Java Programming II

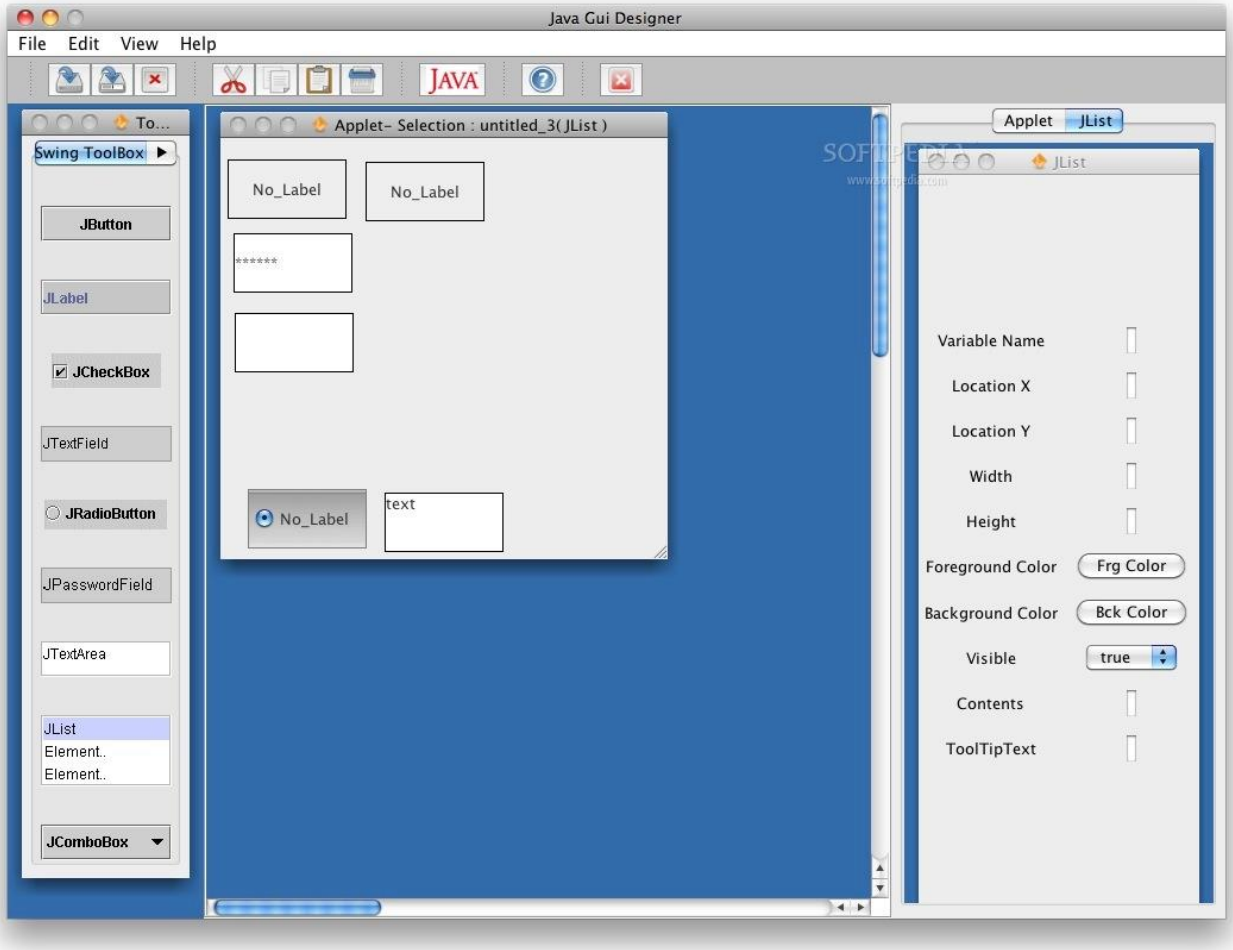
Events and Applet

Contents

- Event Handling (Old Model)
- Events and Delegation Event Model
- Overview of Applets
- First Java Applet
- The Life Cycle of an Applet
- The Graphics Class
- Using Colors
- Displaying Text
- Using Applet Dimensions
- Using Applets in a Web Page
- The Applet Class
- The AppletContext Class
- Using Images

GUI and Event

- When you create a GUI for an application, how can you deal with usr's action?



Event Handling

- An event is an object that describes some state change in a source
- Each time a user interacts with a component an event is generated, e.g.:
  - A button is pressed
  - A menu item is selected
  - A window is resized
  - A key is pressed
- An event informs the program about the action that must be performed

Event Handling (Old Model)

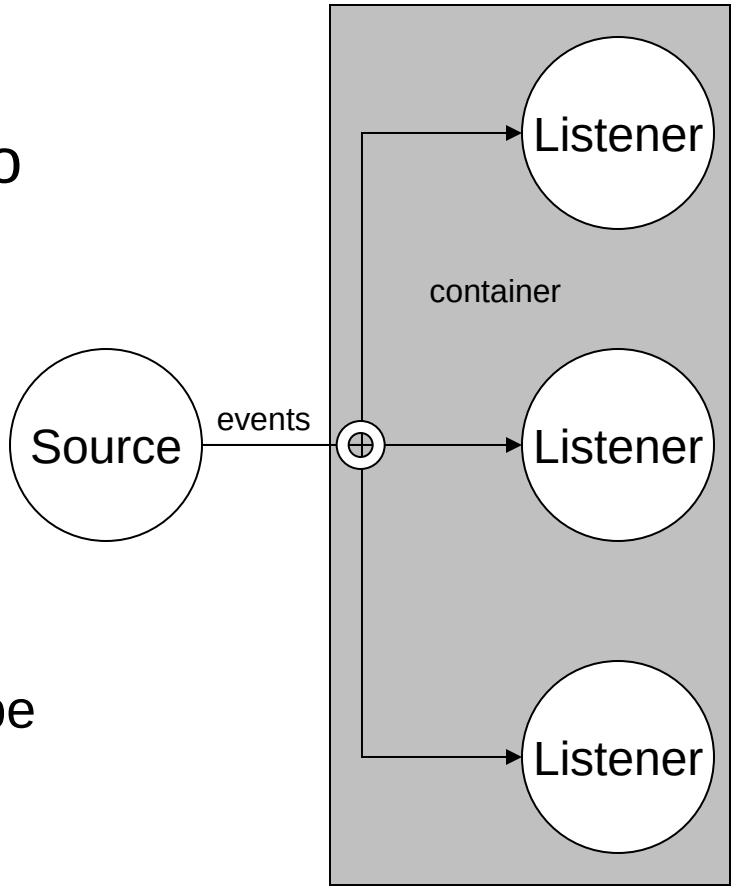
- Using the action method. It needs some code to identify an event source and to do action for the event.

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

/*
<APPLET CODE=EventHandlingOld.class
      WIDTH=400 HEIGHT=200>
</APPLET>
*/


public class EventHandlingOld extends Applet {
    String message = "";
    Button button1 = new Button("Label Set"),
           button2 = new Button("OK");
    public void init() {
        add(button1);
        add(button2);
    }
```

```
    public void paint(Graphics g) {
        g.drawString(message, 20, 70);
    }

    /* Old Model for Event Handling  */
    public boolean action (Event e, Object arg) {
        if ( e.target instanceof Button ) {
            if(e.target.equals(button1))
                message = "Label Set Button Clicked!";
            else if(e.target.equals(button2))
                message = "OK Button Clicked!";
            repaint();
        }
        return true;
    }
    /* end of code for event handling */

} // End of the EventHandlingOld Class
```

The Delegation Event Model

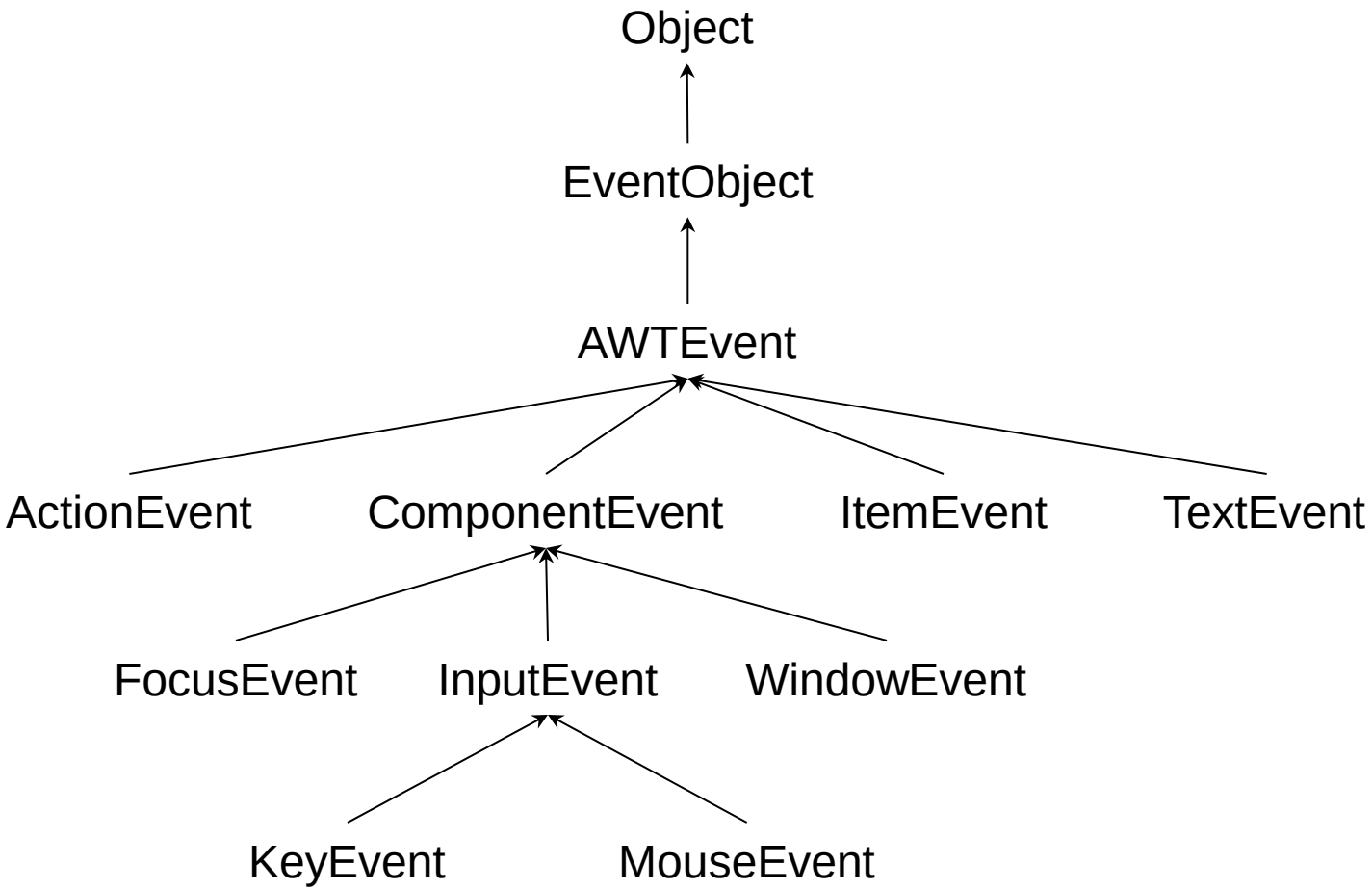
- The model provides a standard mechanism for a **source** to generate an **event** and send it to a set of **listeners**
- A source generates events.
- Three responsibilities of a source:
  - To provide methods that allow listeners to register and unregister for notifications about a specific type of event
  - To generate the event
  - To send the event to all registered listeners.



The Delegation Event Model

- A listener receives event notifications.
- Three responsibilities of a listener:
  - To register to receive notifications about specific events by calling appropriate registration method of the source.
  - To implement an interface to receive events of that type.
  - To unregister if it no longer wants to receive those notifications.
- The delegation event model:
  - A source multicasts an event to a set of listeners.
  - The listeners implement an interface to receive notifications about that type of event.
  - In effect, the source delegates the processing of the event to one or more listeners.

Event Classes Hierarchy



Event Classes

- The EventObject class has the two methods
  - The getSource() method returns the object that generated the event
  - toSource() method returns a string equivalent of the event.
- The AWTEvent(Object source, int id)
  - Source is the object that generates the event and id identifies the type of the event.
  - The class has the getID() method that returns the type of the event.
- Event Listener (java.util.EventListener) interface does not define any constraints or methods but exists only to identify those interfaces that process events
- The Component class has the methods that allow a listener to register and unregister for events:
  - void add*Type*Listener(*Type*Listener tl)
  - void remove*Type*Listener(*Type*Listener tl)

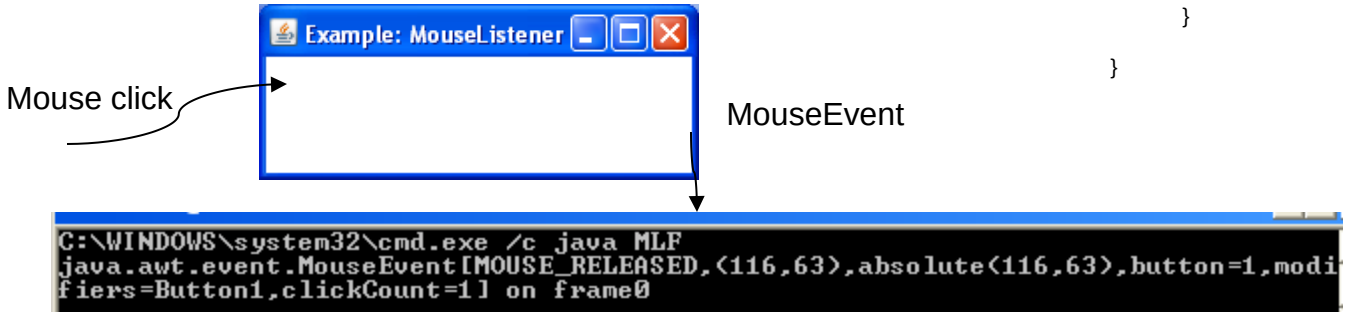
AWT Event Classes and Listener Interfaces

Event Class	Listener Interface
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener
ItemEvent	ItemListener
KeyEvent	KeyListener
MouseEvent	MouseListener, MouseMotionListener
TextEvent	TextListener
WindowEvent	WindowListener

MouseListener (Low-Level Listener)

- The MouseListener interface defines the methods to receive mouse events.
  - mouseClicked(MouseEvent me)
  - mouseEntered(MouseEvent me)
  - mouseExited(MouseEvent me)
  - mousePressed(MouseEvent me)
- A listener must register to receive mouse events

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

class MLF extends Frame implements MouseListener {

        MLF()
        {
                super("Example: MouseListener");

                addMouseListener(this);
                setSize(250,100);
        }

        public void mouseClicked(MouseEvent me)  {
        }

        public void mouseEntered(MouseEvent me)  {
        }

        public void mouseExited(MouseEvent me) {
        }

        public void mousePressed(MouseEvent me) {
        }

        public void mouseReleased(MouseEvent me) {
                System.out.println(me);
        }

        public static void main(String[] args) {
                new MLF().setVisible(true);
        }

}
```



Adapter Classes (Low-Level Event Listener)

- It should be implemented for all the methods in the *Type*Listener interfaces.
- When we use the Adapter Classes (they are abstract classes), it is OK to overrides the methods you want.
- Examples
  - FocusAdapter, WindowAdpater, KeyAdapter, MouseAdapter, MouseMotionAdapter, MouseInputAdapter

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
  <applet code="MouseInnerDemo" width=100
        height=100>
  </applet>
*/

public class MouseInnerDemo extends Applet {

  public void init() {
    setBackground(Color.green);
    addMouseListener(new MyMouseAdapter());
  }

  class MyMouseAdapter extends MouseAdapter {

    public void mousePressed(MouseEvent me) {
      setBackground(Color.red);
      repaint();
    }

    public void mouseReleased(MouseEvent me) {
      setBackground(Color.green);
      repaint();
    }
  }
}
```

Semantic Event Listener

- The semantic events relate to operations on the components in the GUI. There are three semantic event classes: ActionEvent, ItemEvent, and AdjustmentEvent.
  - An ActionEvent is generated when there was an action performed on a component such as clicking on a menu item or a button.
    - Produced by Objects of Type: Buttons, Menus, Text
  - An ItemEvent occurs when a component is selected or deselected.
    - Produced by Objects of Type: Buttons, Menus
  - An AdjustmentEvent is produced when an adjustable object, such as a scrollbar, is adjusted.
    - Produced by Objects of Type: Scrollbar
- Semantic Event Listeners
  - Listener Interface: ActionListener, Method: void actionPerformed(ActionEvent e)
  - Listener Interface: ItemListener, Method: void itemStateChanged (ItemEvent e)
  - Listener Interface: AdjustmentListener, Method: void adjustmentValueChanged (AdjustmentEvent e)

Using the ActionListener

- Stages for Event Handling by ActionListener
  - First, import event class

    import java.awt.event.*;

  - Define an overriding class of event type (implements ActionListener)

```
class ButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                // Write what to be done. . .
            label.setText("Hello World!");
        }
}
```

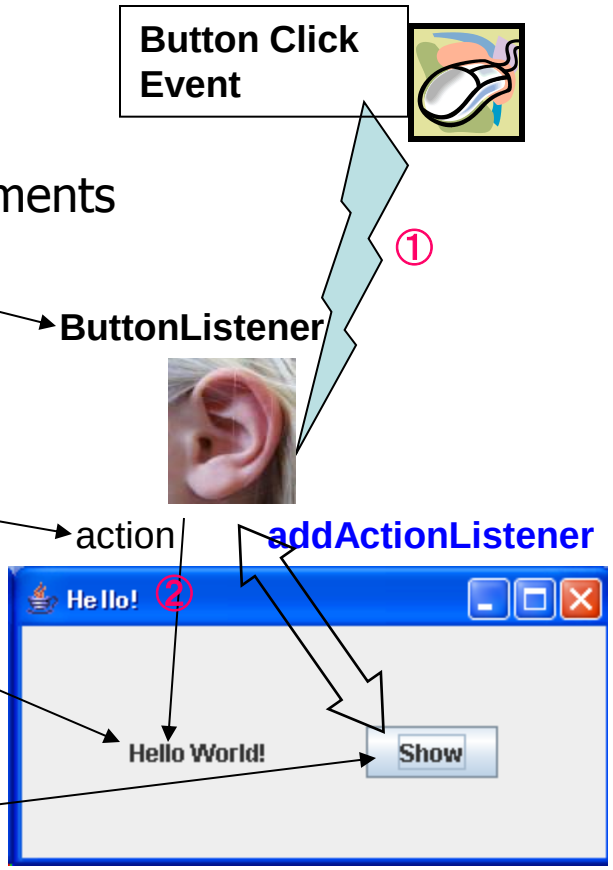
  - Create an event listener object

    ButtonListener bt = new ButtonListener();

  - Register the event listener object

    b1 = new Button("Show");

    b1.**addActionListener**(bt);



A Hello Example Using Button Listener

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

public class HelloAWT extends Frame {    // Using Frame
  Label contents;
  Button dispbutton;

  public HelloAWT() {  // Constructor
    setLayout(new FlowLayout(FlowLayout.CENTER, 50, 50));

    contents = new Label("                    ");  // Create Label object
    add(contents);  // Add the label to this Frame

    dispbutton = new Button("Show");  // Create Button object
    dispbutton.addActionListener(new DispButtonListener()); // Add Event Listener
    add(dispbutton);  // Add the button object to this Frame
  }

  class DispButtonListener implements ActionListener  { // Event Listener
     public void actionPerformed(ActionEvent e) {  // What to do when the button is
        clicked
       contents.setText("Hello World!");
     }
   }

  public static void main (String[] args) {
    HelloAWT f = new HelloAWT();  // Create Hello GUI
    f.setTitle("Hello!");
    f.setSize(400,150);
    f.setVisible(true);
  }
} // end of "HelloAWT.java"
```

Run:

Java HelloAWT

An Overview of Applets

- Applet : A program that can be referenced by HTML source code of a Web page.
- Can be run on Web browser after having been downloaded.
- May be dangerous
- Security Problem

First Java Applet

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="FirstApplet" width=200 height=200>
  </applet>
*/

public class FirstApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("This is my first applet!", 20, 100);
  }
}
```

- Extends Applet
- Graphics by Abstract Window Toolkit (AWT)
- Run: after compile,
- appletviewer FirstApplet.html

  or

  appletviewer FirstApplet.java

The Life Cycle of an Applet

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="AppletLifecycle" width=300
height=50>
  </applet>
*/

public class AppletLifecycle extends Applet {
  String str = "";

  public void init() {
    str += "init; ";
  }

  public void start() {
    str += "start; ";
  }

  public void stop() {
    str += "stop; ";
  }

  public void destroy() {
    System.out.println("destroy");
  }

  public void paint(Graphics g) {
    g.drawString(str, 10, 25);
  }
}
```

- init() : called only when the applet begins execution.
- start() : executed after init() method. Called by the applet viewer or Web browser.
- stop() : when applet viewer is minimized.
- destroy() : called by the applet viewer or Web browser before the applet is terminated.

The Graphics Class

Methods of Graphics Class

abstract void drawArc(int x, int y, int w, int h, int degreesO, i nt degrees1)

abstract boolean drawImage(Image img, int x, int y, Image Observer io)

abstract void drawLine(int x0, int y0, int x, int y1)

abstract void drawOval(int x, int y, int w, int h)

abstract void drawPolygon(int x[], int u[], int n)

abstract void drawPolyline(int x[], int y[], int n)

void drawRect(int x, int y, int w, int h)

abstract void drawString(String str, int x, int y)

abstract void fillArc(int x, int y, int w, int h, int degree0, int degree1)

abstract void fillOval(int x ,int y, int w, int h)

abstract void fillPolygon(int x[], int y[], int n)

void fillRect(int x, int y, int w, int h)

abstract Color getColor()

abstract Font getFont()

abstract FontMetrics getFontMetrics()

abstract void setColor(Color c)

abstract void setFont(Font f)

```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="DrawArc" width=200 height=200>
  </applet>
*/

pulic class DrawArc extends Applet {

  public void paint(Graphics g) {
    g.drawArc(20, 20, 160, 160, 0, 135);
  }
}
```

http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html

Using Colors

Color Constructors

Color(int red, int green, int blue)

Color(int rgb)

Color(float r, float g, float b)

Method of Color Class

static int HSBtoRGB(float h, float s, float b)

static float[] RGBtoHSB(int r, int g, int b, float hsb[])

Color brighter()

Color darker()

static Color decode(String str) throws NumberFormatExcepti on

boolean equals(Object obj)

int getBlue()

int getGreen()

int getRGB()

int getRed()

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
/*
  <applet code="BlueString" width=300 height=100>
  </applet>
*/

public class BlueString extends Applet {

  public void paint(Graphics g) {
    g.setColor(Color.blue);
    g.drawString("Blue String", 100, 50);
  }
}
```

http://docs.oracle.com/javase/8/docs/api/java/awt/Color.html

Displaying Text

- **Font Constructor**

  Font(String name, int style, int ps)

- **setFont() Method**

  abstract void setFont(Font font)

- **FontMetrics Constructor**

  FontMetrics(Font font)

```
import java.applet.Applet;
import java.awt.*;
/*
  <applet code="FontDemo" width=200 height=200>
  </applet>
*/

public class FontDemo extends Applet {

  public void paint(Graphics g) {

    // Draw baseline
    int baseline = 100;
    g.setColor(Color.lightGray);
    g.drawLine(0, baseline, 200, baseline);

    // Draw String
    g.setFont(new Font("Serif", Font.BOLD, 36));
    g.setColor(Color.black);
    g.drawString("Wxyz", 5, baseline);
  }
}
```

http://docs.oracle.com/javase/8/docs/api/java/awt/Font.html

Using Applet Dimensions

- **getSize() Method**

Dimension getSize()

- **Dimension Constructors**

Dimension()
Dimension(Dimension d)
Dimension(int w, int h)

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="Circle" width=200 height=200>
  </applet>
*/

public class Circle extends Applet {

  public void paint(Graphics g) {
    Dimension d = getSize();
    int xc = d.width / 2;
    int yc = d.height / 2;
    int radius = (int)((d.width < d.height) ?
      0.4 * d.width : 0.4 * d.height);
    g.drawOval(xc - radius, yc - radius, 2 * radius, 2 *
      radius);
  }
}
```

http://docs.oracle.com/javase/8/docs/api/java/awt/Dimension.html

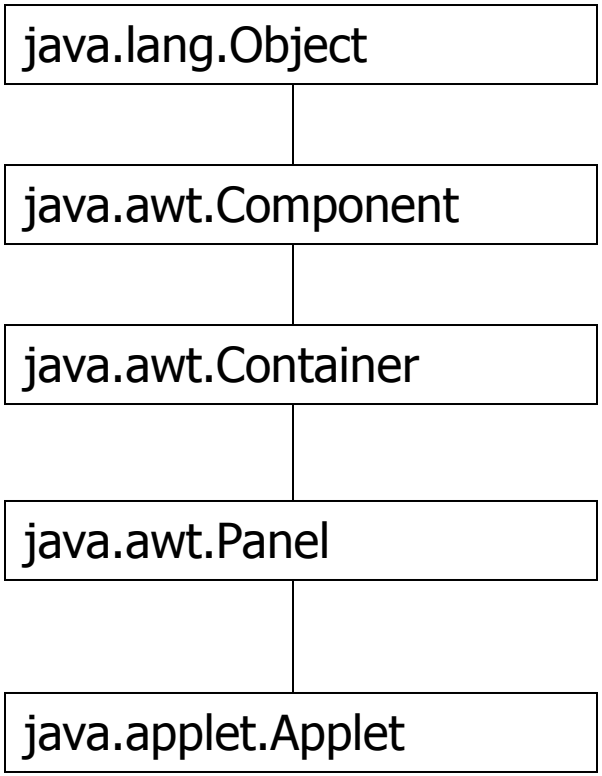
Using Applets in a Web Page

- **Applet HTML Tag**

<applet
[codebase=url]
[alt=text]
[name=appName]
width=wpixels
height=hpixels
[align=alignment]
[vspace=vspixels]
[hspace=hspixels]
>
[<param name=pname1 value=value1>]
[<param name=pname2 value=value2>]
…….
[<param name=pnameN value=valueN>]
</applet>

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="AppletParameters" width=300
height=300>
  <param name="background" value="0xffffff">
  <param name="foreground" value="0x000000">
  <param name="message" value="Testing Applet
Parameters">
  </applet>
*/

public class AppletParameters extends Applet {

  public void paint(Graphics g) {
    String background = getParameter("background");
    String foreground = getParameter("foreground");
    String message = getParameter("message");
    setBackground(Color.decode(background));
    setForeground(Color.decode(foreground));
    Font font = getFont();
    FontMetrics fm = getFontMetrics(font);
    Dimension d = getSize();
    int x = (d.width - fm.stringWidth(message)) / 2;
    int y = d.height / 2;
    g.drawString(message, x, y);
  }
}
```

The Applet Class



Applet and its superclasses

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="BackgroundForeground" width=200
    height=200>
  </applet>
*/

public class BackgroundForeground extends Applet {

  public void paint(Graphics g) {
    setBackground(Color.yellow);
    setForeground(Color.blue);
    g.drawLine(0, 0, 200, 200);
    g.fillRect(100, 40, 50, 50);
  }
}
```

The AppletContext Interface

- **AppletContext Interface**

Applet getApplet(String appName)
Enumeration getApplets()
AudioClip getAudioClip(URL url)
Image getImage(URL url)
void showDocument(URL url)
void showDocument(URL url, String target)
void showStatus(String str)

```
import java.applet.*;
import java.awt.*;
import java.net.*;
/*
  <applet code="ShowDocument" width=200
height=50>
  </applet>
*/

public class ShowDocument extends Applet {

  public void init() {
    AppletContext ac = getAppletContext();
    try {
      URL url = new URL("http://www.osborne.com");
      ac.showDocument(url, "frame2");
    }
    catch(Exception e) {
      showStatus("Exception: " + e);
     }
  }

  public void paint(Graphics g) {
    g.drawString("ShowDocument Applet", 10, 25);
  }
}
```

Using Images

- **getImage() Methods**

Image getImage(URL url)

Image getImage(URL base, String fileName)

- **drawImage() Methods**

abstract boolean drawImage(Image img, int x, int y, I mageObserver io)

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="DrawImage" width=280 height=280>
  <param name="file" value="kids2.jpg">
  </applet>
*/

public class DrawImage extends Applet {
  Image image;

  public void init() {
    image = getImage(getDocumentBase(),
      getParameter("file"));
  }

  public void paint(Graphics g) {
    g.drawImage(image, 0, 0, this);
  }
}
```

Using Threads

update() and repaint() Methods

- *repaint() : request an update of the applet display*
- *update() : clears the applet display with the background color and then invokes the paint() method in default, Can be overrided.*

```
public class Counter extends Applet
implements Runnable {
  int counter;
  Thread t;

  public void init() {

    // Initialize counter
    counter = 0;

    // Start thread
    t = new Thread(this);
    t.start();
  }
```

```
public void run() {
    try {
      while(true) {
        // Request a repaint
        repaint();
        //Sleep before displaying next count
        Thread.sleep(1000);
        //Increment counter
        ++counter;
      }
    }
    catch(Exception e) {    }
  }

  public void paint(Graphics g) {

    // Set Font
    g.setFont(new Font("Serif", Font.BOLD, 36));

    // Get font metrics
    FontMetrics fm = g.getFontMetrics();

    // Display counter
    String str = "" + counter;
    Dimension d = getSize();
    int x = d.width / 2 - fm.stringWidth(str) / 2;
    g.drawString(str, x, d.height / 2);
  }
}
```

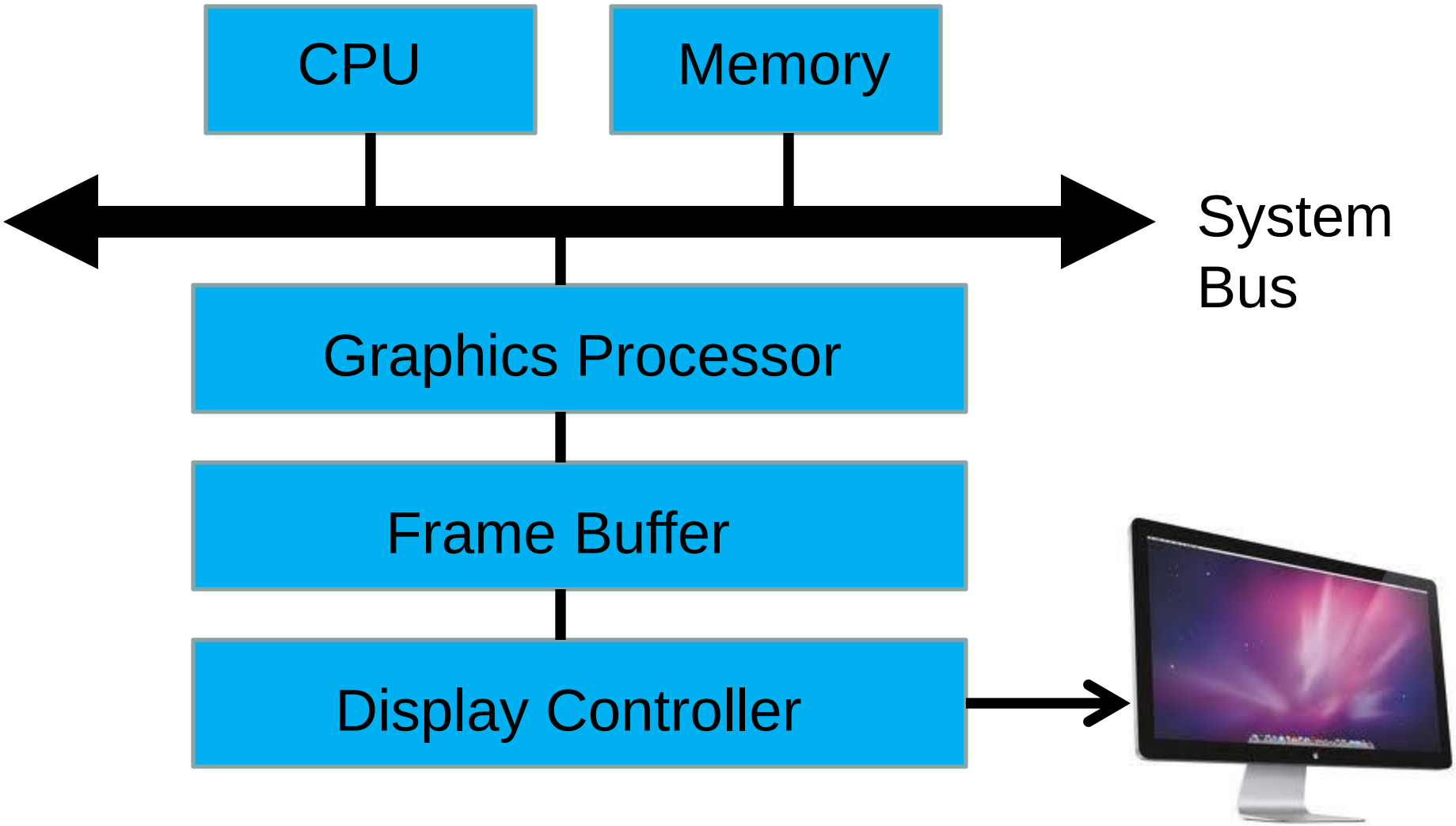
Double Buffering (No Double Buffering)

Double Buffering can be used to avoid display "flicker" in applets.

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="NoDoubleBuffer" width=300
height=100>
  </applet>
*/

public class NoDoubleBuffer extends Applet
implements Runnable {
  int x = 0;
  Thread t;

  public void init() {

    // Start thread
    t = new Thread(this);
    t.start();
  }

  public void run() {
    try {
      while(true) {
```

```
        // Request a repaint
          repaint();

          //Sleep before update
          Thread.sleep(100);
        }
      }
      catch(Exception e) {
      }
    }

    public void paint(Graphics g) {

      // Draw filled circle
      Dimension d = getSize();
      g.fillOval(x, d.height / 4, 50, 50);

      // Increment x
      x += 5;
      if (x + 50 > d.width)
        x = 0;
    }
}
```

Raster Graphics System Architecture



Double Buffering (With Double Buffering)

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="DoubleBuffer" width=300 height=100>
  </applet>
*/

public class DoubleBuffer extends Applet
implements Runnable {
  int x = 0;
  Thread t;
  Image buffer;
  Graphics bufferg;

  public void init() {

    // Start thread
    t = new Thread(this);
    t.start();

    // Create buffer
    Dimension d = getSize();
    buffer = createImage(d.width, d.height);
  }

  public void run() {
    try {
      while(true) {

        //Request a repaint
        repaint();
```

```
        // Sleep before update
          Thread.sleep(100);
      }
    }
    catch(Exception e) {
    }
  }

  public void update(Graphics g) {
    paint(g);
  }

  public void paint(Graphics g) {

    //Get graphics object for buffer
    if (bufferg == null)
      bufferg = buffer.getGraphics();

    //Draw to buffer
    Dimension d = getSize();
    bufferg.setColor(Color.white);
    bufferg.fillRect(0, 0, d.width, d.height);
    bufferg.setColor(Color.black);
    bufferg.fillOval(x, d.height / 4, 50, 50);

    //Update screen
    g.drawImage(buffer, 0, 0, this);

    //Increment x
    x += 5;
    if (x + 50 > d.width)
      x = 0;
  }
}
```