

Similarly, there should be "Help" and "Exit" buttons, which respond to ALT+H and ALT+X. The "Help" button should display another window with the game's instructions. The Exit function should simply quit the game – it doesn't need to save it first (the user can use ALT+S for that).

When the user gets to 2048, you should ask them if they want to continue (until the board is full) via another window that appears. If they say "no", then the game ends then. If they say "yes", then let them keep playing until the board is full.

When the board is full, or they choose to "Exit" (above), display "game over" for 2 seconds before ending the program.

Whenever a game ends (via either mechanism), you should also save the high score in another file called HighScore.dat. If a game ends and the player has just set a new high score, then a "New High Score: xxxxx" window should appear for 5 seconds prior to the "game over" window, and you should play the "ta da" sound from Windows.

You are to code your program in Eclipse, using JavaFX 8. The code you write is to be yours and yours alone. Don't exchange code with other members of the class. Don't leave code on the whiteboard in the lab. You may work on the conceptual parts in groups, but when it goes from "concept" to "code", everyone is to write their own. Don't look for source code from the Internet; it's there, but resist the temptation. Failure to be the sole author of your code will be dealt with most severely. Cases of academic dishonesty can result in anything from a zero on the assignment in question to expulsion from the university.

Document your code in such a way that someone who is not familiar with how you have implemented the game would be able to read your code and understand how your code implements the game. I strongly suggest that you write blocks of comments first to tell you what to write, and then write the code in between the major comments. As you write the code, smaller block comments and/or line comments should become useful.

Your variables should all be documented. Each method (or even blocks of code within a method) should have a comment, explaining what the code does, and, unless it's obvious, how it does it.

Submit a 7-zip archive of your ENTIRE Eclipse workspace, even if you have other projects in your workspace. To do so, first exit Eclipse. Eclipse can take several seconds to shut down and close all open files. If you get any warnings about files being open during the zipping process, wait 15 seconds, and re-try. Next, start Windows Explorer (Windows key + E), and navigate to your workspace (which should be named something like "1510 – Smith, John" in the root of a flash drive, or on your laptop). Right-click on the workspace folder, select "7-Zip", and then "Add to 1510 – Smith, John. 7z" (substituting your name, of course). 7-Zip will then build an archive of your entire workspace in that 7z file. Upload that file to Blackboard as your submission.