ch17-18: Swing

4 Feb 2008
CMPT166
Dr. Sean Ho
Trinity Western University

What's on for today

- Swing: vs. AWT, lightweight vs. heavyweight
  - Superclass structure of Swing
    - Nested and inner classes
  - Event handling
    - Delegate classes
    - Subclasses of ActionEvent
    - Sub-interfaces of EventListener
  - Swing widgets
    - JLabel, JTextField, JPasswordField
    - JButton, JCheckBox, JRadioButton, JComboButton
    - ItemListener interface and ItemEvent class

JOptionPane

- import javax.swing.JOptionPane;
- showInputDialog(String prompt)
  - Prompt to the user, returns a string
- showMessageDialog(pos, msg, title, type)
  - Show dialog box to user
  - pos: null for centered in screen
    - Or pass a reference to widget
  - type: JOptionPane.INFORMATION_MESSAGE
    - Or ERROR_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE

Swing vs. AWT, light vs. heavy

- A Java app can mix Swing and AWT features
- Swing is written in Java and is more portable
  - AWT relies on local platform's windowing system: varies across platforms
- Lightweight: not tied to local platform
- Heavyweight: depends on local platform
  - AWT widgets are heavyweight
  - Most Swing widgets are lightweight

Common superclasses in Swing

- Everything is an Object
- Component (java.awt): GUI, both Swing and AWT
- Container (java.awt): organizes Components
- JComponent (javax.swing):
  - Superclass of all lightweight Swing components
  - Pluggable look-and-feel, shortcut keys, tooltips, localization, etc.
  - JLabel, JTextField, JButton, JCheckBox, JComboBox, JList, JPanel, etc.

Nested classes

- We've seen non-public helper classes defined in the same file as the primary public class:
  - public class Primary { ... }
  - class Helper1 { ... }
- We can also define classes nested in another:
  - public class Primary {
    - class Helper1 { ... }
  - }
- Inner classes: non-static nested classes
  - Can access even private items of top-level
  - Often used for event handlers

Event handling

- We've seen examples like this:

```
public class Histogram extends JPanel implements
  ActionListener {
    public Histogram() { ...
        widget.addActionListener( this ); ... };          // register
    public void actionPerformed() { ... };
    public static void createAndShowGUI() { ... };
    public static void main() { ... };
}
```

- One class does three functions:
  - main()/createAndShowGUI(): setup window
  - Constructor: create, layout widgets
  - actionPerformed(): event handler

Delegate classes

- Alternatively: use separate classes

```
public class Histogram extends JPanel {
    public Histogram() { ...
        InputHandler handler = new InputHandler();
        widget.addActionListener( handler ); ... };
    private class InputHandler implements ActionListener {
        public void actionPerformed() { ... };
    }
}
public class HistogramTest {     // in separate file
    public static void createAndShowGUI() { ... };
    public static void main() { ... };
}
```

- Uses inner class to define event handler