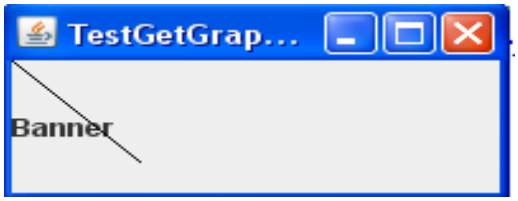
CS112-Section2
Hakan Guldas
Burcin Ozcan
Meltem Kaya
Muge Celiktas
Notes of 6-8 May

Graphics

- Previously, we have seen GUI components, their relationships, containers, layout managers.
- Now we will see how to paint graphics on GUI components

```
import javax.swing.*;

import java.awt.Graphics;


public class TestGetGraphics extends JFrame {

        private JLabel jlblBanner = new JLabel("Banner");


    public TestGetGraphics() {

        add(jlblBanner);

        System.out.println(jlblBanner.getGraphics());

    }


    public static void main(String[] args) {

        TestGetGraphics frame = new TestGetGraphics();

        frame.setTitle("TestGetGraphics");

        frame.setLocationRelativeTo(null); //Center the frame

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(200,100);

        frame.setVisible(true);

        JOptionPane.showMessageDialog(null,"Delay on purpose\nClick OK to dismiss the dialog");
```

```
        Graphics graphics = frame.jlblBanner.getGraphics();
        graphics.drawLine(0,0,50,50);
    }

}
```



In this example if you resize the frame, line disappears, so how can we avoid this problem and display line permanently? -by inheriting Jlabel to draw line!

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

public class TestPaintComponent extends JFrame {

    public TestPaintComponent() {
        add(new NewLabel("Banner"));
    }

    public static void main(String[] args) {
        TestPaintComponent frame = new TestPaintComponent();
        frame.setTitle("TestPaintComponent");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200,100);
        frame.setVisible(true);
    }
```

```
}
class NewLabel extends JLabel {
        public NewLabel(String text) {
                super(text);
        }

        protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawLine(0,0,50,50);
        }
}
```

--The paintComponent method is invoked, whenever a component is first displayed or redisplayed!

--To draw things on component you first declare a class that extends a swing GUI and override paintComponent method

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

public class SmileyFace extends JFrame{

   public SmileyFace() {
        setTitle("Smiley Face");
        setSize(250,220);
        getContentPane().setBackground(Color.yellow);
   }
```

```
    public void paint(Graphics g) {
        super.paint(g); //call the paint method of the superclass,jFrame
        g.setColor(Color.red);
        g.drawOval(85,75,75,75); //the face
        g.setColor(Color.blue);
        g.drawOval(100,95,10,10); //the left eye
        g.drawOval(135,95,10,10); //the right eye
        g.drawArc(102,115,40,25,0,-180); //the mouth
        g.drawString("Smiley Face",90,175);
    }


}
```

The tester of SmileyFace:

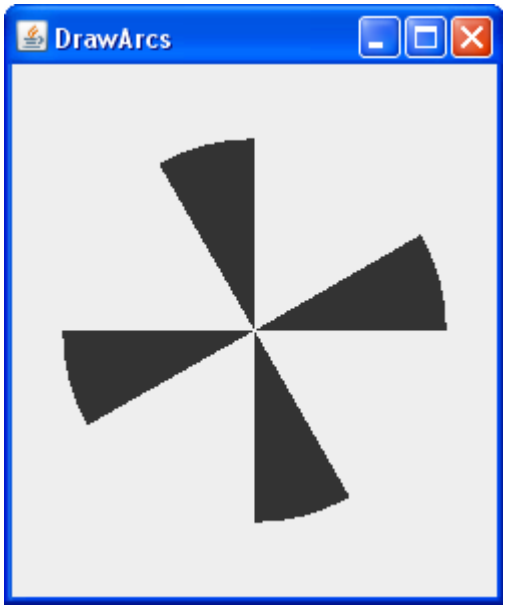
```
public static void main(String[] args) {
        SmileyFace smiley = new SmileyFace();
        smiley.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        smiley.setLocation(300,300);
        smiley.setVisible(true);
    }
```

Drawing Strings, Lines, Rectangles, and Ovals

Drawing arcs

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

public class DrawArcs extends JFrame{

   public DrawArcs() {
        setTitle("DrawArcs");
        add(new ArcsPanel());
   }

   /** Main method*/
   public static void main(String[] args) {
        DrawArcs frame = new DrawArcs();
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(250,300);
        frame.setVisible(true);
   }
}
// The class for drawing arcs on panel
class ArcsPanel extends JPanel{
        //Draw four blazes of a fan
        protected void paintComponent(Graphics g) {
                super.paintComponent(g);
```

```
            int xCenter = getWidth()/2;
            int yCenter = getHeight()/2;
            int radius = (int) (Math.min(getWidth(),getHeight())*0.4);

            int x = xCenter -radius;
            int y = yCenter - radius;
            g.fillArc(x,y,2*radius,1*radius,0,30);
            g.fillArc(x,y,2*radius,1*radius,90,30);
            g.fillArc(x,y,2*radius,1*radius,180,30);
            g.fillArc(x,y,2*radius,1*radius,270,30);
        }
}
```

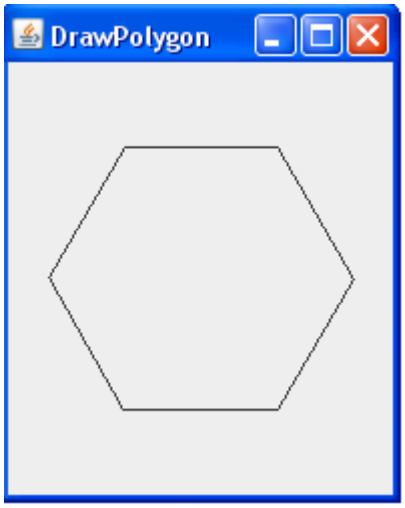


Draw Polygons

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

public class DrawPolygon extends JFrame{
```

```
    public DrawPolygon() {
        setTitle("DrawPolygon");
        add(new PolygonsPanel());
    }
    public static void main(String[] args) {
        DrawPolygon frame = new DrawPolygon();
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200,250);
        frame.setVisible(true);
    }
}
class PolygonsPanel extends JPanel {
        //Draw a polygon in the panel
        protected void paintComponent(Graphics g) {
                super.paintComponent(g);

                int xCenter = getWidth()/2;
                int yCenter = getHeight()/2;
                int radius = (int) (Math.min(getWidth(),getHeight())*0.4);
                Polygon polygon = new Polygon();
                //Add points to polygon
                polygon.addPoint(xCenter + radius,yCenter);
                polygon.addPoint((int) (xCenter + radius * Math.cos(2*Math.PI/6)),(int)
(yCenter - radius * Math.sin(2*Math.PI/6)));
                polygon.addPoint((int) (xCenter + radius *
Math.cos(2*2*Math.PI/6)),(int) (yCenter - radius * Math.sin(2*2*Math.PI/6)));
                polygon.addPoint((int) (xCenter + radius *
Math.cos(3*2*Math.PI/6)),(int) (yCenter - radius * Math.sin(3*2*Math.PI/6)));
```

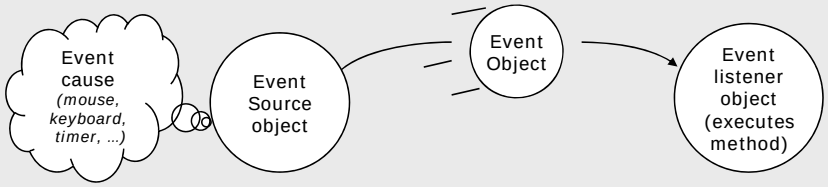
```
            polygon.addPoint((int) (xCenter + radius *
Math.cos(4*2*Math.PI/6)),(int) (yCenter - radius * Math.sin(4*2*Math.PI/6)));

            polygon.addPoint((int) (xCenter + radius *
Math.cos(5*2*Math.PI/6)),(int) (yCenter - radius * Math.sin(5*2*Math.PI/6)));

            g.drawPolygon(polygon);//Draw the polygon


        }

}
```



Event Driven Programming

- In ED programming, code is executed when an event occurs
- The program interacts with user and the events drive its execution.
- Event is defined as a signal to the program that something has happened.
- The component which an event is generated is called source object or source component. E.g., button

Events & Event Handling



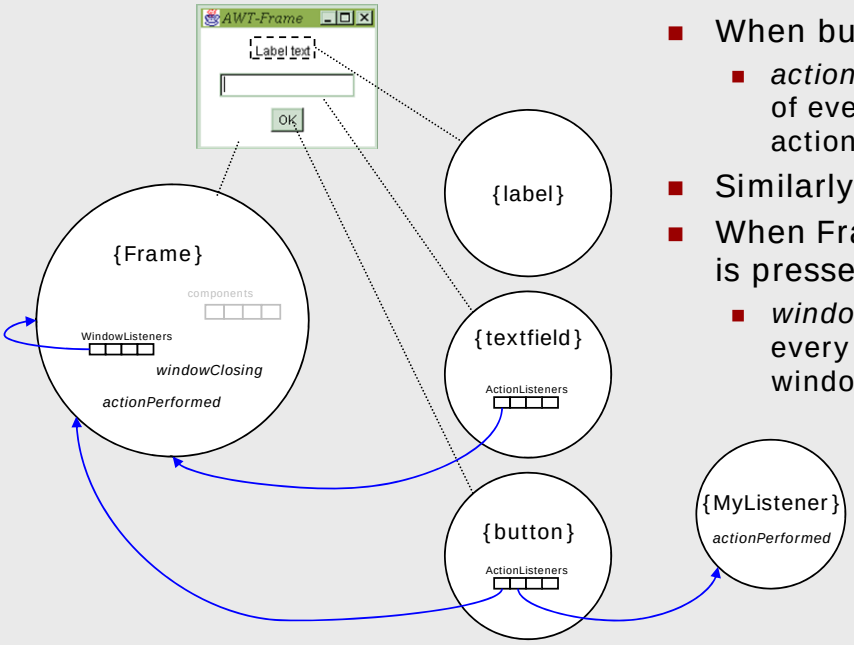
- Example...
  - User clicks on a button
  - Button is source of event object
  - Event object passed to associated listener object
  - Listener object executes associated method to perform desired task (save file, quit program, ...)

Setting up Event Handling

- Create listener class
  - Using new or existing class, *simply*
  - Implement desired event listener interface
  - Putting code for desired action in its methods
- In application *(e.g. Frame)*
  - Create instance of listener class
  - Add as listener of source object

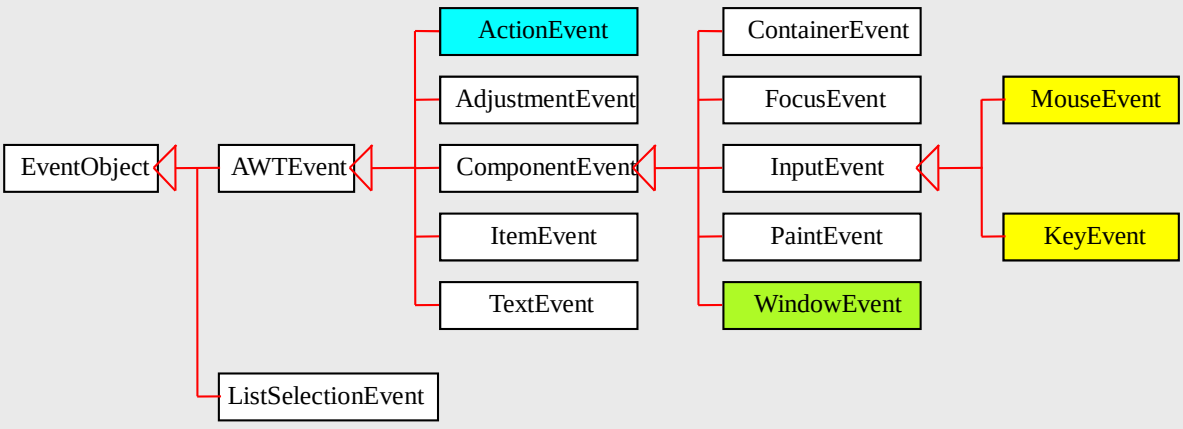
• can have any number of listeners for each event
• Source & listener can be same object!

Understanding Events



- When button is pressed
  - *actionPerformed* method of every item in button's actionListeners list called
- Similarly for textfield
- When Frame close button is pressed
  - *windowClosing* method of every item in Frame's windowListeners list called

Event Classes



How to Implement an Event Handler

- In the declaration for the event handler class, specify that the class either implements a listener interface or extends a class that implements a listener interface. For example:

  public class MyClass implements ActionListener {
- Register an instance of the event handler class as a listener upon one or more components. For example: someComponent.addActionListener(instanceOfMyClass)
- Implement the methods in the listener interface. For example:

  public void actionPerformed(ActionEvent e) { *...//code that reacts to the action...* }

Simple example :

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


public class SimpleEventDemo extends JFrame{

   public SimpleEventDemo() {

        JButton jbtOK = new JButton("OK");
```

```
        setLayout(new FlowLayout());
        add(jbtOK);

        ActionListener listener = new OKListener();
        jbtOK.addActionListener(listener);
    }
    /** Main method*/
    public static void main(String[] args) {
        JFrame frame = new SimpleEventDemo();
        frame.setTitle("SimpleEventDemo");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200,250);
        frame.setVisible(true);
    }
}
class OKListener implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                System.out.println("It is OK");
        }
}
```

Simple example with inner class :

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

public class SimpleEventDemoInnerClass extends JFrame{
```

```
    public SimpleEventDemoInnerClass() {
        JButton jbtOK = new JButton("OK");
        setLayout(new FlowLayout());
        add(jbtOK);

        ActionListener listener = new OKListener();
        jbtOK.addActionListener(listener);
    }
    /** Main method*/
    public static void main(String[] args) {
        JFrame frame = new SimpleEventDemo();
        frame.setTitle("SimpleEventDemo");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(100,80);
        frame.setVisible(true);
    }
    private class OKListener implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            System.out.println("It is OK");
            }
    }
}
```

Another inner class example :

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

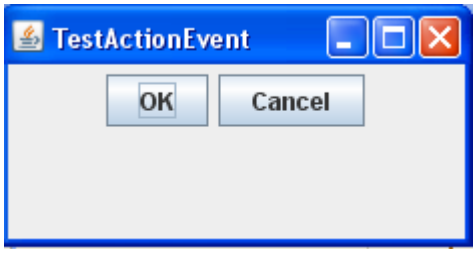
```
public class SimpleEventDemoAnonymousInnerClass extends JFrame{

   public SimpleEventDemoAnonymousInnerClass() {
        JButton jbtOK = new JButton("OK");
        setLayout(new FlowLayout());
        add(jbtOK);

        //Create and register anonymous inner class listener
        jbtOK.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        System.out.println("It is OK");
                }
        });
   }
   /** Main method*/
   public static void main(String[] args) {
        JFrame frame = new SimpleEventDemo();
        frame.setTitle("SimpleEventDemo");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(100,80);
        frame.setVisible(true);
   }
}
```

Another example :

```
import javax.swing.*;
```

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

public class TestActionEvent extends JFrame{
        private JButton jbtOK = new JButton("OK");
        private JButton jbtCancel = new JButton("Cancel");
   public TestActionEvent() {
        setLayout(new FlowLayout());

        add(jbtOK);
        add(jbtCancel);

        jbtOK.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        System.out.println("The"+e.getActionCommand()+"button"+"is
clicked at\n"+new java.util.Date(e.getWhen()));
                }
        });

        jbtCancel.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        System.out.println("The"+e.getActionCommand()+"button"+"is
clicked at\n"+new java.util.Date(e.getWhen()));
                }
        });
   }
   /** Main method*/
   public static void main(String[] args) {
```

```
        JFrame frame = new TestActionEvent();
        frame.setTitle("TestActionEvent");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(100,80);
        frame.setVisible(true);
    }
}
```

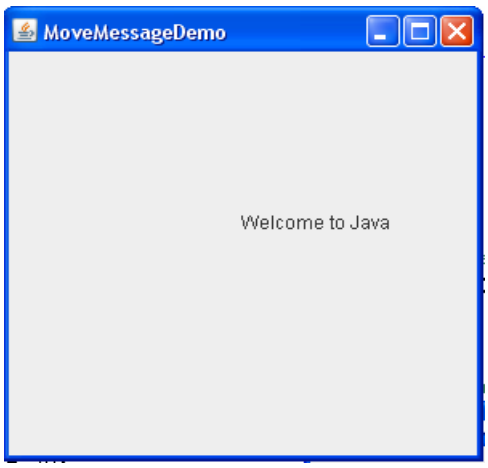


Mouse Example :

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

public class MoveMessageDemo extends JFrame{
    public MoveMessageDemo() {
        //Create MovableMessagePanel instance for moving a message
        MovableMessagePanel p = new MovableMessagePanel("Welcome to Java");
        //Place the message panel in the frame
        getContentPane().setLayout(new BorderLayout());
        getContentPane().add(p);
    }
    public static void main(String[] args) {
        MoveMessageDemo frame = new MoveMessageDemo();
```

```
        frame.setTitle("MoveMessageDemo");
        frame.setLocationRelativeTo(null);//Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300,300);
        frame.setVisible(true);
    }
    static class MovableMessagePanel extends JPanel {// Inner class:
MovableMessagePanel draws a message
        private String message = "Welcome to Java";
        private int x = 20;
        private int y = 20;
        /** Construct a panel to draw string s */
        public MovableMessagePanel(String s) {
            message = s;
            this.addMouseMotionListener(new MouseMotionAdapter() {
                /** Handle mouse dragged event */
                public void mouseDragged(MouseEvent e) {// Get the new
location and repaint the screen
                    x = e.getX();
                    y = e.getY();
                    repaint();
                }
            });
        }
        protected void paintComponent(Graphics g) {/** Paint the component */
        super.paintComponent(g);
        g.drawString(message,x,y);
        }
```

```
    }
}
```



Questions:

Question 1: Give examples to containers,components and helpers and explain
The relation between them?
Ans:
Containers:contain other GUI components. E.g, Window, Panel, Applet, Frame Dialog.
Components:Buttons, ComboBox, List, Slider, Label etc.
Helpers:Font, Color, Dimension, LayoutManager etc.
We can add components,helpers and containers to frame
Question 2: What is event and source component?
Ans:
Event is defined as a signal to the program that something has happened.
The component which an event is generated is called source object or source component. E.g., button
Question 3: Why we use an inner class?
Ans:
Inner class is declared when it is only used by outer class
And inner class can use the reference data and methods in outer class.
Question 4: Can we use ActionListener to make listener for JSlider?
Ans:
No. Since it changes. We must use ChangeListener. For example we use MouseListener for MouseAdapter.