

Graphical User Interface (GUI) Applications

Abstract Windowing Toolkit (AWT)
Events Handling
Applets

1

Introduction

- Java began as a language to be integrated with browsers.
- But it as evolved as a powerful language for developing stand-alone graphical applications and also server-side applications.
- Today, Java has large and powerful libraries to deal with 2D and 3D graphics and imaging, as well as the ability to build complex client-side interactive systems.
- Our focus: Simple GUI apps and Applets and Graphics. More on graphics in your 3rd year subject on "Interactive Computing".

2

AWT - Abstract Windowing Toolkit

- Single Windowing Interface on Multiple Platforms
- Supports functions common to all window systems
- Uses Underlying Native Window system
- AWT provides
 - GUI widgets
 - Event Handling
 - Containers for widgets
 - Layout managers
 - Graphic operations

3

AWT - Abstract Window Toolkit

- Portable GUI - preserves native look and feel
- Standard GUI Components (buttons..)
- Containers - Panels, Frames, Dialogs
- Graphics class for custom drawing
- Layouts responsible for actual positioning of components:
 - BorderLayout, GridLayout, FlowLayout, Null layout

4

Adding Components via Layouts



5

Building Graphical User Interfaces

- `import java.awt.*;`
- Assemble the GUI
 - use GUI components,
 - basic components (e.g., Button, TextField)
 - containers (Frame, Panel)
 - set the positioning of the components
 - use Layout Managers
- Attach events

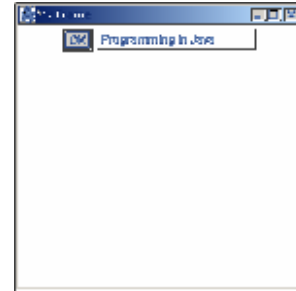
6

A sample GUI program

```
import java.awt.*;
public class MyGui
{
    public static void main(String args[] )
    {
        Frame f = new Frame ("My Frame");
        Button b = new Button("OK");
        TextField tf = new TextField("Programming in Java", 20);
        f.setLayout(new FlowLayout());
        f.add(b);
        f.add(tf);
        f.setSize(300, 300);
        f.setVisible(true);
    }
}
```

7

Output



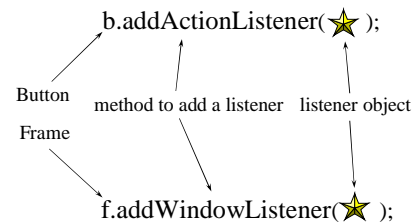
8

Events

- Each GUI component (e.g., a Button) that wishes to respond to an event type (e.g., click), must register an event handler, called a Listener.
- The listener is an object of a "Listener" interface.
- A Listener class can be created by subclassing (through "implements") one of Listener interfaces (all listener interfaces are in the java.awt.event package => must import java.awt.event.*;)
- The registration of the listener is done by a call to a method such as addActionListener(<Listener Object>). Each GUI component class has one or more such add..() methods, where applicable.

9

Events



10

Listener Interfaces in java.awt.event.*

- [1] ActionListener
- [2] ItemListener
- [3] MouseMotionListener
- [4] MouseListener
- [5] KeyListener
- [6] FocusListener
- [7] AdjustmentListener
- [8] ComponentListener
- [9] WindowListener
- [10] ContainerListener
- [11] TextListener

11

Listener Interfaces

- Each listener interface has methods that need to be implemented for handling different kinds of events.
- For example 1, the ActionListener interface has a method actionPerformed() button component is operated.
- For example2, the MouseMotionListener interface has two methods:
 - 1) mouseDragged(MouseEvent) - Invoked when a mouse button is pressed on a component and then dragged.
 - 2) mouseMoved(MouseEvent) - Invoked when the mouse button has been moved on a component (with no buttons down).

12

Implementing the ActionListener Interface and attaching an event handler to a button

```
import java.awt.*;
import java.awt.event.*;
public class MyGui1
{
    public static void main(String args[] )
    {
        Frame f = new Frame ("My Frame");
        MyGuiAction ga = new MyGuiAction(f);
    }
}
class MyGuiAction implements ActionListener
{
    static int count = 0;
    Button b;
    TextField t;
    MyGuiAction(Frame f)
    {
        b = new Button("OK");
        b.addActionListener(this);
        if = new TextField("Hello Java", 20);
        f.setLayout(new FlowLayout());
        f.add(b);
        f.add(t);
        f.setSize(300, 300);
        f.setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == b)
        {
            count++;
            System.out.println("Button is Pressed");
            t.setText("Hello Java Click "+count);
        }
    }
}
```

13

Output and Clicks on "OK" Button



Exec started

1st click on OK button

2nd click on OK button



14

BorderLayout Example

```
import java.awt.*;
public class MyGui2
{
    public static void main(String args[] )
    {
        Frame f = new Frame ("My Frame");
        f.setLayout(new BorderLayout());
        // Add text field to top
        f.add("North",new TextField());
        // Create the panel with buttons at the bottom...
        Panel p = new Panel(); // FlowLayout
        p.add(new Button("OK"));
        p.add(new Button("Cancel"));
        f.add("South",p);
        f.add("Center", new TextField("Center region"));
        f.setSize(300, 300);
        f.setVisible(true);
    }
}
```

15

Output



16