CSE115 Lab 6 Spring 2015

Introduction

2048 is a single-player puzzle-style game. You can read a bit about its history here: http://en.wikipedia.org/wiki/2048_(video_game). You can play the original game here: http://gabrielecirulli.github.io/2048/

Your task, over the course of labs 6, 7 and 8, is to build your own implementation of 2048. The overall learning goals for these three labs include:

- Writing a program with a simple graphical user interface
- Applying the Observer pattern for event handling
- Handling keyboard input
- Working with control structures (conditional and looping statements)
- Working with primitive data types
- Working with generic collections and generic mappings
- Building a software system incorporating a model-view separation
- Working with the same piece of code over an extended period of time

Some of these learning goals include things we have not yet covered in lecture, but don't worry - this lab is just the first of this sequence of three labs, and will focus on applying what we have already discussed in lecture:

- Writing a program with a simple graphical user interface
- Applying the Observer pattern for event handling
- Handling keyboard input
- Building a software system incorporating a model-view separation

You have roughly two weeks to complete this lab (see due dates on course website). In the first week you will work on a system incorporating a somewhat simplified version of the data model. By the second week you will have learned a little more about how we can structure data, and so will be expected to incorporate a more sophisticated data model.

Although you will be revising your work between weeks one and two, you will submit your work only after the second week of lab 6.

Preparatory requirements

1. Log in
2. Start Eclipse
3. Switch to the CVS Repository Exploring perspective
4. Check out the SP15-CSE115-Lab6 project from the Labs repository
5. Switch to the normal Java perspective.

Skills exercised

In this lab you will put the following skills, all discussed in lecture, into practice:

- Implementing interfaces to define event handlers
- Primitives (int, boolean, char)
- Working with Swing classes to define a simple graphical user interface (GUI)

Be sure to attend recitation both weeks of lab 6, and review both your lecture notes and readings prior to each recitation. Also give yourself enough time to visit office hours: don't put off writing code too long, lest you cannot get in to either TA or instructor office hours for help.

Lab requirements

You are responsible for the following requirements by the end of Lab 6.

Model

The data model of our program will, by the end of lab 8, keep track of the grid locations of all the tiles as well as the numeric value on each tile, among other information (such as, perhaps, the current score).

For lab 6 the data model will be a little simpler: it will keep track of four int values. The job of the model is to store these values, and provide ways to access each of them.

To give us a way to verify that the view updates when values in the model change you will define a method that assigns randomly generated values to the four int variables.

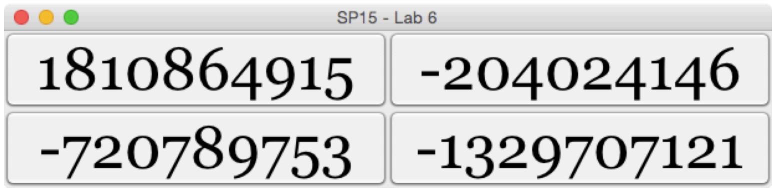
- ❖ *Hint: your TA will explain to you how to generate random int values.*

In order for the view to update when the model changes, the model must be associated with the view. Review your notes on association if you need help remembering your options for establishing an association relationship.

- ❖ *Hint: you should use a mutator method to establish the association (not the constructor).*

View

The view component of your program will provide a visualization of the data in the data model. Here's a picture of what the user interface might look like:



The view consists of four JButtons arranged in a 2x2 grid. Each JButton displays the value of one of the four int variables in the data model.

Whenever the values in the data model change the view must update. Because the view must register itself as an observer of the model, it must have a reference to the model. You can set up an association between the view and the model via the view's constructor.

- ❖ *Hint: your TA will explain/review how to set up your view class so that it implements the Runnable interface, and how to use SwingUtilities.invokeLater to correctly start up the graphics of your program.*

Model-View communication: Event Handling

In order for the view to update whenever the data model changes there needs to be communication between the two. The basic idea is this:

- the view registers as an observer of the model
- when the data changes (which is a type of event!) the model notifies its observer (the view)
- when notified of a change the view pulls the fresh data from the model (using the accessor methods defined for this purpose)

In a later phase of this three-lab project there will be communication from the view to the model too: by pressing the left/right/up/down arrow keys the user will be able to control the movement of tiles in the model (movements which the view then visualizes).

At this stage typing any key (pressing and releasing it) will trigger the randomization of data in the model. For this you will need to use a KeyListener (as opposed to the ActionListener we've seen several times in lecture). The basic idea is the same (KeyListener is also an example of the Observable/Observer design pattern).

A KeyListener specifies three methods:

```
/**
 * Invoked when a key has been typed.
 */
public void keyTyped(KeyEvent e);

/**
 * Invoked when a key has been pressed.
 */
public void keyPressed(KeyEvent e);

/**
 * Invoked when a key has been released.
 */
public void keyReleased(KeyEvent e);
```

A keyTyped event consists of a keyPressed event followed by a keyReleased event.

To handle key events you need to define a class implementing the KeyListener interface, and provide a definition for one of these three methods (I suggest the keyTyped method). The method will be called whenever a key is typed while your program is running. The method must be defined to call the randomization method of the model.

Interfaces

To help give your code structure at the start of lab 6, the starting point project in the repository has two interfaces defined. You can make your view and model classes implement them, as suggested in the comments of the files.

Be aware that the set of needed methods will change over the course of the project, perhaps even from the first to the second week of lab 6. This is just a normal part of the evolution of a piece of software. Just don't become too attached to any one method name :-)

Advice

Start early. Ask questions early. Consult your notes.

You may need to put in time outside of your scheduled recitation to complete lab 6, 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.**