CS2110. GUIs: Listening to Events

Also

anonymous classes

Download the demo zip file from course website and look at the demos of GUI things: sliders, scroll bars, combobox listener, etc.

Listening to events: mouse click, mouse movement into or out of a window, a keystroke, etc.

• An event is a mouse click, a mouse movement into or out of a window, a keystroke, etc.

• To be able to "listen to" a kind of event, you have to:
  1. Have some class C implement an interface IN that is connected with the event.
  2. In class C, override methods required by interface IN; these methods are generally called when the event happens.
  3. Register an object of class C as a listener for the event. That object’s methods will be called when event happens.

We show you how to do this for clicks on buttons, clicks on components, and keystrokes.

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

Listening to a JButton

1. Implement interface ActionListener:
   
   ```java
   public class C extends JFrame implements ActionListener {
   }
   ```

2. In class C override actionPerformed, which is to be called when button is clicked:
   
   ```java
   public void actionPerformed(ActionEvent e) {
   }
   ```

3. Add an instance of class C an "action listener" for button:
   
   ```java
   button.addActionListener(this);
   ```

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

Listening to a JButton

1. Implement interface ActionListener:
   
   ```java
   public class C extends JFrame implements ActionListener {
   }
   ```

2. In class C override actionPerformed, which is to be called when button is clicked:
   
   ```java
   public void actionPerformed(ActionEvent e) {
   }
   ```

3. Add an instance of class C an "action listener" for button:
   
   ```java
   button.addActionListener(this);
   ```

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing

What is a JButton?

Instance: associated with a “button” on the GUI, which can be clicked to do something

jb1 = new JButton()  // jb1 has no text on it
jb2 = new JButton("first")  // jb2 has label “first” on it
jb2.isEnabled()  // true iff a click on button can be detected
jb2.setEnabled(b);  // Set enabled property
jb2.addActionListener(object);  // object must have a method, which is called when button jb2 clicked (next page)

At least 100 more methods; these are most important

JButton is in package javax.swing
A JPanel that is painted

- The JFrame content pane has a JPanel in its CENTER and a "reset" button in its SOUTH.
- The JPanel has a horizontal box b, which contains two vertical Boxes.
- Each vertical Box contains two instances of class Square.

Click a Square that has no pink circle, and a pink circle is drawn. Click a square that has a pink circle, and the pink circle disappears. Click the rest button and all pink circles disappear.

- This GUI has to listen to:
  1. a click on Button reset
  2. a click on a Square (a Box)

These are different kinds of events, and they need different listener methods

Class Graphics

An object of abstract class Graphics has methods to draw on a component (e.g. on a JPanel, or canvas).

Major methods:
- drawString("abc", 20, 30);
- drawLine(x1, y1, x2, y2);
- drawRect(x, y, width, height);
- drawOval(x, y, width, height);
- setColor(Color.red);
- getColor();
- getFont();
- setFont(Font f);

You won’t create an object of Graphics; you will be given one to use when you want to paint a component

Graphics is in package java.awt

Listen to mouse event (click, press, release, enter, leave on a component)

public interface MouseListener {
  void mouseClicked(MouseEvent e);
  void mouseEntered(MouseEvent e);
  void mouseExited(MouseEvent e);
  void mousePressed(MouseEvent e);
  void mouseReleased(MouseEvent e);
}

Having to write all of these in a class that implements MouseListener, even though you don’t want to use all of them, can be a pain. So, a class is provided that implements them in painless way.

In package java.awt.event

Listen to mouse event (click, press, release, enter, leave on a component)

public class MouseInputAdaptor implements MouseListener, MouseInputListener {
  public void mouseClicked(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
  public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}
  ... others ...
}

So, just write a subclass of MouseInputAdaptor and override only the methods appropriate for the application
A class that listens to a mouseclick in a Square

```java
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.event.*;

public class MouseDemo3 extends JFrame {
    private JButton wButt, eButt, BD3, BeLis;

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }

    public void keyTyped(KeyEvent e) {
        char typedChar = e.getKeyChar();
        capsLabel.setLabel(typedChar + typedChar.toUpperCase());
        // Other code...
    }
}
```

Listening to the keyboard

```java
import java.awt.event.KeyListener;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class MD2 extends JFrame {
    public MouseDemo2() {
        super(t); // place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

MouseDemo2

```java
public class MouseDemo2 extends JFrame {
    private JButton jb;
    private Box leftC, rightC;

    public void mouseClicked(MouseEvent e) {
        ...;
    }
}
```

Public class

```java
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt, BD3, BeLis;

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```

Have a different listener for each button

```java
public class MouseDemo2 {
    public MouseDemo2() {
        super(t);
        place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

Doesn't work!

```java
A listener for eastButt
class BeLis extends ActionListener {
    eButt, wButt
    ...;
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```

A class that listens to a mouseclick in a Square

```java
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.event.*;

public class MouseDemo2 {
    public MouseDemo2() {
        super(t);
        place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

public class MouseDemo2 extends JFrame {
    private JButton jb;
    private Box leftC, rightC;

    public void mouseClicked(MouseEvent e) {
        ...;
    }
}
```

Public class

```java
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt, BD3, BeLis;

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```

Have a different listener for each button

```java
public class MouseDemo2 {
    public MouseDemo2() {
        super(t);
        place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

Doesn't work!

```java
A listener for eastButt
class BeLis extends ActionListener {
    eButt, wButt
    ...;
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```

A class that listens to a mouseclick in a Square

```java
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.event.*;

public class MouseDemo2 {
    public MouseDemo2() {
        super(t);
        place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

Public class

```java
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt, BD3, BeLis;

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```

Have a different listener for each button

```java
public class MouseDemo2 {
    public MouseDemo2() {
        super(t);
        place components on content pane
        pack, make unresizeable, visible;
        jb.addActionListener(this);
        b00.addMouseMotionListener(this);
        b01.addMouseMotionListener(this);
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
        mouseDragged(new MouseEvent(...));
    }
}
```

Doesn't work!

```java
A listener for eastButt
class BeLis extends ActionListener {
    eButt, wButt
    ...;
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof Square) {
            ((Square) e.getSource()).complementDisk();
        }
    }
}
```
Solution to problem: Make BeListener an inner class.

```java
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt; …;
    public ButtonDemo3() { … }
    public void actionPerformed(ActionEvent e) { … }
    private class BeListener implements ActionListener {
        public void actionPerformed(ActionEvent e) { body }
    }
    // 1 object of BeListener created. Ripe for making anonymous
    public void m() { …
        eButt.addActionListener(new BeListener());
    }
}
```

Problem: can't give a function as a parameter:

```java
public void m() { …
    eButt.addActionListener(aP);
}
public void aP(ActionEvent e) { body }
```

Why not just give `eButt` the function to call? Can’t do it in Java! Can in some other languages

```java
public void m() { …
    eButt.addActionListener(new C());
}
```

Java says: provide class `C` that wraps method; give `eButt` an object of class `C`

```java
public class C implements IN {
    public void aP(ActionEvent e) { body }
}
```

Have a class for which only one object is created?

Use an anonymous class.

Use sparingly, and only when the anonymous class has 1 or 2 methods in it, because the syntax is ugly, complex, hard to understand.

```java
public class BDemo3 extends JFrame implements ActionListener {
    private JButton wButt, eButt; …;
    public ButtonDemo3() { …
        eButt.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) { body }
        });
    }
    private class BeListener implements ActionListener {
        public void actionPerformed(ActionEvent e) { body }
    }
    public void m() { …
        eButt.addActionListener(new BeListener());
    }
}
```

Making class anonymous will replace `new BeListener()`

Expression that creates object of `BeListener`

```java
public void m() { …
    eButt.addActionListener(new BeListener());
}
```

1. Write `new` 2. Use name of interface that `BeListener` implements
2. Write `new ActionListener()` 3. Put in arguments of constructor call
3. Write `new ActionEvent()` 4. Put in class body
4. Write `new ActionListener() { declarations in class }`
5. Replace `new BeListener()` by `new-expression`