1 Administrivia

- Pset3 Final due Friday, April 13 2012

2 Model View Controller

Model View Controller (MVC) is a common pattern for cleanly separating user interface, user input, and data manipulation. By making these three tasks independent, development and maintenance become much easier as changes to one ideally do not affect the others.

- Model - The representation of data (e.g. a list of words, a database)
- View - How the model is visually shown to the user
- Controller - Receives user input and notifies the model of said input

See slides of Lecture 12 on Stellar.

3 Swing

In lecture, we saw how graphical user interfaces (GUI) can be modeled as a hierarchy. For example, a window might contain a scroll pane and a panel which itself contains buttons. The interface as a whole can be manipulated by removing and adding elements to the tree hierarchy. Java provides a framework called Swing, which is an implementation of this idea. Also included are implementations of many common UI components. All components extend the JComponent abstract class for common functionality, and many are examples of MVC in and of themselves.

3.1 Common Swing Components

- JFrame - A top level window with a title and border. Not all components can be a root of a GUI hierarchy. JFrame is one that can, and often is the main window of an application.
- Container - Abstract class for components that can contain other components. Because a container’s contents are actually stored in its content pane, you never add() a component to a Swing container directly. Instead, you add the component to the container’s content pane. You do that by calling a method named getContentPane()
- JList - Displays a list of objects. Has its own model, a list of Java objects.
- JTextField - Contains a small area for both text input and output.
- JScrollPane - Displays a portion of a view. Can change this portion by using an adjustable scroll bar.

A reference for Swing components can be found at: http://docs.oracle.com/javase/tutorial/uiswing/components/index.html

3.2 Layout

We mentioned components contain other components, but haven’t talked about how they are organized. While you could specify each components absolute coordinates, it would be troublesome to ensure they are in the right place, don’t overlap etc. Swing components’ positions are better controlled by a LayoutManager, which will position components according to predefined patterns, and adjust them as the parent container is resized. For content panes, you set a layoutManager using the setLayout() method.

- BorderLayout - Components are positioned relative to the borders of the panel, or the center.
- GridLayout - Available space is partitioned to equally sized cells, and components fill a particular cell
- GroupLayout - Components are organized into parallel or sequential groups in the horizontal and vertical dimensions. This allows laying out components in an L shape. This is better seen in pictures, so look at this reference for a full description. http://docs.oracle.com/javase/tutorial/uiswing/layout/group.html

Java Reference: http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

### 3.3 Listeners

Listeners are objects implementing the publish-subscribe pattern. Listeners register themselves with particular Swing components and their methods will execute whenever the appropriate action (known as an event) occurs. For example, a mouse listener attached to a JFrame would have its mousePressed method execute when the user clicks on the JFrame. Additionally, information about the event (such as the coordinates of the mouse click) are stored in an Event object.

### 4 Simple Example

```java
//get the container for a JFrame, so that you can add components to it
Container cp = myFrame.getContentPane();

//create a group layout for container cp
 GroupLayout layout = new GroupLayout(cp);
 cp.setLayout(layout);

//create a group
 SequentialGroup h1 = layout.createSequentialGroup();

//add some components
 h1.addComponent(component1);
 h1.addComponent(listComponent);

//set the layout's horizontal group
 layout.setHorizontalGroup( h1);

//add an actionListener
 listComponent.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
     DefaultListModel listModel = (DefaultListModel) list.getModel();
     listModel.addElement("new word")
   }
 });
```

### 5 WordFinder

The goal is to implement a simple UI for a program that does the following. A user types in a sequence of characters into a text field. A list of words (loaded in from a text file) is then searched and the list of words containing those letters are displayed. The code for loading and searching the list is given, we will focus just on the UI.