Introduction

In lab 6 you laid out the basic model-view structure of the 2048 game. In this lab you will make the model a little more flexible, and start to build the core functionality of the game.

By the end of lab 7 your game must, at least:

- Have a 1 by 4 board (a single row of four columns)
- Handle the left and right arrow keys to move a number placed randomly into a column in the row.
- Randomly place a new number into an UNOCCUPIED square.
- Combine like numbers (2 combined with 2 produces 4, 4 combined with 4 produces 8, etc).

By the end of lab 8 you will add the remaining functionality (a 4 by 4 board, movement in all directions, etc.), as well as possibly (no promises) some extra credit features.

Tasks for lab 7

Here is a list of the functionality you need to build for lab 7. We have arranged the items in order so that later tasks build on earlier ones. You don't have to do things in this order, but you might find it helpful.

1) Modify your model so that it uses a HashMap<java.awt.Point,Integer> to store the occupied board positions. Remember that for now you should only have a board of one row and four columns.

2) Add a new tile in the leftmost position of the board.

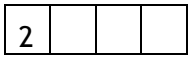
3) Modify your KeyListener so that a keyPress on the right arrow key makes all the tiles move (but not yet combine) as far to the right as possible. Here's a hint:

```
@Override public void keyPressed(KeyEvent e) {
     if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
          // move tile(s) all the way to the right
     }
}
```

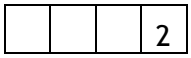
The other methods should have empty bodies.

3) Define a method on the model that will move (but not yet combine) a tile from the left as far as it can go to the right. The tile must be moved as in the 2048 game: it only moves when there are open squares to the right, and does not jump over any occupied positions.

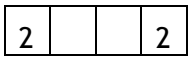
If the row looks like this to start:



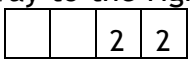
the tile must end up over to the right:



On the other hand, if to row looks like this to start:



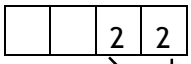
then the 4 should be moved all the way to the right first, then the 2, to end up with:



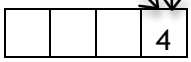
Before just diving in, play the real game a bit to see how movement is done. Sketch out an *algorithm* (a process) for doing movement on paper before you attempt to code it. Using diagrams to explain how the algorithm works is fine. Review your algorithm with your TA.

4) Add a new '2' tile in the leftmost position after each press of the right arrow key.

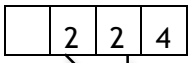
5) Write the code to combine two tiles of the same value when moving to the right. For example, if we start with



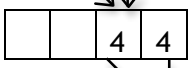
we should end up with:



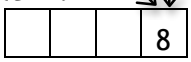
Similarly, if we start with:



We should end up with:



Trying to move right again must result in:



Before just diving in, play the real game a bit to see how the combination of tiles is done. Sketch out an *algorithm* (a process) for doing tile combination on paper before you attempt to code it. Using diagrams to explain how the algorithm works is fine. Review your algorithm with your TA.

6) Modify your 'add a new tile to the board' code so that it adds a tile to an unoccupied position, at random. You can do this by defining a method that returns an ArrayList<java.awt.Point> of all the unoccupied positions. Use the Collections.shuffle(…) method to randomize the order of elements. Extract the first element of the ArrayList by calling get(0) on the list, and add the new tile at that location.

7) Only add a new tile if the board changed since the last keyPress. The board has changed if any of the tiles has changed its location or any tiles were combined.

8) Modify your event handler to listen for and respond to presses on the left arrow key in addition to the right arrow key.

9) Handle movement and combination of tiles when moving left, in addition to when moving right.

Advice

Start early. Ask questions early. Consult your notes.

You may need to put in time outside of your scheduled recitation to complete labs 7 and 8, perhaps 2-6 extra hours. PLAN ACCORDINGLY. If you are not making progress after attending recitation, reviewing your notes and seriously working on the lab for 1-2 hours outside of recitation, see a TA or your instructor during office hours.

Submitting your project to Web-CAT

Submit the lab to Web-CAT as usual, making sure to select the correct section. There will be no automated grading of this lab – it will be manually graded. **Dues dates are listed on the course website.**