Programming in Java

Dr. Nyein Aye Maung Maung
Dr. Eng (Ritsumeikan University, Japan)
Lecturer
Computer Engineering and Information Technology Dept.
Yangon Technological University

Course Schedule

Week	Topics
Week 1	Overview of JAVA, Data Types, Variables and Arrays
Week 2	Operators and Control Statements
Week 3	Classes and A closer look at Methods and Classes
Week 4	Inheritance, Polymorphism, Abstraction and Encapsulation
Week 5	Packages and Interfaces
Week 6	Exception Handling and Multi-threaded Programming
Week 7	String Handling
Week 8	Exploring java.lang and More utilities classes
Week 9	Java Collections Framework
Week 10	Java I/O
Week 11	Basic Graphical User Interface
Week 12	Event Handling
Week 13	Database Programming
Week 14	Applet and Networking

Lecture 12

Event Handling/ Even-Driven Programming

Outline of Class

- Graphic Class
- Event Driven Programming

Topic 1

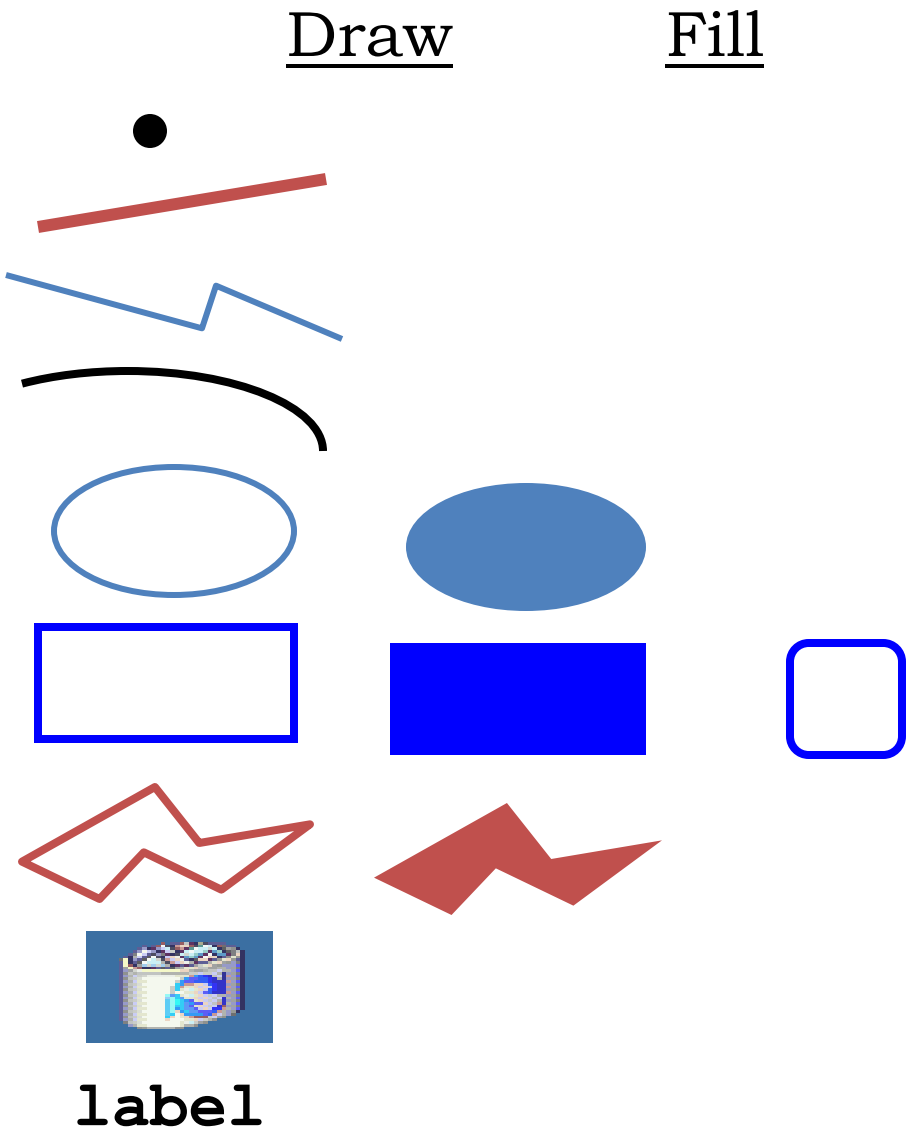
Graphics Class

Graphics Class

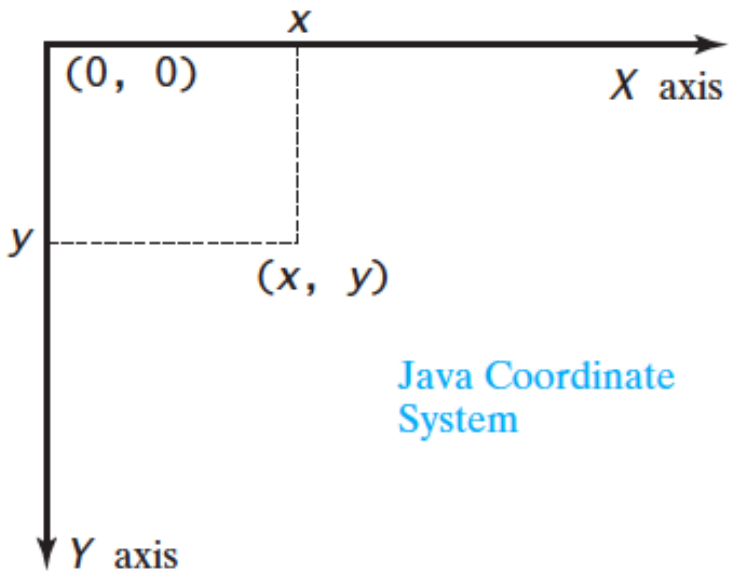
- Provides methods for drawing Strings, Lines, Rectangles, Ovals, Arcs, Polygons and Poly Lines
  - GUI Component → A piece of paper
  - Graphic Object → Pencil or Brush
- Apply methods in Graphics class to draw graphics on a GUI Component
- To draw a component
  - Define **a class that extends JPanel**
  - Override **paintComponent** method to specify what to draw
  - Invoke **repaint( )** method to call paintComponent method to refresh the viewing area

Graphics Primitives

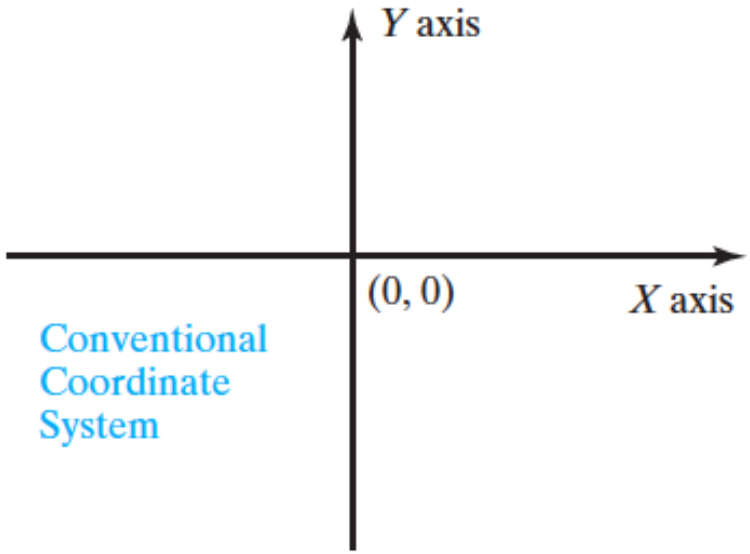
- Point (x,y)
- Line (pt1,pt2)
- PolyLine (pt list)
- Arc
- Oval (pt, w,h)
- Rectangle (pt, w,h)
- RoundRectangle
- Polygon (pt list)
- Image (file, x,y)
- Text (string, x,y)



Coordinate System



Java Coordinate System



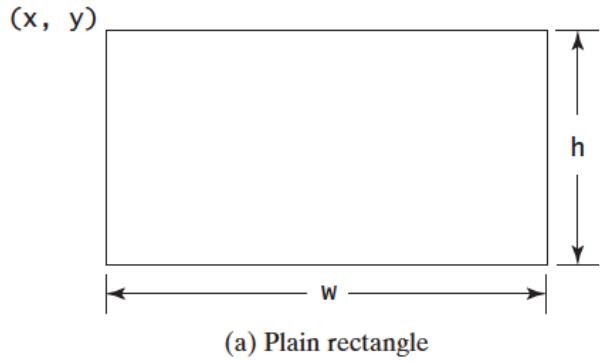
Conventional Coordinate System

- Methods for Graphics Object g

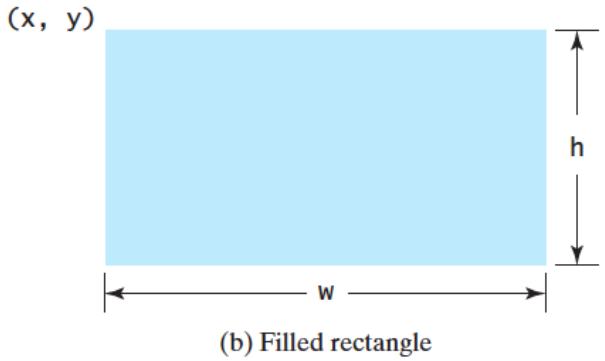
void drawString(s: String, x: int, y: int)
void drawLine(x1: int, y1: int, x2: int, y2: int)

```
void drawRect(int x, int y, int width, int height);
void drawRoundRect(int x, int y, int w, int h,
                   int arcWidth, int arcHeight);
void drawOval(int x, int y, int width, int height);
void fillRect(int x, int y, int width, int height);
void fillRoundRect(int x, int y, int w, int h,
                   int arcWidth, int arcHeight);
void fillOval(int x, int y, int width, int height);
void drawString(String text, int x, int y);
void drawLine(int x1, int y1, int x2, int y2);
void draw3DRect(int x, int y, int w, int h, boolean raised);
void fill3DRect(int x, int y, int w, int h, boolean raised);
void drawArc(int x, int y, int w, int h,
                   int startAngle, int arcAngle);
void fillArc(int x, int y, int w, int h,
                   int startAngle, int arcAngle);
void setColor(Color c);
void setFont(Font f);
```

Drawing Shapes



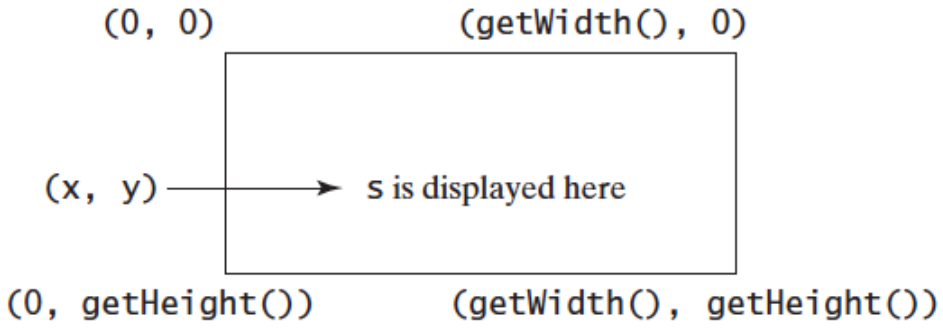
(a) Plain rectangle

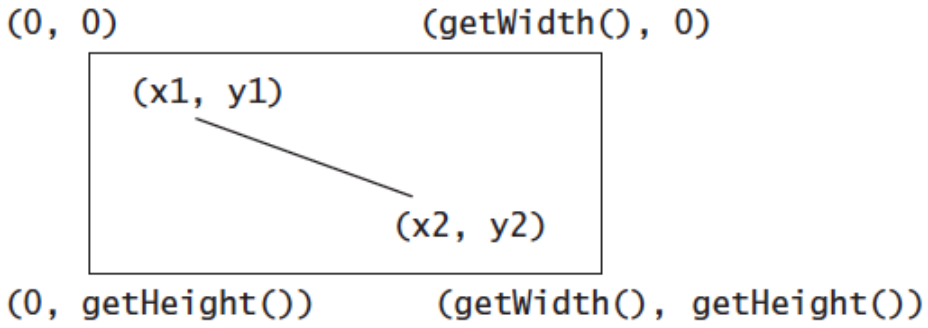


(b) Filled rectangle

drawRect(x, y, w, h)

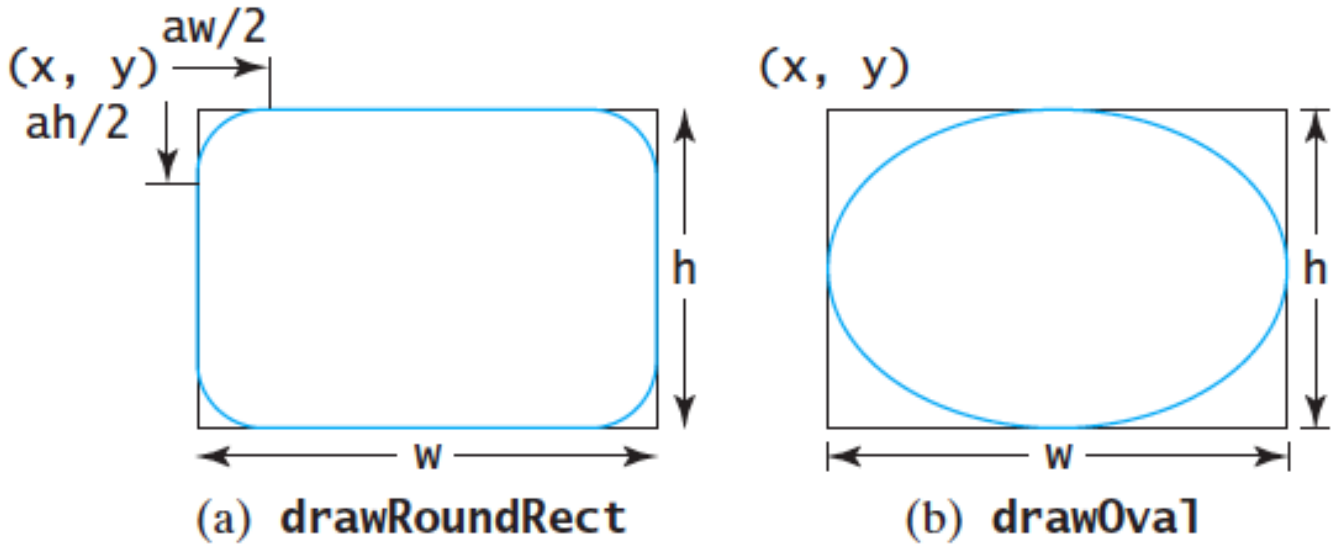
fillRect(x, y, w, h)





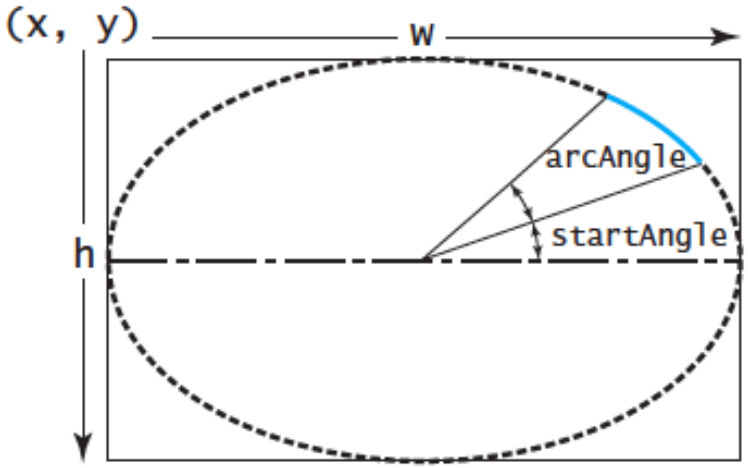
drawString(s, x, y)

drawLine(x1, y1, x2, y2)



drawRoundRect(x, y, w, h, aw, ah)

drawOval(x, y, w, h)



drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)
fillArc(int x, int y, int w, int h, int startAngle, int arcAngle)

- **Example 1**: Paint a line and paint a string on Panel

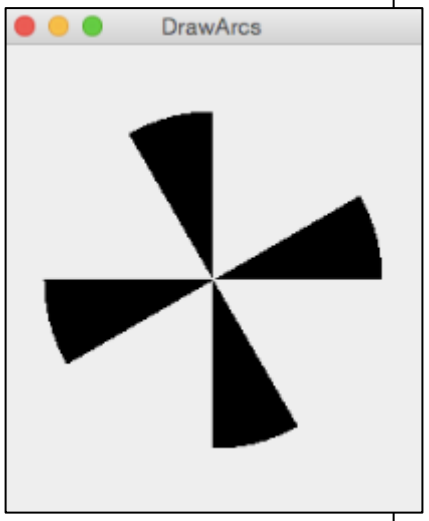
① Create frame
② Add Panel to Frame
③ Design a class that extends Panel to paint preferred components
④ Override paintComponent method to draw a string and a line

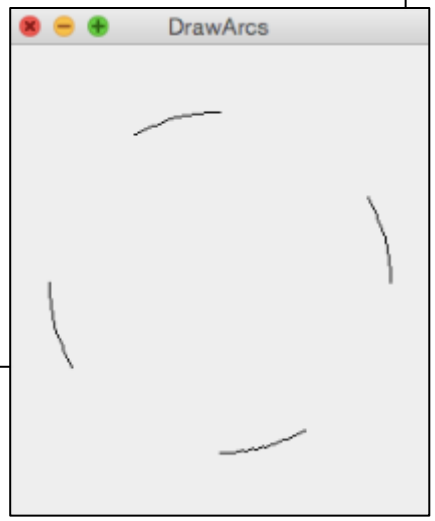
```
import javax.swing.*;
import java.awt.Graphics;

public class TestPaintComponent extends JFrame {
  public TestPaintComponent() {
②   add(new NewPanel());
  }
  public static void main(String[] args) {
①   TestPaintComponent frame = new TestPaintComponent();
    frame.setTitle("TestPaintComponent");
    frame.setSize(200, 100);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
③ class NewPanel extends JPanel {
  @Override
④ protected void paintComponent(Graphics g) {
    g.drawLine(0, 0, 50, 50);
    g.drawString("Banner", 0, 40);
  }
}
```

- **Example 2**: Demo for drawing Arcs

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;
public class DrawArc extends JFrame {
  public DrawArc() {
    setTitle("DrawArcs");
    add(new ArcsPanel());
  }
  /** Main method */
  public static void main(String[] args) {
    DrawArc frame = new DrawArc();
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(250, 300);
    frame.setVisible(true);
  }
}
```

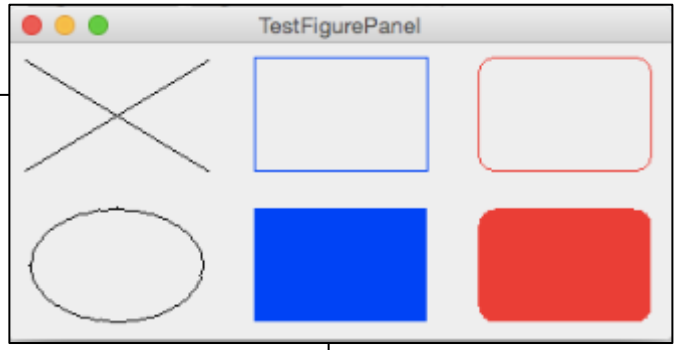
```
// The class for drawing arcs on a panel
class ArcsPanel extends JPanel {
  @Override
  protected void paintComponent(Graphics g) {
    int xCenter = getWidth() / 2;
    int yCenter = getHeight() / 2;
    int radius = (int)(Math.min(getWidth(), getHeight()) * 0.4);

    int x = xCenter - radius;
    int y = yCenter - radius;

    g.fillArc(x, y, 2 * radius, 2 * radius, 0, 30);
    g.fillArc(x, y, 2 * radius, 2 * radius, 90, 30);
    g.fillArc(x, y, 2 * radius, 2 * radius, 180, 30);
    g.fillArc(x, y, 2 * radius, 2 * radius, 270, 30);
  }
}
```





g.drawArc →

- **Example 3**: Draw Multiple Objects on a frame



add 6 panels to frame

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

public class TestFigurePanel extends JFrame {
  public TestFigurePanel() {
    setLayout(new GridLayout(2, 3, 5, 5));
    add(new FigurePanel(FigurePanel.LINE));
    add(new FigurePanel(FigurePanel.RECTANGLE));
    add(new FigurePanel(FigurePanel.ROUND_RECTANGLE));
    add(new FigurePanel(FigurePanel.OVAL));
    add(new FigurePanel(FigurePanel.RECTANGLE, true));
    add(new FigurePanel(FigurePanel.ROUND_RECTANGLE, true));
  }

  public static void main(String[] args) {
    TestFigurePanel frame = new TestFigurePanel();
    frame.setSize(400, 200);
    frame.setTitle("TestFigurePanel");
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

FigurePanel class to draw components

```
import java.awt.*;
import javax.swing.JPanel;
public class FigurePanel extends JPanel {
  // Define constants
  public static final int LINE = 1;
  public static final int RECTANGLE = 2;
  public static final int ROUND_RECTANGLE = 3;
  public static final int OVAL = 4;
  private int type = 1;
  private boolean filled = false;
  /** Construct a default FigurePanel */
  public FigurePanel() {
  }
  /** Construct a FigurePanel with the specified type */
  public FigurePanel(int type) {
    this.type = type;

  }
  /** Construct a FigurePanel with the specified type and filled */
  public FigurePanel(int type, boolean filled) {
    this.type = type;
    this.filled = filled;

  }
```

new FigurePanel(FigurePanel.LINE))
new FigurePanel(FigurePanel.RECTANGLE))
new FigurePanel(FigurePanel.ROUND_RECTANGLE))
new FigurePanel(FigurePanel.OVAL))

new FigurePanel(FigurePanel.RECTANGLE),true)
new FigurePanel(FigurePanel.ROUND_RECTANGLE),true)

```
@Override // Draw a figure on the panel
protected void paintComponent(Graphics g) {
  // Get the appropriate size for the figure
  int width = getWidth();
  int height = getHeight();
  switch (type) {
    case LINE: // Display two cross lines
      g.setColor(Color.BLACK);
      g.drawLine(10, 10, width - 10, height - 10);
      g.drawLine(width - 10, 10, 10, height - 10);
      break;
    case RECTANGLE: // Display a rectangle
      g.setColor(Color.BLUE);
      if (filled)
        g.fillRect((int)(0.1 * width), (int)(0.1 * height),
          (int)(0.8 * width), (int)(0.8 * height));
      else
        g.drawRect((int)(0.1 * width), (int)(0.1 * height),
          (int)(0.8 * width), (int)(0.8 * height));
      break;
```

```
      case ROUND_RECTANGLE: // Display a round-cornered rectangle
        g.setColor(Color.RED);
        if (filled)
          g.fillRoundRect((int)(0.1 * width), (int)(0.1 * height),
            (int)(0.8 * width), (int)(0.8 * height), 20, 20);
        else
          g.drawRoundRect((int)(0.1 * width), (int)(0.1 * height),
            (int)(0.8 * width), (int)(0.8 * height), 20, 20);
        break;
      case OVAL: // Display an oval
        g.setColor(Color.BLACK);
        if (filled)
          g.fillOval((int)(0.1 * width), (int)(0.1 * height),
            (int)(0.8 * width), (int)(0.8 * height));
        else
          g.drawOval((int)(0.1 * width), (int)(0.1 * height),
            (int)(0.8 * width), (int)(0.8 * height));
    }
  }
```

Topic 2

Event Driven Programming

Making an Interactive GUI Program

To make an interactive GUI program, you need:

Components (Event Source)

buttons, windows, menus, etc.

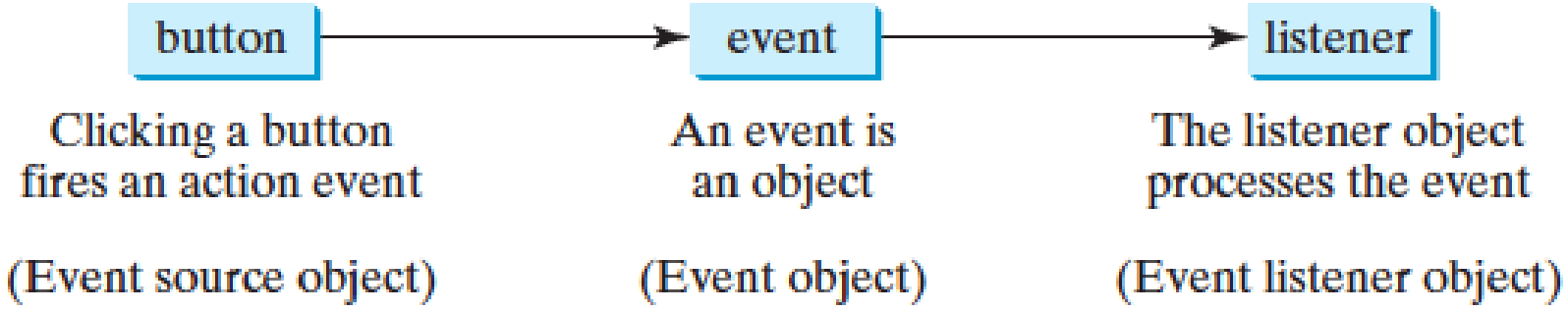
Events

mouse clicked, window closed, button clicked, etc.

Event listeners (interfaces) and event handlers (methods)

listen for events to be trigged, and then perform actions to handle them

Event Driven Programming



- Event Source
  - The component that creates an event and fires it is called the event source object
  - eg. button, check box, etc.
- Event
  - An event is an object created from an event source that represents a user's interaction with the event source;
  - eg. mouse click on button
  - Firing an event means to create an event and delegate the listener to handle the event

- Listeners, Registrations, and Handling Events
  - A listener is an object that must be registered with an event source object, and it must be an instance of an appropriate event-handling interface
  - eg. XListener for Xevent
    - ActionListener for ActionEvent
    - ItemListener for ItemEvent
    - MouseListener for MouseEvent
    - KeyListener for KeyEvent
- Registering Lister to the event source
  - EventSource.add**X**Listerner for **X**event
- Handling Events
  - The listener interface contains the method(s), known as the event handler(s) , for processing the event
    - eg. actionPerformed for ActionListener
    - itemStateChanged for ItemListener

User Action, Source Object, Event Type, Listener Interface, and Handler

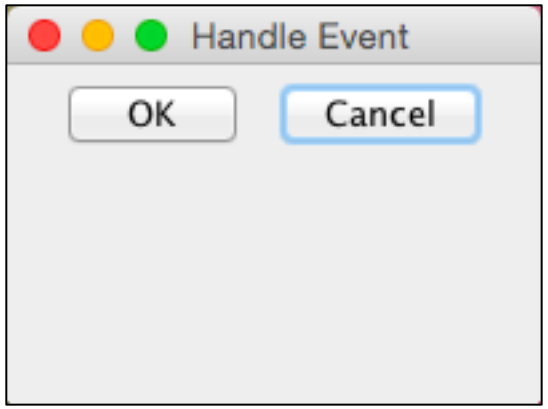
Event Source	Action	Event type	Listener Interface	Listener Interface Methods
JButton	Click	ActionEvent	ActionListener	actionPerformed (ActionEvent e)
JTextField	Press enter in txt field	ActionEvent	ActionListener	actionPerformed (ActionEvent e)
JComboBox	Select a new item	ActionEvent ItemEvent	ActionListener ItemListener	actionPerformed itemStateChanged
JRadioButton	Check or uncheck	ActionEvent ItemEvent	ActionListener ItemListener	actionPerformed itemStateChanged
JCheckBox	Check or uncheck	ActionEvent ItemEvent	ActionListener ItemListener	actionPerformed itemStateChanged

User Action, Source Object, Event Type, Listener Interface, and Handler

Event Source	Action	Event type	Listener Interface	Listener Interface Methods
Component	Mouse pressed	MouseEvent	MouseListener	mousePressed
	Mouse released		MouseListener	mouseReleased
	Mouse clicked		MouseListener	mouseClicked
	Mouse entered		MouseListener	mouseEntered
	Mouse exited		MouseListener	mouseExited
	Mouse moved		MouseMotionListener	mouseMoved
	Mouse dragged		MouseMotionListener	mouseDragged
Component	Key pressed	KeyEvent	KeyListener	KeyPressed
	Key released			KeyReleased
	Key typed			KeyTyped

- Example 1: A program that listens event on button.

```
import javax.swing.*;
import java.awt.event.*;
public class HandleEvent extends JFrame {
  public HandleEvent() {
    // Create two buttons
    JButton jbtOK = new JButton("OK");
    JButton jbtCancel = new JButton("Cancel");
    // Create a panel to hold buttons
    JPanel panel = new JPanel();
    panel.add(jbtOK);
    panel.add(jbtCancel);
    add(panel); // Add panel to the frame
    // Register listeners
    OKListenerClass listener1 = new OKListenerClass();
    CancelListenerClass listener2 = new CancelListenerClass();
    jbtOK.addActionListener(listener1);
    jbtCancel.addActionListener(listener2);
  }
  public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    frame.setTitle("Handle Event");
    frame.setSize(200, 150);
    frame.setLocation(200, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```



```
class OKListenerClass implements ActionListener {

  public void actionPerformed(ActionEvent e) {

    System.out.println("OK button clicked");

  }

}
class CancelListenerClass implements ActionListener {

  public void actionPerformed(ActionEvent e) {

     System.out.println ("Cancel button clicked");

  }

}
```

- Alternative Ways of Defining Listener Classes (1)

```
import javax.swing.*;
import java.awt.event.*;
public class HandleEvent extends JFrame {
  public HandleEvent() {
    // Create two buttons
          JButton jbtOK = new JButton("OK");
          JButton jbtCancel = new JButton("Cancel");
    // Create a panel to hold buttons
    JPanel panel = new JPanel();
    panel.add(jbtOK);
    panel.add(jbtCancel);
    add(panel); // Add panel to the frame
    // Register listeners
    jbtOK.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e) {
                System.out.println("OK button clicked");
              }
    });
    jbtCancel.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e) {

                System.out.println("Cancel button clicked");
          }
    });
  }
```

```
public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    frame.setTitle("Handle Event");
    frame.setSize(200, 150);
    frame.setLocation(200, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

- Alternative Ways of Defining Listener

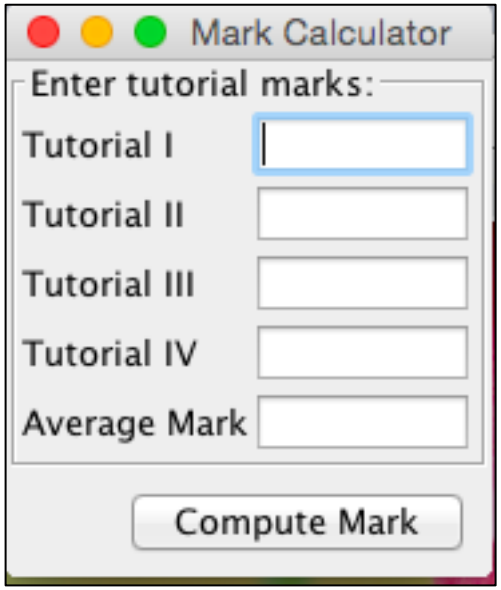
```
import javax.swing.*;
import java.awt.event.*;
public class HandleEvent extends JFrame {
   private JButton jbtOK = new JButton("OK");
   private JButton jbtCancel = new JButton("Cancel");
  public HandleEvent() {
   // Create a panel to hold buttons
   JPanel panel = new JPanel();
   panel.add(jbtOK);
   panel.add(jbtCancel);
   add(panel); // Add panel to the frame
   // Register listeners
   ButtonListener listener = new ButtonListener();
   jbtOK.addActionListener(listener);
   jbtCancel.addActionListener(listener);
  }
  //Implement Listener Class as Inner Class
  class ButtonListener implements ActionListener {

            public void actionPerformed(ActionEvent e) {
                    if (e.getSource() == jbtOK)
                    System.out.println("OK button clicked");
                    else
                System.out.println("Cancel button clicked");
            }
          }
```

return event source object

```
public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    frame.setTitle("Handle Event");
    frame.setSize(200, 150);
    frame.setLocation(200, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

Exercises on Event Driven Programming

- **Exercise 1**. Write a GUI application program that lets the user enter four tutorial marks and find average by clicking Compute Mark button as shown. Result should be displayed in Average Mark Box.



```
import java.awt.*;
public class MarkCalculator extends JFrame {
  private JTextField tutorial1 = new JTextField();
  private JTextField tutorial2 = new JTextField();
  private JTextField tutorial3 = new JTextField();
  private JTextField tutorial4 = new JTextField();
  private JTextField averageMark = new JTextField();
  private JButton jbtComputeMark = new JButton("Compute Mark");
  public MarkCalculator() {
    // Panel p1 to hold labels and text fields
    JPanel p1 = new JPanel(new GridLayout(5, 2));
    p1.add(new JLabel("Tutorial I"));
    p1.add(tutorial1);
    p1.add(new JLabel("Tutorial II"));
    p1.add(tutorial2);
    p1.add(new JLabel("Tutorial III"));
    p1.add(tutorial3);
    p1.add(new JLabel("Tutorial IV"));
    p1.add(tutorial4);
    p1.add(new JLabel("Average Mark"));
    p1.add(averageMark);
    p1.setBorder(new
      TitledBorder("Enter tutorial marks:"));
    // Panel p2 to hold the button
    JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
    p2.add(jbtComputeMark);
    // Add the panels to the frame
    add(p1, BorderLayout.CENTER);
    add(p2, BorderLayout.SOUTH);
    // Register listener
    jbtComputeMark.addActionListener(new ButtonListener());
  }
```

```
  /** Handle the Compute Mark button */
  private class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      // Get values from text fields
      int t1 = Integer.parseInt(tutorial1.getText());
      int t2 = Integer.parseInt(tutorial2.getText());
      int t3 = Integer.parseInt(tutorial3.getText());
      int t4 = Integer.parseInt(tutorial4.getText());
      double avg=(t1+t2+t3+t4)/4;
      // Display monthly payment and total payment
      averageMark.setText(String.valueOf(avg));
    }
  }
  public static void main(String[] args) {
    MarkCalculator frame = new MarkCalculator();
    frame.pack();
    frame.setTitle("Mark Calculator");
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

Events for JCheckBox, JRadioButton, and JTextField

- Exercise 2: Write a program that displays a label and allows the user to set the colors of the text in the label using radio buttons, set fonts using check boxes, and set new text entered from a text field, as shown in Figure.

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

public class GUIEventDemo2 extends JFrame {
  private JLabel jlblMessage = new JLabel("Hello", JLabel.CENTER);

  // Create check boxes to set the font for the message
  private JCheckBox jchkBold = new JCheckBox("Bold");
  private JCheckBox jchkItalic = new JCheckBox("Italic");

  // Create three radio buttons to set message colors
  private JRadioButton jrbRed = new JRadioButton("Red");
  private JRadioButton jrbGreen = new JRadioButton("Green");
  private JRadioButton jrbBlue = new JRadioButton("Blue");

  // Create a text field for setting a new message
  private JTextField jtfMessage = new JTextField(10);

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

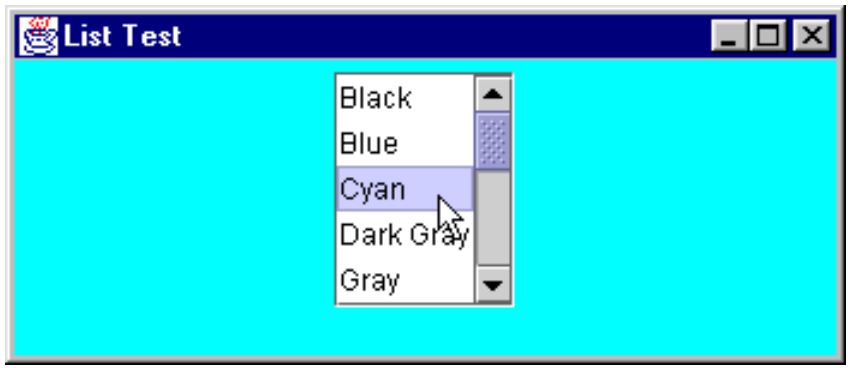
```
public GUIEventDemo2() {
    jlblMessage.setBorder(new LineBorder(Color.BLACK, 2));
    add(jlblMessage, BorderLayout.CENTER);
    // Create a new panel to hold check boxes
    JPanel jpCheckBoxes = new JPanel();
    jpCheckBoxes.setLayout(new GridLayout(2, 1));
    jpCheckBoxes.add(jchkBold);
    jpCheckBoxes.add(jchkItalic);
    add(jpCheckBoxes, BorderLayout.EAST);
    // Create a new panel to hold check boxes
    JPanel jpRadioButtons = new JPanel();
    jpRadioButtons.setLayout(new GridLayout(3, 1));
    jpRadioButtons.add(jrbRed);
    jpRadioButtons.add(jrbGreen);
    jpRadioButtons.add(jrbBlue);
    add(jpRadioButtons, BorderLayout.WEST);
    // Create a radio-button group to group three buttons
    ButtonGroup group = new ButtonGroup();
    group.add(jrbRed);
    group.add(jrbGreen);
    group.add(jrbBlue);
```

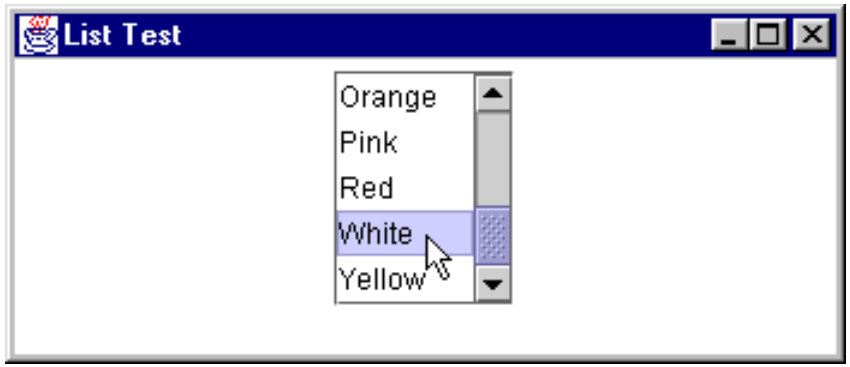
```
    // Set initial message color to blue
    jrbBlue.setSelected(true);
    jlblMessage.setForeground(Color.blue);
    // Create a new panel to hold label and text field
    JPanel jpTextField = new JPanel();
    jpTextField.setLayout(new BorderLayout(5, 0));
    jpTextField.add(
      new JLabel("Enter a new message"), BorderLayout.WEST);
    jpTextField.add(jtfMessage, BorderLayout.CENTER);
    jtfMessage.setHorizontalAlignment(JTextField.RIGHT);
    add(jpTextField, BorderLayout.NORTH);

    // Register listeners with check boxes
    jchkBold.addActionListener(new HandleEvent());
    jchkItalic.addActionListener(new HandleEvent());
    // Register listeners for radio buttons
    jrbRed.addActionListener(new HandleEvent());
    jrbGreen.addActionListener(new HandleEvent());
    jrbBlue.addActionListener(new HandleEvent());
// Register listener for text field
    jtfMessage.addActionListener(new HandleEvent());
  }
```

```
class HandleEvent implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                if (e.getSource() == jchkBold)
                        setNewFont();
                else if(e.getSource() == jchkItalic)
                        setNewFont();
                else if(e.getSource() == jrbRed)
                        jlblMessage.setForeground(Color.red);
                else if(e.getSource() == jrbGreen)
                        jlblMessage.setForeground(Color.green);
                else if(e.getSource() == jrbBlue)
                        jlblMessage.setForeground(Color.blue);
                else if (e.getSource() == jtfMessage)
                        jlblMessage.setText(jtfMessage.getText());
 }
 }
 private void setNewFont() {
   // Determine a font style
   int fontStyle = Font.PLAIN;
   fontStyle += (jchkBold.isSelected() ? Font.BOLD : Font.PLAIN);
   fontStyle += (jchkItalic.isSelected() ? Font.ITALIC : Font.PLAIN);
   // Set font for the message
     jlblMessage.setFont(new Font("TimesRoman", fontStyle, 12));
 }
}
```

Events for JComboBox, Jlist and JMenu

- Exercise 3: Write a program that changes color of the background as selected on the color list.



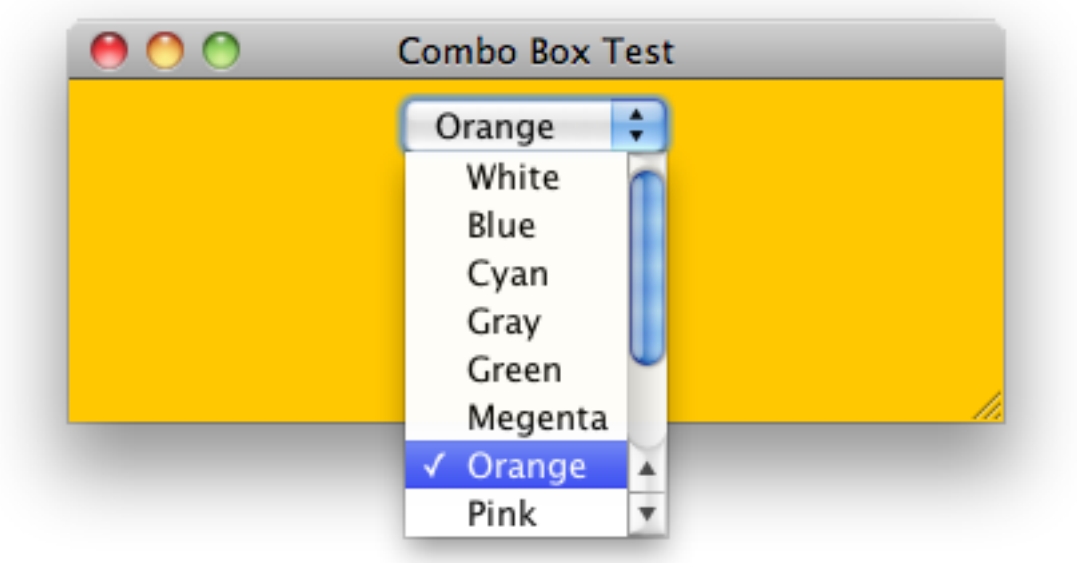


```
import java.awt.*;
import javax.swing.*;
public class ListTest extends JFrame
{
        private JList colorList;
        private JPanel c;
        private String colorName[ ] = {"Black", "Blue", "Cyan", "Gray", "Green",
        "Megenta", "Orange", "Pink", "Red", "White", "Yellow"};
        private Color colors[ ] = { Color.black, Color.blue, Color.cyan, Color.gray,
        Color.green, Color.magenta, Color.orange, Color.pink, Color.red,
        Color.white, Color.yellow};
        public ListTest()
      {
          setTitle("List Test" );
          c=new JPanel();
          c.setLayout (new FlowLayout());
          colorList = new JList(colorName);
          colorList.setVisibleRowCount(5);
          colorList.setSelectionMode( ListSelectionModel.SINGLE_SELECTION);
          c.add (new JScrollPane (colorList) );
```

```
            colorList.addListSelectionListener (new ListSelectionListener(){
            public void valueChanged (ListSelectionEvent e)
               {
                                    int index=colorList.getSelectedIndex();
                                    c.setBackground(colors[index]);
               }
            });
            add(c);

      }
         public static void main(String args[ ])
      {
        ListTest app = new ListTest();
        app. setSize(350, 150);
        app.setVisible(true);
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      }
}
```

Events for JComboBox, Jlist and JMenu

- Exercise 4: Write a program that changes color of the background as selected on the color chooser combo box.

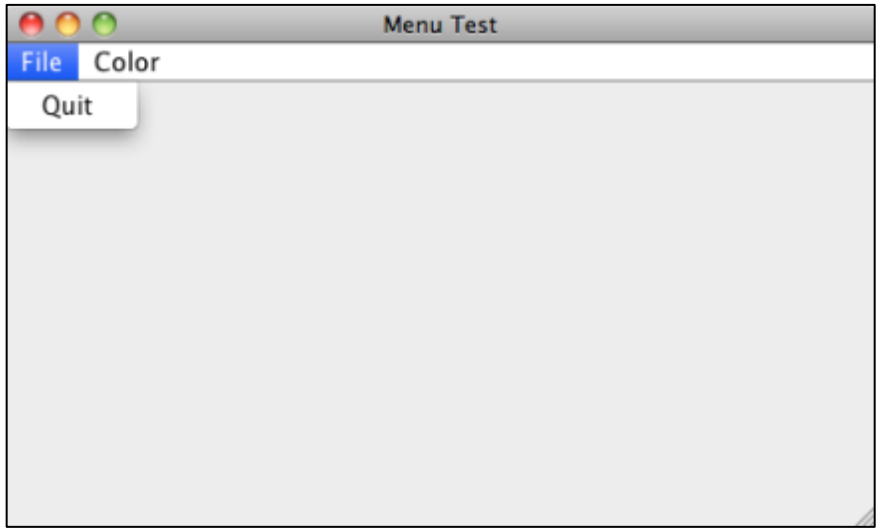


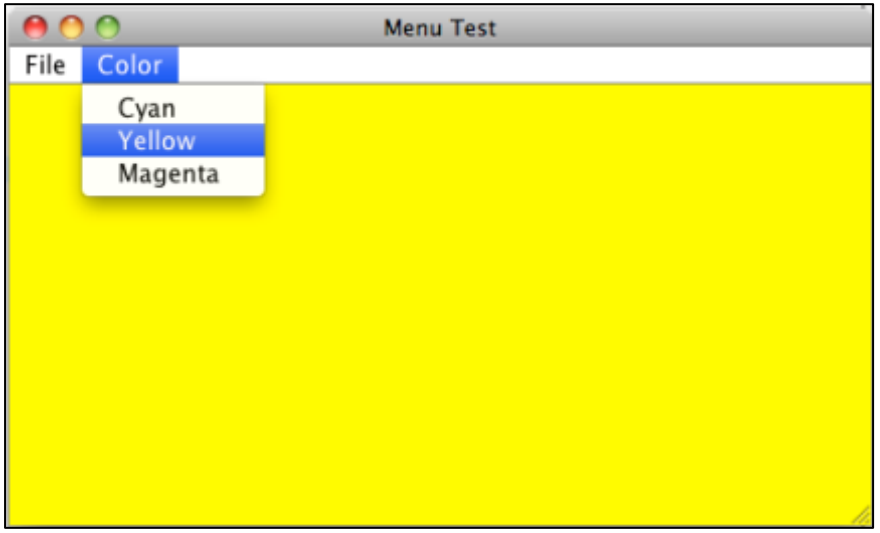
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class ColorTest extends JFrame
{
        private JComboBox colorList;
        private JPanel c;
private String colorName[ ] = {"White", "Blue", "Cyan", "Gray", "Green", "Megenta",
"Orange", "Pink", "Red", "Black", "Yellow"};
private Color colors[ ] = { Color.black, Color.blue, Color.cyan, Color.gray,
Color.green, Color.magenta, Color.orange, Color.pink, Color.red,   Color.white,
Color.yellow};
        public ColorTest()
       {
           setTitle("Combo Box Test" );
           c=new JPanel();
           c.setBackground(Color.white);//Default color
           c.setLayout (new FlowLayout());
           colorList = new JComboBox(colorName);
```

```
        colorList.addActionListener (new ActionListener(){
        public void actionPerformed (ActionEvent e)
           {
                                int index=colorList.getSelectedIndex();
                                c.setBackground(colors[index]);
            }
        });
        c.add(colorList);
        add(c);

    }
       public static void main(String args[ ])
   {
     ColorTest app = new ColorTest();
     app. setSize(350, 150);
     app.setVisible(true);
     app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Events for JComboBox, Jlist and JMenu

- Exercise 5: Write a program that provides a menu bar with File → Quit and Color → Yellow,Cyan,Magenda that allows the color to be picked to change the background color.





```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class MenuTest extends JFrame
{
   JPanel panel;
   JMenuBar mb;
   JMenu file,color;
   JMenuItem quit,cyanItem,yellowItem,magentaItem;
MenuTest()
{
        setTitle("Menu Test");
        panel = new JPanel();
        add(panel, "Center");
        mb = new JMenuBar();
        file = new JMenu("File");
        quit = new JMenuItem("Quit");
        file.add(quit);
        quit.addActionListener(new ChangeColor());
        mb.add(file);
        color = new JMenu("Color");
        cyanItem = new JMenuItem("Cyan");
        color.add(cyanItem);
        cyanItem.addActionListener(new ChangeColor());
        yellowItem = new JMenuItem("Yellow");
        color.add(yellowItem);
        yellowItem.addActionListener(new ChangeColor());
        magentaItem = new JMenuItem("Magenta");
        color.add(magentaItem);
        magentaItem.addActionListener(new ChangeColor());
        mb.add(color);
        setJMenuBar(mb);
}
```

```
class ChangeColor implements ActionListener{
public void actionPerformed(ActionEvent e)
{
        Object source = e.getSource();
        if (source==quit)
                System.exit(0);
        else if (source==cyanItem)
                panel.setBackground(Color.cyan);
        else if (source==yellowItem)
                panel.setBackground(Color.yellow);
        else if (source==magentaItem)
                panel.setBackground(Color.magenta);

}
}
public static void main(String[] args)
{
        JFrame jf = new MenuTest();
        jf.setSize(500, 300);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

Mouse Events

- A mouse event is fired whenever
  - a mouse button is pressed, released, or clicked,
  - the mouse is moved, or
  - the mouse is dragged onto a component

java.awt.event.MouseEvent	
+getButton(): int	Indicates which mouse button has been clicked.
+getClickCount(): int	Returns the number of mouse clicks associated with this event.
+getPoint(): java.awt.Point	Returns a `Point` object containing the *x*- and *y*-coordinates.
+getX(): int	Returns the *x*-coordinate of the mouse point.
+getY(): int	Returns the *y*-coordinate of the mouse point.

Mouse Event Listener Interface and Listener Interface Methods

Event Source	Action	Event type	Listener Interface	Listener Interface Methods
Component	Mouse pressed	MouseEvent	MouseListener	mousePressed
	Mouse released		MouseListener	mouseReleased
	Mouse clicked		MouseListener	mouseClicked
	Mouse entered		MouseListener	mouseEntered
	Mouse exited		MouseListener	mouseExited
	Mouse moved		MouseMotionListener	mouseMoved
	Mouse dragged		MouseMotionListener	mouseDragged

- **Example 1:** Write a program that displays a message in a panel and enables the message to be moved using a mouse. The message moves as the mouse is dragged, and it is always displayed at the mouse point

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

public class MoveMessageDemo extends JFrame {
  public MoveMessageDemo() {
    // Create a MovableMessagePanel instance for moving a message
    MovableMessagePanel p = new MovableMessagePanel( );
    // Place the message panel in the frame
    add(p);
  }

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

```
// Inner class: MovableMessagePanel draws a message
 static class MovableMessagePanel extends JPanel {
   private String message = "Welcome to Java";
   private int x = 20;
   private int y = 20;
   /** Construct a panel to draw string s */
   public MovableMessagePanel( ) {
       addMouseMotionListener(new MouseMotionListener() {
      @Override /** Handle mouse-dragged event */
      public void mouseDragged(MouseEvent e) {
        // Get the new location and repaint the screen
        x = e.getX();
        y = e.getY();
        repaint();
      }
      @Override /** Handle mouse-moved event */
      public void mouseMoved(MouseEvent e) {
      }
    });
   }
   @Override
   protected void paintComponent(Graphics g) {
     super.paintComponent(g);
     g.drawString(message, x, y);
   }
 }}
```

- **Example 2:** Write a program that displays number of mouse click on a panel as illustrated in Fig.


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class clickCounter extends JFrame {
  public clickCounter() {
            DisplayPanel panel = new DisplayPanel();
    add(panel);
}
  /** Main method */

  public static void main(String[] args) {
    JFrame frame = new clickCounter();
    frame.setTitle("clickCounter");
    frame.setSize(300, 300);
    frame.setLocationRelativeTo(null); // Center the frame
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
```

```
class DisplayPanel extends JPanel {
        int clkCount=0;
        public DisplayPanel() {
                JTextField count=new JTextField(3);
                JLabel label=new JLabel("Number of clicks:");
                add(label);
                add(count);
                this.addMouseListener(new MouseListener() {
                public void mouseClicked(MouseEvent e) {
                        clkCount++;
                        count.setText(String.valueOf(clkCount));
                  }
                @Override
                public void mousePressed(MouseEvent e) {
                }
                @Override
                public void mouseReleased(MouseEvent e) {
                }
                @Override
                public void mouseEntered(MouseEvent e) {
                }
                @Override
                public void mouseExited(MouseEvent e) {
                }
    });
  }
}
```

Listener Interface Adapters

- *A* listener interface adapter *is a class that provides the default implementation for all the methods in the listener interface.*

Adapter	*Interface*
`MouseAdapter`	`MouseListener`
`MouseMotionAdapter`	`MouseMotionListener`
`KeyAdapter`	`KeyListener`
`WindowAdapter`	`WindowListener`

```
addMouseMotionListener(
    new MouseMotionListener() {
  @Override /** Handle mouse-dragged event */
  public void mouseDragged(MouseEvent e){
    x = e.getX();
    y = e.getY();
    repaint();
  }

  @Override /** Handle mouse-moved event */
  public void mouseMoved(MouseEvent e) {
  }
});
```

(a) Using a listener interface

```
addMouseMotionListener(
    new MouseMotionAdapter() {
  @Override /** Handle mouse-dragged event */
  public void mouseDragged(MouseEvent e){
    x = e.getX();
    y = e.getY();
    repaint();
  }
});
```

(b) Using a listener interface adapter

- **Example 3:** Write a program that displays the mouse position when the mouse button is pressed and ceases to display it when the mouse button is released.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MousePressRelease extends JFrame {
  public MousePressRelease() {
    add(new DisplayCoordinatePanel());
  }
/**Main method*/
  public static void main(String[] args) {
 // Create a frame
            MousePressRelease frame = new MousePressRelease();
    frame.setTitle(“Mouse Position");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setVisible(true);
  }
}
```

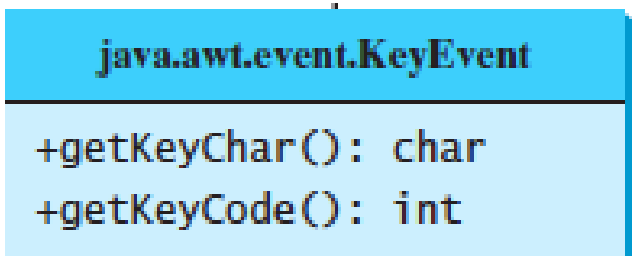
```
class DisplayCoordinatePanel extends JPanel {
  // Point is a Java object for representing points in a plane.
  // p.x and p.y are the x and y coordinates.
  private Point p = new Point(0, 0);
  private boolean pressed = false;
  public DisplayCoordinatePanel() {
    addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent e) {
        p.x = e.getX();
        p.y = e.getY();
        pressed = true;
        repaint();
      }
      public void mouseReleased(MouseEvent e) {
            pressed = false;
            repaint();
      }
    });
  }
  @Override // Draw a small solid square around the point
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if (pressed) {
      g.drawString("(" + p.x + ", " + p.y + ")", p.x, p.y);
    }
  }
}
```

- **Example 4:** Write a program that draws a circle at mouse point on each click

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DrawCircles extends JFrame{
    public DrawCircles()
    {
        add(new DrawCirclesPanel());
    }
     public static void main(String[] args) {
 // Create a frame
    DrawCircles frame = new DrawCircles();
    frame.setTitle("Mouse Position");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setVisible(true);
 }
```

```
class DrawCirclesPanel extends JPanel{
        int x,y;
        boolean pressed;
        public DrawCirclesPanel() {
           addMouseListener(new MouseAdapter() {
           public void mousePressed(MouseEvent e) {
               x = e.getX();
                y = e.getY();
            repaint();
               }
           });
        }
           protected void paintComponent(Graphics g) {
              g.setColor(Color.red);
               g.fillOval(x, y, 30, 30);


 }
    }
```

Key Events



Returns the character associated with the key in this event.

Returns the integer key code associated with the key in this event.

Key Event Listener Interface and Listener Interface Methods

Event Source	Action	Event type	Listener Interface	Listener Interface Methods
Component	Key pressed	KeyEvent	KeyListener	KeyPressed
	Key released			KeyReleased
	Key typed			KeyTyped

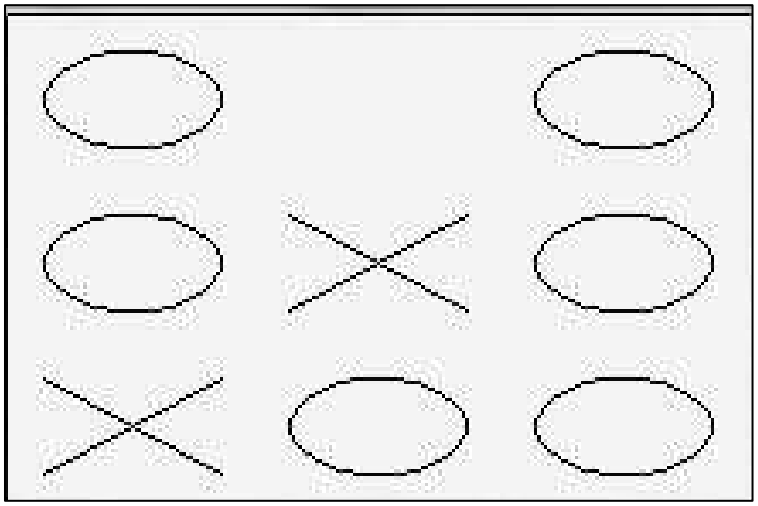
- **Example 4:** Write a program that displays a user-input character on a panel. The user can move the character up, down, left, and right, using the arrow keys VK_UP , VK_DOWN , VK_LEFT , and VK_RIGHT.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class KeyEventDemo extends JFrame {
  private KeyboardPanel keyboardPanel = new KeyboardPanel();
  /** Initialize UI */
  public KeyEventDemo() {
    // Add the keyboard panel to accept and display user input
    add(keyboardPanel);
    // Set focus
    keyboardPanel.setFocusable(true);
  }
  /** Main method */
  public static void main(String[] args) {
    KeyEventDemo frame = new KeyEventDemo();
    frame.setTitle("KeyEventDemo");
    frame.setSize(300, 300);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
```

```
// Inner class: KeyboardPanel for receiving key input
 static class KeyboardPanel extends JPanel {
   private int x = 100;
   private int y = 100;
   private char keyChar = 'A'; // Default key
   public KeyboardPanel() {
     addKeyListener(new KeyAdapter() {
      @Override
      public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
          case KeyEvent.VK_DOWN: y += 10; break;
          case KeyEvent.VK_UP: y -= 10; break;
          case KeyEvent.VK_LEFT: x -= 10; break;
          case KeyEvent.VK_RIGHT: x += 10; break;
          default: keyChar = e.getKeyChar();
        }
        repaint();
      }
    });
   }
   @Override
   protected void paintComponent(Graphics g) {
     super.paintComponent(g);
     g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
     g.drawString(String.valueOf(keyChar), x, y);
   }
 }
```

Assignments

1. Write a program to draw the following figure.



2. Write a program to perform addition, subtraction, multiplication, and division, as shown in Figure.

3. Write a program that displays the background color of a panel as black when the mouse button is pressed and as white when the mouse button is released.
4. Write a program that converts miles and kilometers. If you enter a value in the Mile text field and press the Enter key, the corresponding kilometer measurement is displayed in the Kilometer text field. Likewise, if you enter a value in the Kilometer text field and press the Enter key, the corresponding miles is displayed in the Mile text field.
5. Write a program that draws rectangle when a user clicks on the panel. The message 'Maximum click reaches' will be displayed on the panel when the user reaches above ten clicks.

Next Week Lecture

- Database Programming