Introduction to Swing

8 Feb 2010
CMPT166
Dr. Sean Ho
Trinity Western University

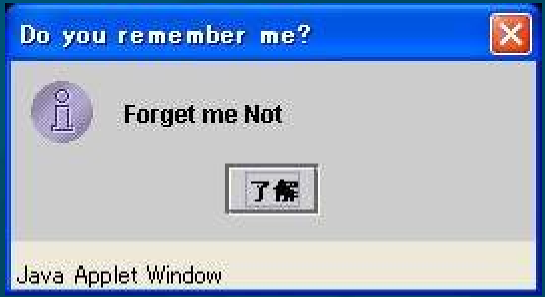
What's on for today

- Basic dialogues with JOptionPane
- Swing: vs. AWT, lightweight vs. heavyweight
- Superclass structure of Swing
- Swing windows: JFrame
- Event handling: ActionListener
    - Anatomy of a Swing program
    - Nested and inner classes
    - Delegate classes

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

- Component (java.awt): GUI, both Swing + 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.

JFrame: a Swing window

- To create a window in Swing, subclass JFrame
  - **import javax.swing.JFrame;**
  - **public class MyWin extends JFrame {**
- In the constructor, call the superclass first:
  - **public MyWin() { super();**
- Add widgets, and show the window:
  - **setVisible(true);**
- By default, the 'X' button merely hides the window. Change this with:
  - **setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);**

Event handling: ActionListener

- If you want your widgets to respond to user actions, you must provide an event handler:
  - An object that implements the ActionListener interface
  - Implements an actionPerformed() method, which takes one ActionEvent parameter
    - **import java.awt.*;**
- When a button is clicked, actionPerformed() is called: all relevant info is in the ActionEvent
- The event handler can be a different object or the same object as your JFrame window

All-in-one Swing program

- The Histogram example does triple-duty:

```
public class Histogram extends JFrame implements
  ActionListener {
    public Histogram() { …
        widget.addActionListener( this ); … };
    public void actionPerformed() { … };
    public static void main() { … new Histogram(); … };
}
```

- main(): create new window
- Constructor: create+layout widgets
- actionPerformed(): event handler

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 are non-static nested classes
    - Can access even private items of top-level
    - Often used for event handlers

Delegate classes

- Use inner classes to define event handlers:

```
public class Histogram extends JFrame {
    public Histogram() { ...
        MyHandler handler = new MyHandler();
        widget.addActionListener( handler );
    ... };

    private class MyHandler implements ActionListener
    { public void actionPerformed() { ... }; }

    public static void main() { ... new Histogram(); ... };
}
```