

# CIS 27P

Class 10

# Events

- Supported by `java.awt.event`
- Events
  - selecting an item in a list
  - clicking a mouse
- Event Sources
  - Window
  - Button
  - Checkbox
  - Menu Item
- Event Listeners
  - `MouseListener` Interface
  - `KeyListener` Interface
  - `TextListener` Interface

# Event Classes

- `java.util.EventObject`
  - `Object getSource()`
- `java.awt.event.AWTEvent`
  - `int getId()`
- `AWTEvent` subclasses:
  - `ActionEvent`, `AdjustmentEvent`,  
`ComponentEvent`, `ContainerEvent`,  
`FocusEvent`, `InputEvent`, `ItemEvent`,  
`KeyEvent`, `MouseEvent`, `TextEvent`,  
`WindowEvent`

# Event Sources

- Event Sources are objects that generate events.
- Common event sources:
  - Button, Checkbox, Choice, List, Menu Item, Scrollbar, Text components, Window
- Event Listener interfaces define appropriate methods corresponding to events for that type of listener
  - `MouseListener.mouseClicked()`
  - `KeyListener.keyPressed()`

# Using Listeners

- Implement listener interface to receive events desired
- Register the listener with a source as a recipient for the event notification
- Code methods defined in the listener interface for desired functionality

# Adapter classes

- Adapter classes simplify some event listener by implementing empty event handler methods.
- Available Adapters:
  - Component Adapter
  - ContainerAdapter
  - FocusAdapter
  - KeyAdapter
  - MouseAdapter
  - MouseMotionAdapter
  - WindowAdapter

# Inner & Anonymous Classes

- Inner classes simplify the syntax of using adapters.
- Anonymous inner classes simplify the process further.

# Containers

- Extend `java.awt.Component`
  - Allows components and other containers to be stored inside
- Panel
  - Plain top level window, superclass of Applet
- Window
  - Top level main application window.
- Frame
  - Adds title bar, borders, menus
- Canvas
  - Plain area for drawing

# Graphics

- Use Graphics class
- `g.fillOval( int x, int y, int w, int h );`
- `g.fillRect( int x, int y, int w, int h );`
- `g.drawString(String s, int x, int y );`
- `g.setFont( “SansSerif”, Font.BOLD, 12 );`
- `g.setColor( Color.red );`

# Button

- P 723 - 727 of textbook
- Instantiate  
`Button b = new Button("Submit");`
- To add to applet window ( in `init()` ):  
`add( b );`
- Implement `ActionListener`
- Handle click in method:  
`public void actionPerformed(ActionEvent ae)`
- To check which button:  
`if ( b == ae.getSource() )`

# TextField

- P 743-745 of textbook
- Instantiate:  
`TextField f = new TextField(25);`
- To add to applet window ( in `init()` ):  
`add( f );`
- To get value:  
`f.getText();`

# Using Frames

- Create a subclass of Frame
- use setVisible( boolean b ) method to display and hide
- Pass reference to other windows that need to interact

```
/** This class represents a Circle to be drawn in the  
    Frame Application Window. */
```

```
class Circle
```

```
{
```

```
    private int x, y, w, h;
```

```
    public Circle( int inX, int inY, int inW, int inH )
```

```
    {
```

```
        x = inX;
```

```
        y = inY;
```

```
        w = inW;
```

```
        h = inH;
```

```
    }
```

```
    public int getX() { return x; }
```

```
    public int getY() { return y; }
```

```
    public int getW() { return w; }
```

```
    public int getH() { return h; }
```

```
}
```

```
/**This class represents a Rectangle to be drawn in the  
Frame Application Window.**/  

```

```
class Rectangle
```

```
{
```

```
    private int x, y, w, h;
```

```
    public Rectangle( int inX, int inY, int inW, int inH )
```

```
    {
```

```
        x = inX;
```

```
        y = inY;
```

```
        w = inW;
```

```
        h = inH;
```

```
    }
```

```
    public int getX() { return x; }
```

```
    public int getY() { return y; }
```

```
    public int getW() { return w; }
```

```
    public int getH() { return h; }
```

```
}
```

```
/** This class represents a String to be drawn in the  
Frame Application Window. */
```

```
class Text
```

```
{
```

```
    private int x, y, s;
```

```
    String v;
```

```
    public Text( int inX, int inY, int inS, String inV )
```

```
    {
```

```
        x = inX;
```

```
        y = inY;
```

```
        s = inS;
```

```
        v = inV;
```

```
    }
```

```
    public int getX() { return x; }
```

```
    public int getY() { return y; }
```

```
    public int getS() { return s; }
```

```
    public String getV() { return v; }
```

```
}
```

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

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

```
import java.util.*;
```

```
public class FrameApplication extends Frame  
    implements ActionListener
```

```
{
```

```
    private Button circle, rectangle, text, clear;
```

```
    private HashSet images = new HashSet();
```

```
/**
FrameApplication.main()
Starts main application window.
*/
public static void main( String args[] )
{
    FrameApplication fm = new FrameApplication();
    fm.setSize(500, 500 );
    fm.setTitle("Frame Application Sample");
    fm.setVisible( true );
}
```

```
/**FrameApplication Constructor.  
Create main application window and add listeners. */  
public FrameApplication()  
{  
    setLayout( new FlowLayout( ) );  
    addWindowListener( new ExitWindowAdapter() );  
    circle = new Button("Circle");  
    rectangle = new Button("Rectangle");  
    text = new Button("Text");  
    clear = new Button("Clear");  
    add( circle );  
    add( rectangle );  
    add( text );  
    add( clear );  
  
    circle.addActionListener( this );  
    rectangle.addActionListener( this );  
    text.addActionListener( this );  
    clear.addActionListener( this );  
}
```

```
/**  
Class ExitWindowAdapter  
Adapter that closes main application window.  
*/  
class ExitWindowAdapter extends WindowAdapter  
{  
    // Exits when main window is closed.  
    public void windowClosing( WindowEvent we )  

```

```

/**FrameWindow.paint( Graphics g )
Iterate through HashSet images and draw..*/
public void paint(Graphics g)
{
    Iterator i = images.iterator();
    while ( i.hasNext() )
    {
        Object o = i.next();
        if ( o instanceof Circle )
        {
            g.setColor( Color.green );
            Circle c = (Circle) o;
            g.drawOval( c.getX(), c.getY(),
                c.getWidth(), c.getHeight() );
        }
        else if ( o instanceof Rectangle )
        {
            g.setColor( Color.blue );
            Rectangle r = (Rectangle) o;
            g.drawRect( r.getX(), r.getY(),
                r.getWidth(), r.getHeight() );
        }
        else if ( o instanceof Text )
        {
            g.setColor( Color.red );
            Text t = (Text) o;
            Font f = new Font("SansSerif",
                Font.BOLD, t.getSize());
            g.setFont( f );
            g.drawString( t.getText(), t.getX(),
                t.getY() );
        }
    }
}
}

```

```

/**
FrameApplication.actionPerformed( ActionEvent ae )
Called when any of the top buttons are clicked.
*/
public void actionPerformed( ActionEvent ae )
{
    Object o = ae.getSource();
    Frame popup = null;
    String component = null;

    if ( o == clear )
    {
        //clears all images from hashset
        images = new HashSet();
    }
    else
    {
        //determine which type of window to create based on
        //what button was clicked
        if ( o == circle ) popup= new CircleChoices( this );
        if ( o == rectangle ) popup = new RectangleChoices( this );
        if ( o == text ) popup = new TextChoices( this );
        popup.setSize(500, 100 );
        popup.setLocation( 250, 250 );
        popup.setVisible( true );
    }

    repaint();
}

```

```

/**Class that draws and processes the Circle Configuration window.*/
class CircleChoices extends Frame implements ActionListener
{
    Button submitButton;
    FrameApplication parent;
    Label lx, ly, lw, lh;
    TextField tx, ty, tw, th;

    public CircleChoices( FrameApplication p )
    {
        setTitle("Circle Configuration");
        setLayout( new FlowLayout( ) );
        parent = p;
        addWindowListener( new PopupWindowAdapter() );

        submitButton = new Button("Submit");
        tx = new TextField(3);
        ty = new TextField(3);
        tw = new TextField(3);
        th = new TextField(3);
        add( new Label("X:") );
        add( tx );
        add( new Label("y:") );
        add( ty );
        add( new Label("w:") );
        add( tw );
        add( new Label("h:") );
        add( th );
        add( submitButton );

        submitButton.addActionListener( this );
    }
}

```

```
/**
Adapter that closes popup configuration windows.
*/
class PopupWindowAdapter extends WindowAdapter
{
    // Called when any of the submit buttons for
    // configuration windows are pressed.
    public void windowClosing( WindowEvent we )
    {
        Frame f = (Frame)we.getSource();
        f.setVisible( false );
    }
}
```

```

/**
CircleChoices.actionPerformed( ActionEvent ae )
Called when popup window submit is pressed.
*/
public void actionPerformed( ActionEvent ae )
{
    int x=250, y=250, w=50, h=50;
    try
    {
        x = Integer.parseInt( tx.getText() );
        y = Integer.parseInt( ty.getText() );
        w = Integer.parseInt( tw.getText() );
        h = Integer.parseInt( th.getText() );
    }
    catch ( Exception e ){}

    Circle c = new Circle ( x, y, w, h );
    parent.addImage( c );
    setVisible( false );
}

```

```
/**
```

```
FrameApplication.addImage( Object img )
```

```
Accept a new image from popup window, adds it to  
HashSet, and calls repaint().
```

```
*/
```

```
public void addImage( Object img )
```

```
{
```

```
    images.add( img );
```

```
    repaint();
```

```
}
```

```
/**RectangleChoices Class: draws and processes rectangle window */
class RectangleChoices extends Frame implements ActionListener
{
    Button submitButton;
    FrameApplication parent;
    Label lx, ly, lw, lh;
    TextField tx, ty, tw, th;

    public RectangleChoices( FrameApplication p )
    {
        setTitle("Rectangle Configuration");
        setLayout( new FlowLayout( ) );
        parent = p;
        addWindowListener( new PopupWindowAdapter() );

        submitButton = new Button("Submit");
        tx = new TextField(3);
        ty = new TextField(3);
        tw = new TextField(3);
        th = new TextField(3);
        add( new Label("X:") );
        add( tx );
        add( new Label("y:") );
        add( ty );
        add( new Label("w:") );
        add( tw );
        add( new Label("h:") );
        add( th );
        add( submitButton );

        submitButton.addActionListener( this );
    }
}
```

```
/*
    RectangleChoices.actionPerformed( ActionEvent ae )
    called when Submit is selected from Rectangle
    Configuration
*/
public void actionPerformed( ActionEvent ae )
{
    int x=250, y=250, w=50, h=50;
    try
    {
        x = Integer.parseInt( tx.getText() );
        y = Integer.parseInt( ty.getText() );
        w = Integer.parseInt( tw.getText() );
        h = Integer.parseInt( th.getText() );
    }
    catch ( Exception e ){}

    Rectangle r = new Rectangle ( x, y, w, h );
    parent.addImage( r );
    setVisible( false );
}
```

```

/**TextChoices Class: draws and processes text window */
class TextChoices extends Frame implements ActionListener
{
    Button submitButton;
    FrameApplication parent;
    Label lx, ly, ls, lv;
    TextField tx, ty, ts, tv;

    // creates popup accepting text configuration
    public TextChoices( FrameApplication p )
    {
        setTitle("Text Configuration");
        setLayout( new FlowLayout( ) );
        parent = p;
        addWindowListener( new PopupWindowAdapter() );
        submitButton = new Button("Submit");
        tx = new TextField(3);
        ty = new TextField(3);
        ts = new TextField(3);
        tv = new TextField(15);
        add( new Label("X:") );
        add( tx );
        add( new Label("y:") );
        add( ty );
        add( new Label("size:") );
        add( ts );
        add( new Label("text:") );
        add( tv );
        add( submitButton );
        submitButton.addActionListener( this );
    }
}

```

```
/*
    TextChoices.actionPerformed( ActionEvent ae )
    called when Submit is selected from Text
    Configuration
*/
```

```
public void actionPerformed( ActionEvent ae )
{
    int x=250, y=250, s=50;
    String v="Text";
    try
    {
        x = Integer.parseInt( tx.getText() );
        y = Integer.parseInt( ty.getText() );
        s = Integer.parseInt( ts.getText() );
        v = tv.getText();
    }
    catch ( Exception e ){}

    Text t = new Text ( x, y, s, v );
    parent.addImage( t );
    setVisible( false );
}
```