

## Section 9: AWT GUI Objects and Event Handling

### 1) AWT GUI Objects

GUI objects are those features that allow the creation of a graphics user interface that allows the program to respond to the user's input.

### 2) Events and Event Handling

- a) Events occur when a user interacts with GUI objects. Event Handling is how the program reacts to that interaction.
- b) Java programs cycle looking for events
- c) Requires importing class event: `import java.awt.event.*;`
- d) Events are: key, button, mouse, scrollbar, etc.
- e) `repaint()` causes appletviewer or browser to invoke `paint()`

### 3) ShowStatus()

- a) outputs string to status line of Applet
- b) syntax: `showStatus(string);`

### 4) To using GUI objects, your program needs:

- a) the implements class
- b) the declaration and initialization
- c) the registration
- d) the event handling

### 5) scrollbars

```
e)implements: AdjustmentListener { // add to class header
f)Scrollbar identifier;
g)public void init() // allways called first
{
    scrollbaridentifier = new Scrollbar(Scrollbar.HORIZONTAL,
        initial value, step size, lower limit, relative upper limitr);
    scrollbaridentifer = new Scrollbar(Scrollbar.VERTICAL,
        initial value, step size, lower limit, relative
        upper limitr);
    add(scrollbaridentifer);
    scrollbaridentifier.addAdjustmentListener(this);
}
h)public void adjustmentValueChanged(AdjustmentEvent e)
{
    intvariable = scrollbaridentifer.getValue();
    repaint();
}
```

example of event with scrollbar:

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

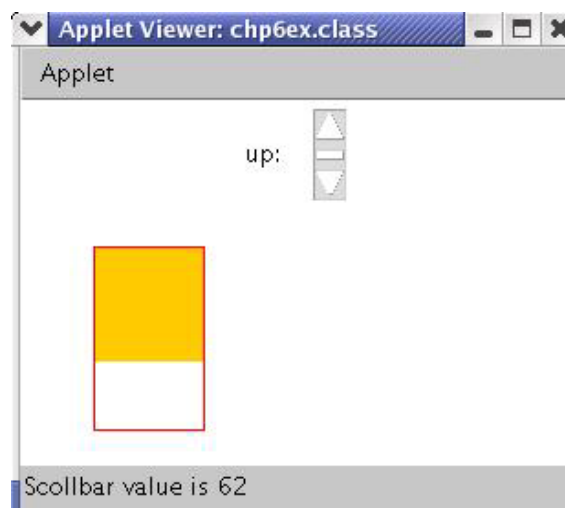
public class chp6ex extends Applet implements AdjustmentListener {

    private Scrollbar slider;
    private int sliderValue;

    public void init() {
        Label title;
        title = new Label("up:");
        add(title);
        slider = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,100);
        add(slider);
        slider.addAdjustmentListener(this);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.red);
        showStatus("Scollbar value is " + sliderValue);
        g.drawRect(40,80,60,100);
        g.setColor(Color.orange);
        g.fillRect(41,81,59,sliderValue);
    }

    public void adjustmentValueChanged(AdjustmentEvent e) {
        sliderValue = slider.getValue();
        repaint();
    }
}
```



## 6) Buttons

```
a)import java.awt.event.*;
b)implements ActionListener
c)private Button buttonIdentifier;
d)buttonIdentifier = new Button("label");
e)add(buttonIdentifier);
f)buttonIdentifier.addActionListener(this);
g)public void actionPerformed(ActionEvent e)
```

example:

```
public void actionPerformed(ActionListener e)
{
    if (e.getSource() == buttonIdentifier)
        statements
}
```

Example of using buttons:

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

public class chp7ex extends Applet implements ActionListener {

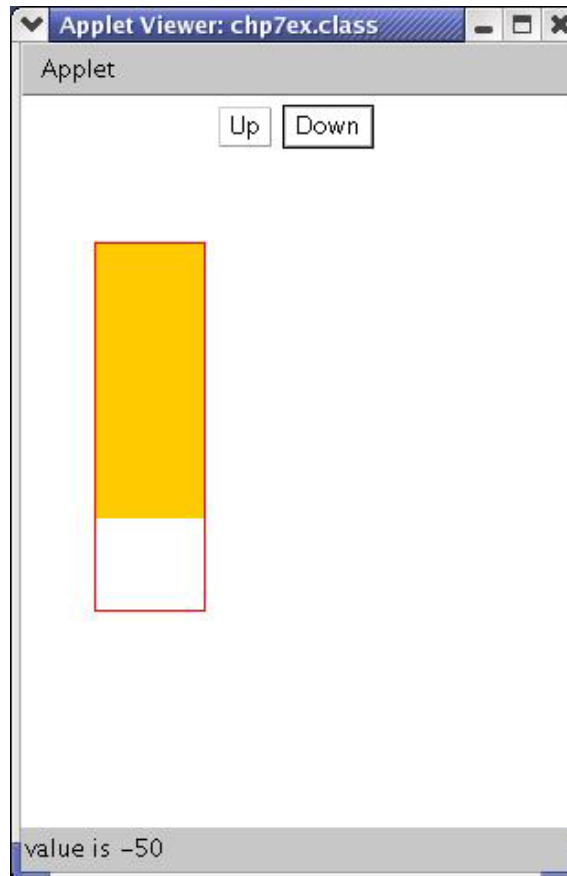
    private Button up, down;
    private int value;

    public void init() {
        value = 0;
        String titleUp, titleDown;
        titleUp = "Up";
        titleDown = "Down";
        up = new Button(titleUp);
        down = new Button(titleDown);
        add(up);
        add(down);
        up.addActionListener(this);
        down.addActionListener(this);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.red);
        showStatus("value is " + value);
        g.drawRect(40,80,60,201);
        g.setColor(Color.orange);
        g.fillRect(41,81,59,-value+100);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == up && value < 100)
            value+=10;
        if (e.getSource() == down && value >-100)
            value-=10;
    }
}
```

```
        repaint();  
    }  
}
```



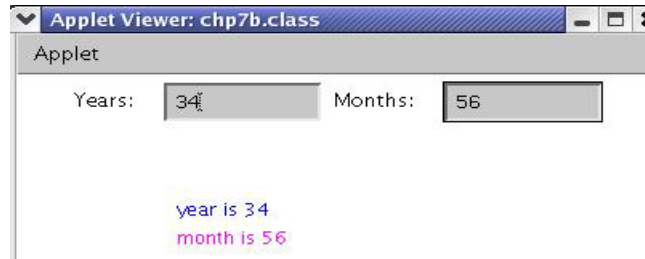
## 7) TextFields

```
a)private TextField textIdentifier;  
b)textIdentifier = new TextField(num characters);  
c)add(textIdentifier);  
d)textIdentifier.addActionListener(this);  
e)public void actionPerformed(ActionEvent e) {  
    StringVar = textIdentifier.getText();  
    IntVar    = Integer.parseInt(textIdentifier.getText());  
f)also: TextField message = new TextField(45);  
    message.setText("whatdaya wanna say goes here");
```

Example of using Text Fields:

```
import java.awt.*;  
import java.applet.Applet;  
import java.awt.event.*;  
  
public class chp7b extends Applet implements ActionListener {  
  
    private TextField tYears, tMonths;  
    private int years, months, spam;  
  
    public void init() {  
        Label lYears, lMonths;  
        lYears = new Label("Years:");  
        lMonths = new Label("Months:");  
        tYears = new TextField(10);  
        tMonths = new TextField(10);  
        add(lYears);  
        add(tYears);  
        tYears.addActionListener(this);  
        add(lMonths);  
        add(tMonths);  
        tMonths.addActionListener(this);  
    }  
  
    public void paint(Graphics g) {  
        setBackground(Color.white);  
        g.setColor(Color.blue);  
        g.drawString("year is " + years,100,100);  
        g.setColor(Color.magenta);  
        g.drawString("month is " + months, 100, 120);  
    }  
  
    public void actionPerformed(ActionEvent e) {  
        if (e.getSource() == tYears)  
            years = Integer.parseInt(tYears.getText());  
        if (e.getSource() == tMonths)
```

```
        months = Integer.parseInt(tMonths.getText());  
        repaint();  
    }  
}
```



## 8) Check Boxes

- a) implements class: ItemListener
- b) d&i: private Checkbox i; i = new Checkbox("label");
- c) reg: add(i); i.addItemListener(this);
- d) event handling:

example:

```
import java.awt.*;  
import java.applet.Applet;  
import java.awt.event.*;  
  
public class chp17 extends Applet implements ItemListener {  
  
    private Checkbox spam, catsup, coffee, toast;  
    private boolean cSpam, cCatsup, cCoffee, cToast;  
  
    public void init() {  
        cSpam    = false;  
        cCatsup = false;  
        cCoffee  = false;  
        cToast   = false;  
  
        spam    = new Checkbox("Spam");  
        add(spam);  
        spam.addItemListener(this);  
  
        catsup = new Checkbox("Catsup");  
        add(catsup);  
        catsup.addItemListener(this);  
  
        coffee = new Checkbox("Coffee");  
        add(coffee);  
        coffee.addItemListener(this);  
  
        toast  = new Checkbox("Toast");  
        add(toast);  
    }  
}
```

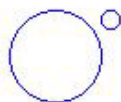
```
        toast.addItemListener(this);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.blue);
        g.drawOval(100,50,50,50);
        g.drawOval(150,50,10,10);

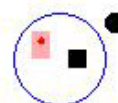
        if (cSpam) {
            g.setColor(Color.pink);
            g.fillRect(110,60,10,15);
        }
        if (cCatsup) {
            g.setColor(Color.red);
            g.fillOval(113,63,4,4);
        }
        if (cCoffee) {
            g.setColor(Color.black);
            g.fillOval(150,50,10,10);
            g.drawOval(150,50,10,10);
        }
        if (cToast) {
            g.setColor(Color.black);
            g.fillRect(130,70,10,10);
        }
    }

    public void itemStateChanged(ItemEvent e) {
        if (e.getSource() == spam)
            cSpam = spam.getState();
        if (e.getSource() == catsup)
            cCatsup = catsup.getState();
        if (e.getSource() == coffee)
            cCoffee = coffee.getState();
        if (e.getSource() == toast)
            cToast = toast.getState();
        repaint();
    }
}
```

Spam  Catsup  Coffee  Toast



Spam  Catsup  Coffee  Toast



## 9) Check Box Groups

a) allows on only one selection

b) implements: ItemListener

c) d&i: private CheckboxGroup cbg; Checkbox i; cbg = new CheckboxGroup();

d) reg: i = new Checkbox("label", cbg, boolean); add(i); i.addItemListener();

e) event handling:

example:

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

public class chp17a extends Applet implements ItemListener {

    private CheckboxGroup liquids;
    private Checkbox spam, catsup, coffee, tea, toast;
    private boolean cSpam, cCatsup, cCoffee, cTea, cToast;

    public void init() {
        cSpam = false;
        cCatsup = false;
        cCoffee = true;
        cTea = false;
        cToast = false;

        liquids = new CheckboxGroup();

        spam = new Checkbox("Spam");
        add(spam);
        spam.addItemListener(this);

        catsup = new Checkbox("Catsup");
        add(catsup);
        catsup.addItemListener(this);

        coffee = new Checkbox("Coffee", liquids, true);
        add(coffee);
        coffee.addItemListener(this);

        tea = new Checkbox("Tea", liquids, false);
        add(tea);
        tea.addItemListener(this);

        toast = new Checkbox("Toast");
        add(toast);
        toast.addItemListener(this);
    }
}
```

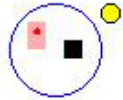
```
public void paint(Graphics g) {
    setBackground(Color.white);
    g.setColor(Color.blue);
    g.drawOval(100,50,50,50);
    g.drawOval(150,50,10,10);

    if (cSpam) {
        g.setColor(Color.pink);
        g.fillRect(110,60,10,15);
    }
    if (cCatsup) {
        g.setColor(Color.red);
        g.fillOval(113,63,4,4);
    }
    if (cCoffee) {
        g.setColor(Color.black);
        g.fillOval(150,50,10,10);
        g.drawOval(150,50,10,10);
    }
    if (cTea) {
        g.setColor(Color.yellow);
        g.fillOval(150,50,10,10);
        g.setColor(Color.black);
        g.drawOval(150,50,10,10);
    }
    if (cToast) {
        g.setColor(Color.black);
        g.fillRect(130,70,10,10);
    }
}

public void itemStateChanged(ItemEvent e) {
    if (e.getSource() == spam)
        cSpam = spam.getState();
    if (e.getSource() == catsup)
        cCatsup = catsup.getState();
    if (e.getSource() == coffee) {
        cCoffee = coffee.getState();
        cTea = tea.getState();
    }
    if (e.getSource() == tea) {
        cTea = tea.getState();
        cCoffee = coffee.getState();
    }
    if (e.getSource() == toast)
        cToast = toast.getState();
    repaint();
}
```

}

Spam  Catsup  Coffee  Tea  Toast



## 10) Pull Down Lists

- a) implements class: ItemListener
- b) `Choice i; i = new Choice();`
- c) `reg: i.add("label"); add(i); i.addItemListener(this);`
- d) event handling:

example:

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

public class chp17b extends Applet implements ItemListener {

    private Choice pickOver;
    private String picked;

    public void init() {
        pickOver = new Choice();

        pickOver.add("--Select--");
        pickOver.add("Magenta");
        pickOver.add("Blue");
        pickOver.add("Green");
        pickOver.add("Yellow");
        pickOver.add("Orange");
        pickOver.add("Red");

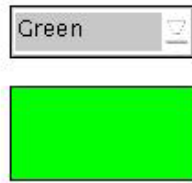
        add(pickOver);

        pickOver.addItemListener(this);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.black);
        g.drawRect(49,49,101,51);
        g.setColor(Color.white);
        if (picked == "Magenta")
```

```
        g.setColor(Color.magenta);
    else if (picked == "Blue")
        g.setColor(Color.blue);
    else if (picked == "Green")
        g.setColor(Color.green);
    else if (picked == "Yellow")
        g.setColor(Color.yellow);
    else if (picked == "Orange")
        g.setColor(Color.orange);
    else if (picked == "Red")
        g.setColor(Color.red);
        g.fillRect(50,50,100,50);
    }

    public void itemStateChanged(ItemEvent e) {
        if (e.getSource() == pickOver)
            picked = e.getItem().toString();
            repaint();
    }
}
```



## 11) Lists

i) implements class: ActionListener  
j) d&i: List i; i = new List(#,boolean);  
k) reg: i.add("label"); add(i); i.addActionListener(this);  
l) event handling:

example:

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

public class chp17c extends Applet implements ActionListener {

    private List pickOver;
    private String picked;

    public void init() {
        pickOver = new List(6,false); // true - multiple selections

        pickOver.add("Magenta");
```

```
pickOver.add("Blue");
pickOver.add("Green");
pickOver.add("Yellow");
pickOver.add("Orange");
pickOver.add("Red");

add(pickOver);

pickOver.addActionListener(this);
}

public void paint(Graphics g) {
    setBackground(Color.white);
    g.setColor(Color.black);
    g.drawRect(49,149,101,51);
    g.setColor(Color.white);
    if (picked == "Magenta")
        g.setColor(Color.magenta);
    else if (picked == "Blue")
        g.setColor(Color.blue);
    else if (picked == "Green")
        g.setColor(Color.green);
    else if (picked == "Yellow")
        g.setColor(Color.yellow);
    else if (picked == "Orange")
        g.setColor(Color.orange);
    else if (picked == "Red")
        g.setColor(Color.red);
    g.fillRect(50,150,100,50);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == pickOver)
        picked = pickOver.getSelectedItemAt();

        // for multiple selections:
        // String [] s = list.getSelectedItems();

    repaint();
}
}
```



## 12) Text Areas activated by Button Click

- a) allows text entry across multiple lines
- b) button implements class: ActionListener
- c) dec: Button b = new Button("label");  
      TextArea i = new TextArea(rows, columns);
- d) reg: add(b); b.addActionListener(this);
- e) event handling:

example:

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

public class chp17d extends Applet implements ActionListener {

    private TextArea t;
    private Button go;
    private String textEntered;

    public void init() {
        t = new TextArea(5,20);
        go = new Button("Push when Done");

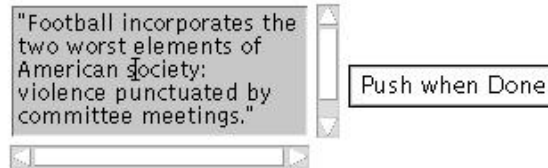
        add(t);
        add(go);

        go.addActionListener(this);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.black);
        g.drawString(textEntered, 10, 150);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == go)
            textEntered = t.getText();
    }
}
```

```
        repaint();  
    }  
}
```



"Football incorporates the two worst elements of American society: violence punctuated by committee meetings."—George Will

### 13) Canvases

- a) areas to draw on that can be registered so as not to mess up other areas
- b) declared as an internal class that extends Canvas
- c) class consists of paint function to draw on canvas
- d) containing class declares an object of internal class
- e) internal class object is initialized via calls to `.setBackground`, `.setSize`
- f) internal class object is registered via `add`

example:

```
import java.awt.*;  
import java.applet.Applet;  
import java.awt.event.*;  
  
public class chp17e extends Applet {  
    private DaCanvas c = new DaCanvas();  
  
    public void init() {  
        c.setBackground(Color.yellow);  
        c.setSize(200,150);  
        add(c);  
    }  
  
    class DaCanvas extends Canvas {  
        public void paint(Graphics g) {  
            g.setColor(Color.red);  
            g.fillRect(50,50,100,50);  
        }  
    }  
}
```



## 14) Panels

- a) grouping of components
- b) d&i: `Panel p = new Panel();`
- c) registering objects in a panel: `p.add(i);`
- d) registering the panel: `add(p);`
- e) objects in the panel are treated in the usual ways

example:

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

public class chp17f extends Applet {

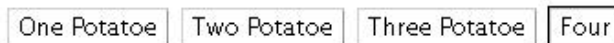
    private Panel p = new Panel();
    private Button b = new Button("One Potatoe");
    private Button bb = new Button("Two Potatoe");
    private Button bbb = new Button("Three Potatoe");
    private Button bbbb = new Button("Four");

    public void init() {
        p.add(b);
        p.add(bb);
        p.add(bbb);
        p.add(bbbb);

        add(p);
    }

    public void paint(Graphics g) {
        setBackground(Color.white);
    }

}
```



## 15) Layout Managers

- a) flow layout - default for applets (puts components side by side in rows)
- b) border layout - default for applications (puts components in zones or areas NORTH, SOUTH, EAST, WEST, CENTER)
- c) area specified by adding area to declaraton (in quotes) as parameter
- d) invoked: `setLayoutI(new BorderLayout());`

e)to avoid buttons being streached to fill whole area, put in panels

example:

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

public class chp17g extends Applet {

    Button bNorth = new Button("El Norte");
    Button bSouth = new Button("The Saouth");
    Button bEast = new Button("East Coast");
    Button bWest = new Button("Left Coast");
    Button bCenter = new Button("Dead Center");

    Panel pNorth = new Panel ();
    Panel pSouth = new Panel ();
    Panel pEast = new Panel ();
    Panel pWest = new Panel ();
    Panel pCenter = new Panel ();

    public void init() {
        setLayout(new BorderLayout());

        pNorth.add(bNorth);
        add("North", pNorth);

        pSouth.add(bSouth);
        add("South", pSouth);

        pEast.add(bEast);
        add("East", pEast);

        pWest.add(bWest);
        add("West", pWest);

        pCenter.add(bCenter);
        add("Center", pCenter);
    }
}
```



## 16) Text Fields without Buttons

```
a)private TextField textIdentifier;  
b)textIdentifier = new TextField(num characters);  
c)add(textIdentifier);  
d)textIdentifier.addActionListener(this);  
e)public void actionPerformed(ActionEvent e) {  
    StringVar = textIdentifier.getText();  
    IntVar    = Integer.parseInt(textIdentifier.getText());  
    FloatVar  = Float.parseFloat(inputField.getText());  
f)also: TextField message = new TextField(45);  
        message.setText("whatdaya wanna say goes here");
```

Example of using Text Fields:

```
import java.awt.*;  
import java.applet.Applet;  
import java.awt.event.*;  
  
public class chp7b extends Applet implements ActionListener {  
  
    private TextField tYears, tMonths;  
    private int years, months, spam;  
  
    public void init() {  
        Label lYears, lMonths;  
        lYears = new Label("Years:");  
        lMonths = new Label("Months:");  
        tYears = new TextField(10);  
        tMonths = new TextField(10);  
        add(lYears);  
        add(tYears);  
        tYears.addActionListener(this);  
        add(lMonths);  
        add(tMonths);  
        tMonths.addActionListener(this);  
    }  
  
    public void paint(Graphics g) {  
        setBackground(Color.white);  
        g.setColor(Color.blue);  
        g.drawString("year is " + years,100,100);  
        g.setColor(Color.magenta);  
        g.drawString("month is " + months, 100, 120);  
    }  
  
    public void actionPerformed(ActionEvent e) {  
        if (e.getSource() == tYears)  
            years = Integer.parseInt(tYears.getText());  
        if (e.getSource() == tMonths)  
            months = Integer.parseInt(tMonths.getText());  
        repaint();  
    }  
}
```

}



## 17) Responding to Key Presses

- a) Implement the interface `KeyListener`

```
public final class class className extends Applet implements KeyListener {
```

- b) Interface `KeyListener` requires three methods: `keyPressed`, `keyReleased`, `keyTyped`

- i) `keyPressed`

```
public void keyPressed(KeyEvent e) { }
```

- ii) `keyReleased`

```
public void keyReleased(KeyEvent e) { }
```

- iii) `keyTyped`

```
public void keyTyped(KeyEvent e) {  
    char c = e.getKeyChar();  
    // stuff based on value of c  
}
```

- c) add the `KeyListener` before the main loop: `addKeyListener(this)`

- d) in the main loop, place a the top of the loop body: `requestFocus();`