Event-Driven Programs

CS 1025 Computer Science Fundamentals I

Stephen M. Watt

University of Western Ontario
Command Line vs Events

• So far, all our programs have started from a “main” method which calls other methods, takes input, makes objects, prints output, etc.

• Another model of programming is to have programs that respond to “events.”

• These events could be
  – Mouse clicks
  – Key presses
  – Pressing a brake pedal
  – Lifting a telephone receiver
Event-Driven Code

- Event-driven code associates particular functions or methods to be called when specified events occur.

- In Java, this is done by creating “handler” objects from which specific methods get invoked.

- These handler objects are then placed on the things that receive events, like buttons, scroll bars etc.
• Graphical user interfaces are programmed in Java using
  – the abstract window toolkit (old, lower level) and
  – Swing (new, higher level)

• The next slides give an example of creating a simple Swing application.
Start Eclipse
Create a Project
Create a Class

Java Class
The use of the default package is discouraged.

Source folder: Buttons/src
Package: (default)
Enclosing type:

Name: MyButtonBox
Modifiers: public
Superclass: java.lang.Object
Interfaces:

Which method stubs would you like to create?
- public static void main(String[]) args
- Constructors from superclass
- Inherited abstract methods

Do you want to add comments as configured in the properties of the current project?
- Generate comments

Finish Cancel
The Program – note the Listener
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyButtonBox extends JFrame {
    public static void main(String[] args) {
        new MyButtonBox();
    }

    private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent ev) {
            System.out.println("That tickles...");
        }
    }

    public MyButtonBox() {
        super("Using JButton");
        Container content = getContentPane();
        content.setLayout(new FlowLayout());
        JButton button1 = new JButton("Print on the console!");
        button1.addActionListener(new MyActionListener());
        content.add(button1);
        JButton button2 = new JButton("Java");
        content.add(button2);
        pack();
        setVisible(true);
    }
}