Computer/Human Interaction
Lecture 15

Overview:
- Java Basics
- Introduction to Swing
- Input elements and listeners

References:
- JTUT, JSWI, JAPI

Java Basics

- Syntax is almost the same as C++
- Everything is part of a class, including static functions
- Application is a class with a static main function
- Class can extend another class; i.e., inherits superclass data and methods
- Class can implement an interface; i.e., inherits and must implement the methods of the interface

Java Basics 2

- All objects are dynamically created
  JLabel title = new JLabel ("A label");
- Console output using System.out
  System.out.print ("Output without newline");
  System.out.println ("Output with newline");
- String concatenation operator is +, objects convert using toString method
  title.setText ("A new label: " + number);

Java Swing

- Classes for building GUIs; modern replacement for AWT.
- Import javax.swing.* package
- Every application must have a base top-level component, most commonly: JFrame or JApplet
- GUI components are added to the top-level component, then displayed

HelloWorldSwing.java

- Demonstrates minimum Swing application, GUI with no interaction
- createAndShowGUI function is the application code
- Main function creates an object of an anonymous inner class that implements the Runnable interface
- The event-dispatcher will execute the code in the run function in a separate thread

Purple Pizza Parlor, v1

- Download from web or copy files
  /home/hwang/cs350/lecture15/*.*
- Can use Eclipse if know how
- Command-line compilation and execution:
  javac <filename>.java
  java <filename>
- On csserver, need to add to shell path variable
  /usr/java/jdk1.5.0_08/bin
GUI Components

- JLabel – label with text, can have HTML
- JTextField – one line of text
- JTextArea – multiple lines of text
- JRadioButton – radio button, grouped using ButtonGroup, text can have HTML
- JCheckBox – check boxes, text can have HTML
- JButton – button with text label, text can have HTML

GUI Components 2

- JPanel – container with layout, used to position components
- Most components allow setting borders
  ```java
  // create padding above and below
  title.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));
  // create border with a title and padding
  panel.setBorder(BorderFactory.createCompoundBorder(
      BorderFactory.createTitledBorder("Choose one"),
      BorderFactory.createEmptyBorder(5, 5, 5, 5)));
  ```

Java Events

- GUI interaction is event-driven similar to VB
- Event-dispatching thread is automatically created
- Events include: clicks, keypress, mouse move, etc.
- Event-handling interfaces defined by AWT, so import `java.awt.*` and `java.awt.event.*` packages

Listeners

- Interaction is provided by adding a `listener` to a component; defined by the AWT, so import those packages
- Listeners are interfaces with methods that must be implemented. E.g., ActionListener interface has `actionPerformed`; ItemListener interface has `itemStateChanged`
- Can use an anonymous inner class to implement these interfaces.

Layout Managers

- Layout manager determines where components appear in a panel; each panel can have a different layout manager (more next time)
- Examples in PizzaApp1:
  - FlowLayout: default manager, left to right placement
  - BorderLayout: top, bottom, left, right, and center
  - GridLayout: equal-sized cells of specified number of rows and columns (0 is undetermined)
  - BoxLayout: centered in either horizontally or vertically

In-class Exercise

- Try the following
  - Separate the toppings into meats and vegetables. Create a bordered panel of checkboxes for each.
  - Have the handler for the submit action do more, e.g. check that that name and address fields are filled in and write a message to the debug area if not.
  - Add pricing information and compute a total that is displayed in the debug area or on the console.