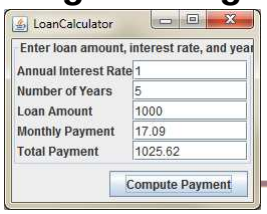
Java Event-Driven Programming

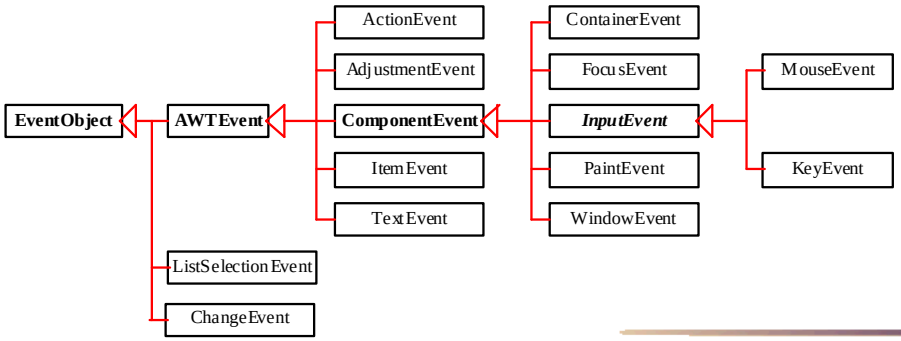
- **Sections 1 – 7: Events and Source, Class Listeners, Registering Listeners and Handling Events, Inner Classes, JButton and JTextField Sources**
- **Create a GUI program that lets the user enter numerical data process it and displays results**
- **What is required to accomplish the task?**
  - **What comes first the GUI design or event coding?**
  - **What Java class objects that need to be used?**
  - **How do you know if program works?**
- **Procedural vs. Event-Driven Programming**
  - ***Procedural programming* is executed in procedural order**
  - **Event-driven programming, code is executed upon activation of events.**



R.M. Laurie 1

Event Classes

- **An *event* can be defined as a signal to the program that something has happened**
  - **The event is triggered by user actions such as mouse clicks and keystrokes, or operating system timers**
  - ***Source object* is the component that triggers the event**



 2

Selected User Actions

Event *source object* contains whatever properties are pertinent to the event.

You can identify the *source object* of the event using the getSource() instance method in the EventObject class. The subclasses of EventObject deal with special types of events, such as button actions, window events, component events, mouse movements, and keystrokes.

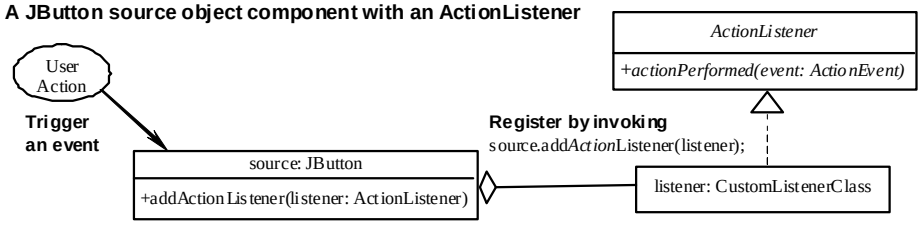
Table 15.1 lists external user actions, source objects, and generated events

User Action	*Source Object*	Event Type Generated
Click a button	`JButton`	`ActionEvent`
Click a check box	`JCheckBox`	`ItemEvent`, `ActionEvent`
Click a radio button	`JRadioButton`	`ItemEvent`, `ActionEvent`
Press return on a text field	`JTextField`	`ActionEvent`
Select a new item	`JComboBox`	`ItemEvent`, `ActionEvent`
Window opened, closed, etc.	`Window`	`WindowEvent`
Mouse pressed, released, etc.	`Component`	`MouseEvent`
Key released, pressed, etc.	`Component`	`KeyEvent`

 3

The Delegation Model

A JButton source object component with an ActionListener

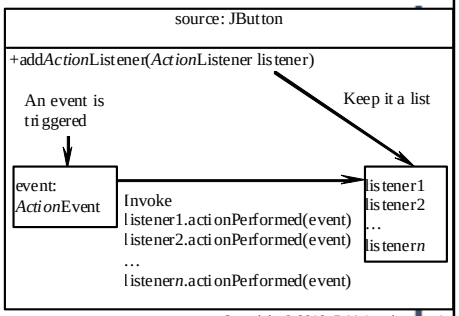



```
1. // Create Source Objects for GUI
2. JButton btnYes = new JButton("Yes");
3. JButton btnNo = new JButton("No");

4. // Create Listener Objects for GUI
5. ListenerYes listenerYes = new ListenerYes();
6. ListenerNo listenerNo = new ListenerNo();

7. // Register Listeners Objects for GUI
8. btnYes.addActionListener(listenerYes);
9. btnNo.addActionListener(listenerNo);
```

Internal function of a JButton object



 4

ActionListener HandleEvent

```
1.  import javax.swing.*;
2.  import java.awt.event.*;
3.  public class LikeJava extends JFrame {
4.    // This is the Constructor used to create GUI
5.    public LikeJava() {
6.      // Create Source Objects for GUI
7.      JButton btnYes = new JButton("Yes");
8.      JButton btnNo = new JButton("No");
9.      JLabel lblQuestion = new JLabel("Do you like to do Java GUI programming?");
10.     JPanel panMain = new JPanel();
11.     panMain.add(lblQuestion);
12.     panMain.add(btnYes);
13.     panMain.add(btnNo);
14.     add(panMain); // Add panel to frame
15.     // Create Listener Objects for GUI
16.     ListenerYes listenerYes = new ListenerYes();
17.     ListenerNo listenerNo = new ListenerNo();
18.     // Register Listeners Objects for GUI
19.     btnYes.addActionListener(listenerYes);
20.     btnNo.addActionListener(listenerNo);
21.   }
22.   public static void main(String[] args) {
23.     JFrame fraWindow = new LikeJava();
24.     fraWindow.setTitle("Java GUI Programming");
25.     fraWindow.setSize(300, 100);
26.     fraWindow.setLocation(200, 100);
27.     fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28.     fraWindow.setVisible(true);
29.   }
30. }
31. class ListenerYes implements ActionListener {
32.   public void actionPerformed(ActionEvent event) {
33.     System.out.println("I am glad you like GUI programming.");
34.   }
35. }
36. class ListenerNo implements ActionListener {
37.   public void actionPerformed(ActionEvent event) {
38.     System.out.println("You will like Java GUI if you study!");
39.   }
40. }
```


```
I am glad you like GUI programming.
You will like Java GUI if you study!
You will like Java GUI if you study!
I am glad you like GUI programming.
You will like Java GUI if you study!
```

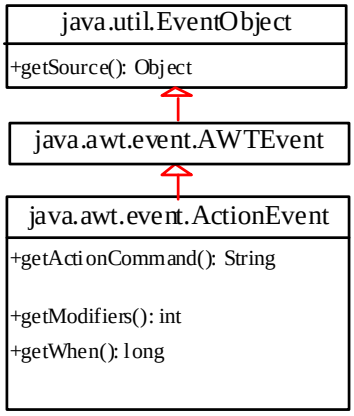
Selected Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
ActionEvent	*ActionListener*	*actionPerformed(ActionEvent)*
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent)
		windowOpened(WindowEvent)
		windowIconified(WindowEvent)
		windowDeiconified(WindowEvent)
		windowClosed(WindowEvent)
		windowActivated(WindowEvent)
		windowDeactivated(WindowEvent)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent)
		componentRemoved(ContainerEvent)
MouseEvent	MouseListener	mousePressed(MouseEvent)
		mouseReleased(MouseEvent)
		mouseClicked(MouseEvent)
		mouseExited(MouseEvent)
		mouseEntered(MouseEvent)
KeyEvent	KeyListener	keyPressed(KeyEvent)
		keyReleased(KeyEvent)
		keyTypeed(KeyEvent)

 6

java.awt.event.ActionEvent

- Event object contains information about event
  - `event.getSounce()` returns the source object
  - `event.getWhen()` returns time event occured



Returns the object on which the event initially occurred.

Returns the command string associated with this action. For a button, its text is the command string.

Returns the modifier keys held down during this action event.

Returns the timestamp when this event occured. The time is the number of milliseconds since January 1, 1970, 00:00:00 GMT.

Inner Classes

- Inner class: A class that is a member of another class
  - Advantages: Inner classes can make programs simple and concise
  - Inner class can reference data and methods defined in the outer class
  - No need to pass reference of outer class to constructor of inner class
- Inner class supports the work of its containing outer class
  - The inner class InnerClass in OuterClass is compiled into *OuterClass$InnerClass*.class
- inner class can be declared
  - public, protected, or private
  - Same visability rules applied to a member of the class
- inner class can be declared static.
  - static inner class can be accessed using the outer class name.
  - static inner class cannot access nonstatic members of outer class

```
/public class OuterClassExample {
  private int data;
  /** A method in the outer class */
  public void m() {
    // Do something
  }
  // An inner class
  class InnerClass {
    /** A method in the inner class */
    public void mi() {
      // Can reference data and method
      // defined in its outer class
      data++;
      m();
    }
  }
}
```


Outer Class & Inner Class Example

```
1.  public class OuterClass2 {
2.    private int nX = 42;
3.    MyInner oInner = new MyInner();
4.    public static void main(String[] args) {
5.      OuterClass2 oOuter = new OuterClass2();
6.      oOuter.printXouter();
7.      oOuter.oInner.printYinner();
8.      oOuter.oInner.printXinner();
9.    }
10.   public void printXouter() {
11.     System.out.println("X=" + nX + " (outer)");
12.   }
13.   class MyInner {
14.     private int nY = 23;
15.     /** A method in the inner class */
16.     public void printYinner() {
17.       System.out.println("Y=" + nY + " (inner)");
18.     }
19.     public void printXinner() {
20.       System.out.println("X=" + nX + " (inner)");
21.     }
22.   }
23. }
```

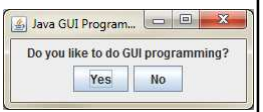
X=42 (outer)
Y=23 (inner)
X=42 (inner)

 9

Inner Class Listeners

```
1.  import javax.swing.*;
2.  import java.awt.event.*;
3.  public class LikeJavaInnerClass extends JFrame {
4.    JLabel lblQuestion; // Need to declare here for inner class visability
5.    public LikeJavaInnerClass() {
6.      JButton btnYes = new JButton("Yes");
7.      JButton btnNo = new JButton("No");
8.      lblQuestion = new JLabel("Do you like to do GUI programming?");
9.      JPanel panMain = new JPanel();
10.     panMain.add(lblQuestion);
11.     panMain.add(btnYes);
12.     panMain.add(btnNo);
13.     add(panMain); // Add panel to frame
14.     ListenerYes listenYes = new ListenerYes();
15.     btnYes.addActionListener(listenYes);
16.     ListenerNo listenNo = new ListenerNo();
17.     btnNo.addActionListener(listenNo);
18.   }
19.   public static void main(String[] args) {
20.     JFrame fraWindow = new LikeJavaInnerClass();
21.     fraWindow.setTitle("Java GUI Programming");
22.     fraWindow.setSize(250, 100);
23.     fraWindow.setLocation(200, 100);
24.     fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25.     fraWindow.setVisible(true);
26.   }
27.   class ListenerYes implements ActionListener {
28.     public void actionPerformed(ActionEvent event) {
29.       lblQuestion.setText("I love it, because it is fun to design GUI");
30.     }
31.   }
32.   class ListenerNo implements ActionListener {
33.     public void actionPerformed(ActionEvent event) {
34.       lblQuestion.setText("I hate it, because it is way too complicated");
35. } } }
```

- A listener class is designed specifically to create a listener object for a GUI component (e.g., a button)
- Define the listener class inside the frame class as an inner class, because not shared by other applications

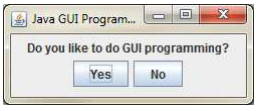


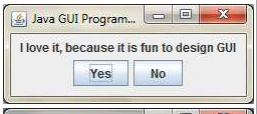
Anonymous Inner Classes

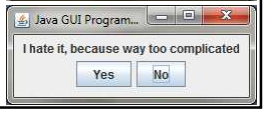
- An anonymous inner class must always extend a superclass or implement an interface, but it cannot have an explicit extends or implements clause
- An anonymous inner class must implement all the abstract methods in the superclass or in the interface
- An anonymous inner class always uses the no-arg constructor from its superclass to create an instance
- If an anonymous inner class implements an interface, the constructor is Object()
- An anonymous inner class is compiled into a class named OuterClassName$*n*.class
  - Outer class Test has two anonymous inner classes, these two classes are compiled into Test$1.class and Test$2.class
- Inner class listeners can be shortened
  - An anonymous inner class is an inner class without a name
  - It combines declaring an inner class and creating an instance of the class in one step

Anonymous Class Listeners

```
1.  import javax.swing.*;
2.  import java.awt.event.*;
3.  public class LikeJavaInnerAnon extends JFrame {
4.    JLabel lblQuestion; // Need to declare for inner class visability
5.    public LikeJavaInnerAnon() {
6.      JButton btnYes = new JButton("Yes");
7.      JButton btnNo = new JButton("No");
8.      lblQuestion = new JLabel("Do you like to do GUI programming?");
9.      JPanel panMain = new JPanel();
10.     panMain.add(lblQuestion);
11.     panMain.add(btnYes);
12.     panMain.add(btnNo);
13.     add(panMain); // Add panel to frame
14.     ListenerYes listenYes = new ListenerYes();
15.     btnYes.addActionListener( // Note open parenthesis
16.       new ActionListener() {  // Anonymous Action Listener btnYes starts here
17.         public void actionPerformed(ActionEvent theEvent) {
18.           lblQuestion.setText("I love it, because it is fun to design GUI");
19.         }   // Closing actionPerformed method
20.       }   // Closing Anonymous action listener object for btnYes
21.     );  // Note closing parenthesis
22.     ListenerNo listenNo = new ListenerNo();
23.     btnNo.addActionListener(  // Note open parenthesis
24.       new ActionListener() {  // Anonymous Action Listener btnNo starts here
25.         public void actionPerformed(ActionEvent theEvent) {
26.           lblQuestion.setText("I hate it, because way too complicated");
27.         }   // Closing actionPerformed method
28.       }   // Closing Anonymous action listener object for btnNo
29.     );  // Note closing parenthesis
30.   }
31.   public static void main(String[] args) {
32.     JFrame fraWindow = new LikeJavaInnerAnon();
33.     fraWindow.setTitle("Java GUI Programming");
34.     fraWindow.setSize(250, 100);
35.     fraWindow.setLocation(200, 100);
36.     fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37.     fraWindow.setVisible(true);
38.   }
39. }
```







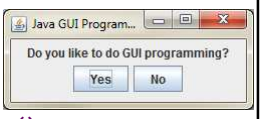
Alternative Way of Defining Listener Classes

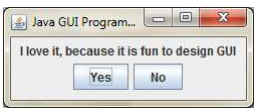
- Create Common Listener & Detect Source
  - Register one listener with several buttons
  - Let the listener detect the event source, that is which button fires the event
- You may also define the custom frame class that implements ActionListener
  - Frame extends JFrame and implements ActionListener
  - Class is listener class for action events
  - Not prefered because too much in one class
  - Described in Listing 16.5

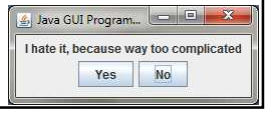
 13

Create Common Listener & Detect Source

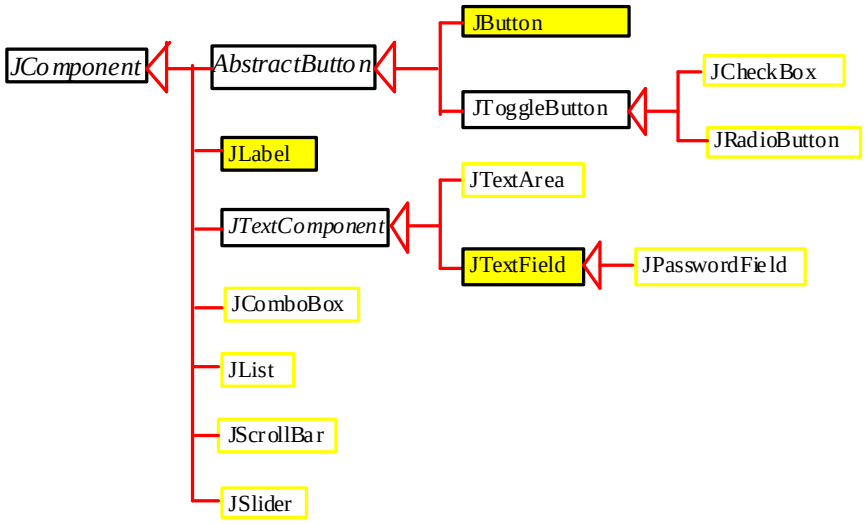
```
1.  import javax.swing.*;
2.  import java.awt.event.*;
3.  public class LikeJavaAlt1 extends JFrame {
4.    JButton btnYes = new JButton("Yes");
5.    JButton btnNo = new JButton("No");
6.    JLabel lblQuestion = new JLabel("Do you like to do GUI programming?");
7.    public LikeJavaAlt1() {
8.      JPanel panMain = new JPanel();
9.      panMain.add(lblQuestion);
10.     panMain.add(btnYes);
11.     panMain.add(btnNo);
12.     add(panMain); // Add panel to frame
13.     AllButtonsListener listenerButtons = new AllButtonsListener();
14.     btnYes.addActionListener(listenerButtons);
15.     btnNo.addActionListener(listenerButtons);
16.   }
17.   class AllButtonsListener implements ActionListener {
18.     public void actionPerformed(ActionEvent theButtonEvent) {
19.       if(theButtonEvent.getSource() == btnYes)
20.         lblQuestion.setText("I love it, because it is fun to design GUI");
21.       else if(theButtonEvent.getSource() == btnNo)
22.         lblQuestion.setText("I hate it, because way too complicated");
23.     }   // Closing actionPerformed method
24.   }
25.   public static void main(String[] args) {
26.     JFrame fraWindow = new LikeJavaAlt1();
27.     fraWindow.setTitle("Java GUI Programming");
28.     fraWindow.setSize(250, 100);
29.     fraWindow.setLocation(200, 100);
30.     fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31.     fraWindow.setVisible(true);
32.   }
33. }
```







Introductory GUI Components

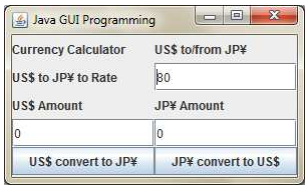


Beginning Swing JComponents

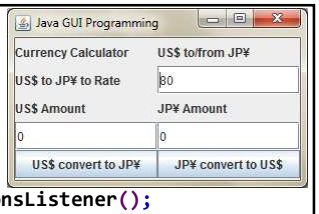
- **JLabel Constructors**
  - `JLabel()`
  - `JLabel(String text)`
  - `JLabel(Icon icon)`
  - `JLabel(String text, Icon icon, int horizontalAlignment)`
- **JButton Constructors**
  - `JButton()`
  - `JButton(String text)`
  - `JButton(String text, Icon icon)`
  - `JButton(Icon icon)`
- **JTextField Constructors and Methods**
  - `JTextField(int columns)`
  - `JTextField(String text)`
  - `JTextField(String text, int columns)`
  - `JTextField Constructors`
  - `getText()  // Returns the string from the text field.`
  - `setText(String text)  // Puts string in the text field.`

Step 1: Make GUI for Currency Converter

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.    public class Calculator_USD_JPY extends JFrame {
5.      private JButton btn2USD = new JButton("JP¥ convert to US$");
6.      private JButton btn2JPY = new JButton("US$ convert to JP¥");
7.      private JTextField txtUSD = new JTextField("0.00");
8.      private JTextField txtJPY = new JTextField("0");
9.      private JTextField txtY2DRate = new JTextField("80");
10.     public Calculator_USD_JPY() {
11.       JPanel panMain = new JPanel(new GridLayout(5,2));
12.       panMain.add(new JLabel("Currency Calculator "));
13.       panMain.add(new JLabel("US$ to/from JP¥"));
14.       panMain.add(new JLabel("US$ to JP¥ to Rate"));
15.       panMain.add(txtY2DRate);
16.       panMain.add(new JLabel("US$ Amount"));
17.       panMain.add(new JLabel("JP¥ Amount"));
18.       panMain.add(txtUSD);
19.       panMain.add(txtJPY);
20.       panMain.add(btn2JPY);
21.       panMain.add(btn2USD);
22.       add(panMain); // Add panel to frame
23.     }
24.     public static void main(String[] args) {
25.       JFrame fraWindow = new Calculator_USD_JPY();
26.       fraWindow.setTitle("Java GUI Programming");
27.       fraWindow.setSize(300, 180);
28.       fraWindow.setLocation(200, 100);
29.       fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30.       fraWindow.setVisible(true);
31.     }
32.   }
```



Step 2: Process Events for Currency Converter




```
20.       panMain.add(btn2JPY);
21.       panMain.add(btn2USD);
22.       add(panMain); // Add panel to frame
23.       ButtonsListener listenerButtons = new ButtonsListener();
24.       btn2JPY.addActionListener(listenerButtons);
25.       btn2USD.addActionListener(listenerButtons);
26.     }
27.     private class ButtonsListener implements ActionListener {
28.       public void actionPerformed(ActionEvent theButtonEvent) {
29.         double dRate = Double.parseDouble(txtY2DRate.getText());
30.         if(theButtonEvent.getSource() == btn2JPY) {
31.           double dUSD = Double.parseDouble(txtUSD.getText());
32.           txtJPY.setText(String.format("%.0f", dUSD * dRate));
33.         }
34.         else if(theButtonEvent.getSource() == btn2USD) {
35.           double dJPY = Double.parseDouble(txtJPY.getText());
36.           txtUSD.setText(String.format("%.2f", dJPY / dRate));
37.         }
38.       }
39.     }
40.     public static void main(String[] args) {
41.       JFrame fraWindow = new Calculator_USD_JPY();
42.       fraWindow.setTitle("Java GUI Programming");
43.       fraWindow.setSize(300, 180);
44.       fraWindow.setLocation(200, 100);
45.       fraWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46.       fraWindow.setVisible(true);
47.     }
48.   }
```