Applets

- Applets
  - Small programs that can be transmitted over the internet.
  - Provides all of the necessary support for window based activities.
- Things which an applet can do
  - Perform arithmetic calculations
  - Display graphics
  - Play sounds
  - Accept user input
  - Create animations and play interactive games

- An applet is a window based program and is different from console based programs.
- Applets are event driven.
- Applets do not have a main() method.
- Applets are displayed in a window and they use AWT to perform input and output.
- Applets cannot read from or write to the files in the local computer.

Steps involved in developing and testing an applet are:

1. Building an applet code(.java file)
2. Creating an executable applet(.class file)
3. Designing a web page using HTML tags
4. Incorporating an <APPLET> tag into the web page.
5. Testing the applet code.

```
import java.awt.*;
import java.applet.*;
…………..
…………..
public class appletclassname extends Applet
{
………
………
  public void paint(Graphics g)
{
………                                    //Applet operations code
………
}
………..
………..
}
```

A sample Applet program

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

public class HelloJava extends Applet
{
public void paint(Graphics g)
{
g.drawString("Hello Java",10,100);
}
}
```

- Two ways to run an applet

  1. Within a Java Compatible web browser.

  2. Using an applet viewer – a standard JDK Tool

When an applet begins, the AWT calls the following methods in this sequence

```
init()
start()
paint()
```

When an applet is terminated, the following sequence of method calls take place.

stop()

destroy()

init()

- The first method to be called
- Initializes variables
- This method is called only once during the runtime of an applet

start()

- called after init()
- start() is called each time an applet’s HTML document is displayed on the screen. So if the user leaves a web page and comes back, the applet resumes execution at start().

paint()

- Called each time applet’s output must be redrawn.
- Also called when the applet begins execution.
- The paint method has one parameter of type Graphics.

stop()

- called when the web browser leaves the HTML document containing applet.

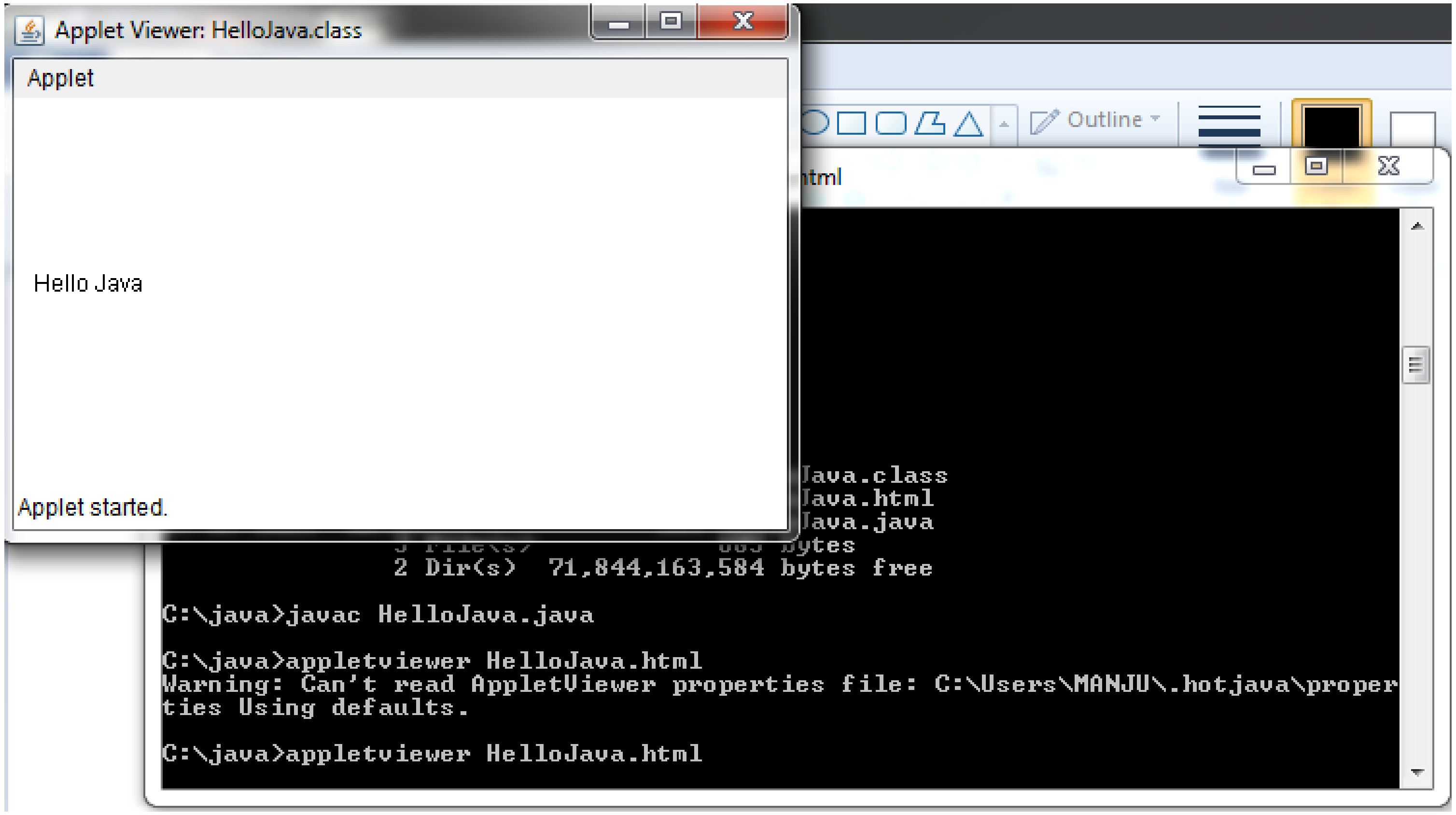
destroy()

- called when the environment determined that the applet needs to be removed completely from memory. The stop() method is always called before destroy().

HelloJava.html

```
<HTML>
<HEAD>
<TITLE>Welcome to Java Applets</TITLE>
</HEAD>
<BODY>
<APPLET CODE=HelloJava.class WIDTH=400 HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

```
> javac  HelloJava.java
> appletviewer HelloJava.html
```



```
                                                 Java.class
                                                 Java.html
                                                 Java.java
               3 File(s)            885 bytes
               2 Dir(s)  71,844,163,584 bytes free

C:\java>javac HelloJava.java

C:\java>appletviewer HelloJava.html
Warning: Can't read AppletViewer properties file: C:\Users\MANJU\.hotjava\proper
ties Using defaults.

C:\java>appletviewer HelloJava.html
```

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

public class SumNumbers extends Applet
{
 public void paint(Graphics g)
 {
  int val1 =10;
  int val2=20;
  int sum= val1+val2;
  String s="sum   :"+String.valueOf(sum);
 g.drawString(s,100,100);
}}
```

```
<html>
<applet
   code=SumNumbers.class
   width=300
   height=300>
</applet>
</html>
```

Graphics Programming

- Using Java applets we can draw lines, figures of different shapes, images, and text in different fonts and styles.
- A java applet draws graphical image inside its space using the coordinate system.
- DRAWING METHODS OF GRAPHICS Class
- drawLine() – displays a line in current drawing color.

  void drawLine(int startX, int startY, int ednX,int endY)

  g.drawLine(0,0,100,100);

drawRect() and fillRect()

– to display an outlined and filled rectangle.

void drawRect(int top, int left, int width,int height)

void fillRect(int top, int left, int width,int height)

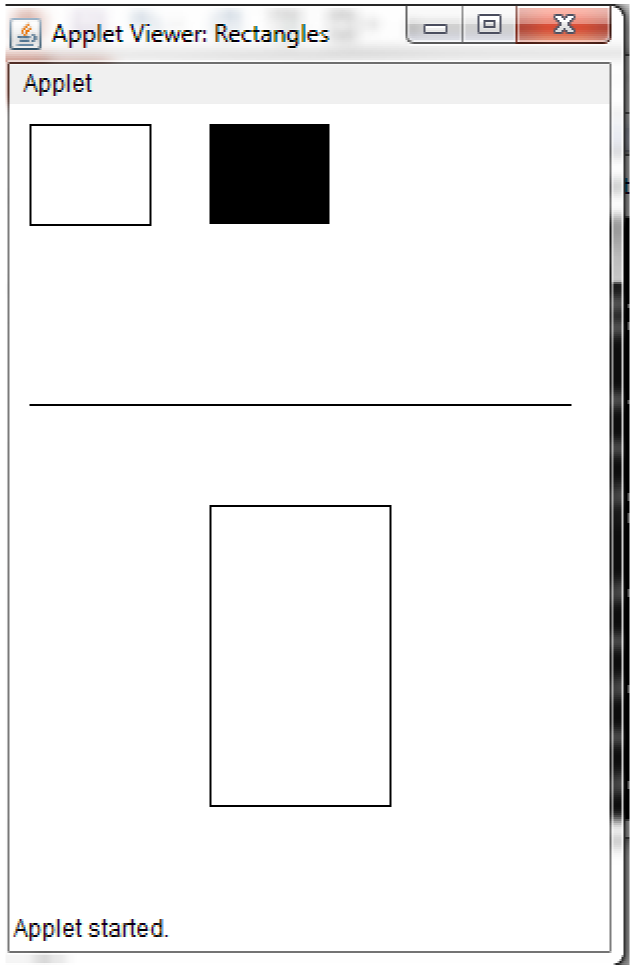
```
import java.awt.*;
import java.applet.*;

/* <applet code="Rectangles" width=300 height=400>
</applet>
*/
public class Rectangles extends Applet
{
public void paint(Graphics g)
{
g.drawRect(10,10,60,50);
g.fillRect(100,10,60,50);
g.drawLine(10,150,280,150);
g.drawRect(100,100,90,150);

}}
```

```
javac Rectangles.java

appletviewer Rectangles.java
```



Applet started.

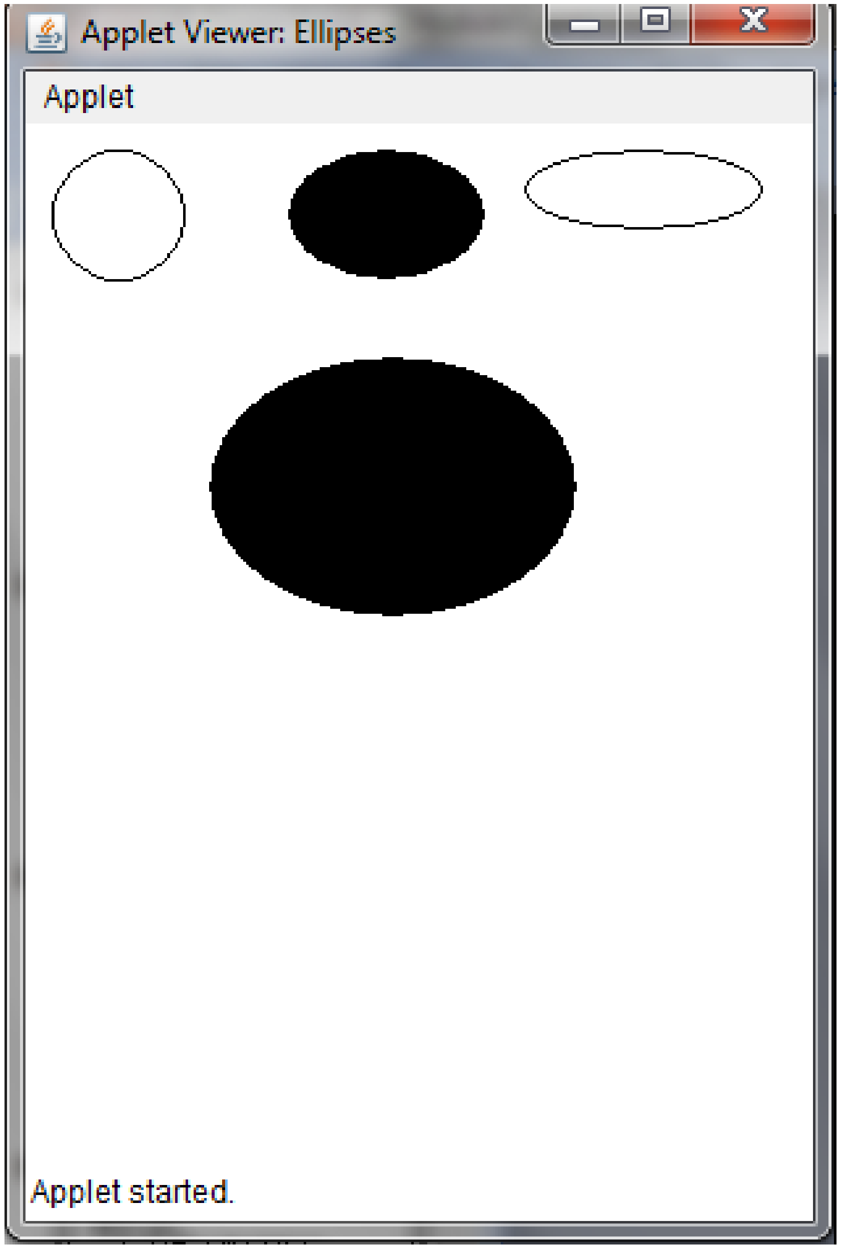
drawOval() – to draw an ellipse

void drawOval(int top, int left, int width, int height)

fillOval()- to fill an ellipse

void fillOval(int top, int left, int width, int height)

```
import java.awt.*;
import java.applet.*;
/* <applet code="Ellipses" width=300 height=400>
</applet>  */
public class Ellipses extends Applet
{
public void paint(Graphics g)
{
g.drawOval(10,10,50,50);
g.fillOval(100,10,75,50);
g.drawOval(190,10,90,30);
g.fillOval(70,90,140,100);
}}
```



Applet started.

Drawing Arcs

Arcs can be drawn with drawArc() and fillArc()

void drawArc(int top, int left,int width, int height, int startAngle,int sweepAngle)

void fillArc(int top, int left,int width, int height, int startAngle,int sweepAngle)

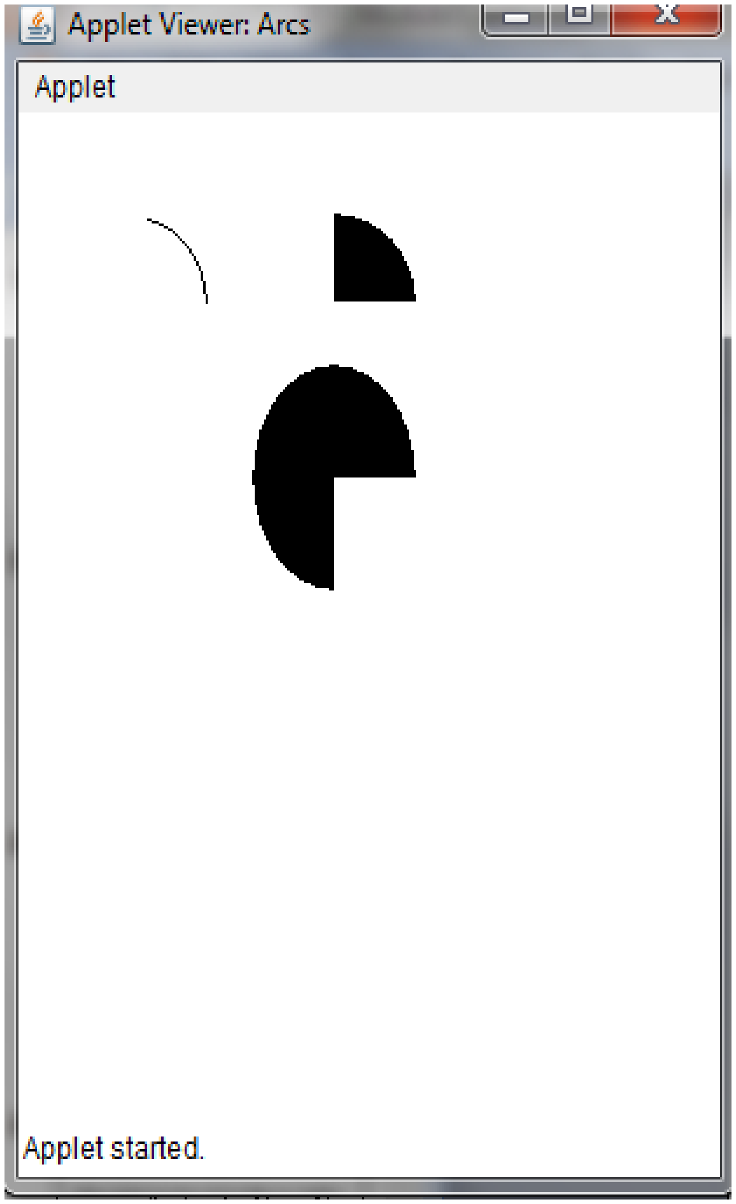
If sweep angle is positive, arc is drawn counterclockwise.

Otherwise clockwise.

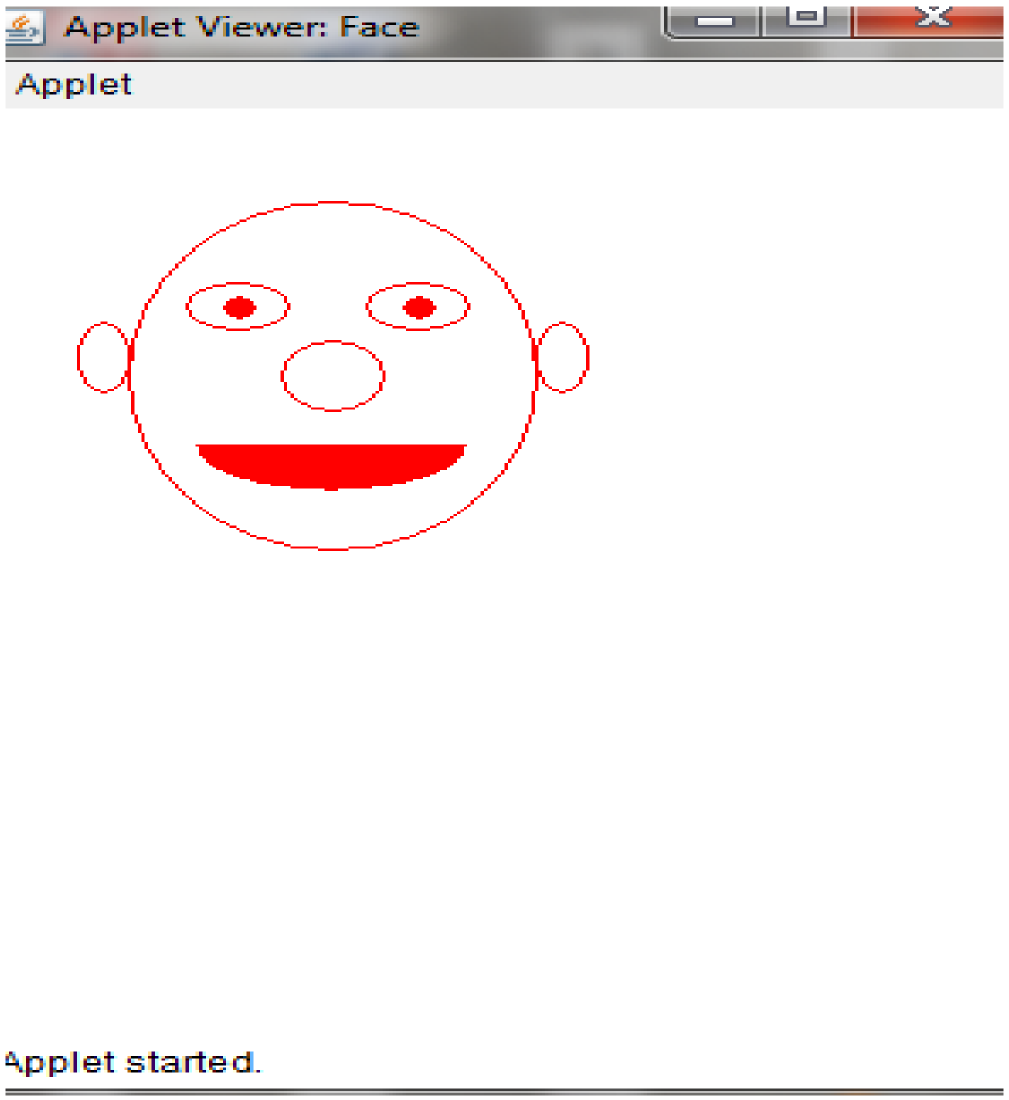
```
import java.awt.*;
import java.applet.*;

/* <applet code="Arcs" width=300 height=400>
</applet>
*/
public class Arcs extends Applet
{
public void paint(Graphics g)
{
g.drawArc(10,40,70,70,0,75);
g.fillArc(100,40,70,70,0,90);
g.fillArc(100,100,70,90,0,270);

}}
```



Write an applet program to draw a human face.



void drawString(String msg,int x,int y)

- Will not recognize newline characters

To set the background color of an applet's window

setBackground()

void setBackground(Color *newColor*)

To set the foreground color

setForeground()

void setForeground(Color *newColor*)

setBackground(Color.green);

setForeground(Color.red);

Color.black	Color.magenta
Color.blue	Color.Orange
Color.cyan	Color.pink
Color.darkGray	Color.red
Color.gray	Color.white
Color.green	Color.yellow
Color.lightGray	

```
Color getBackground();
Color getForeground();
```

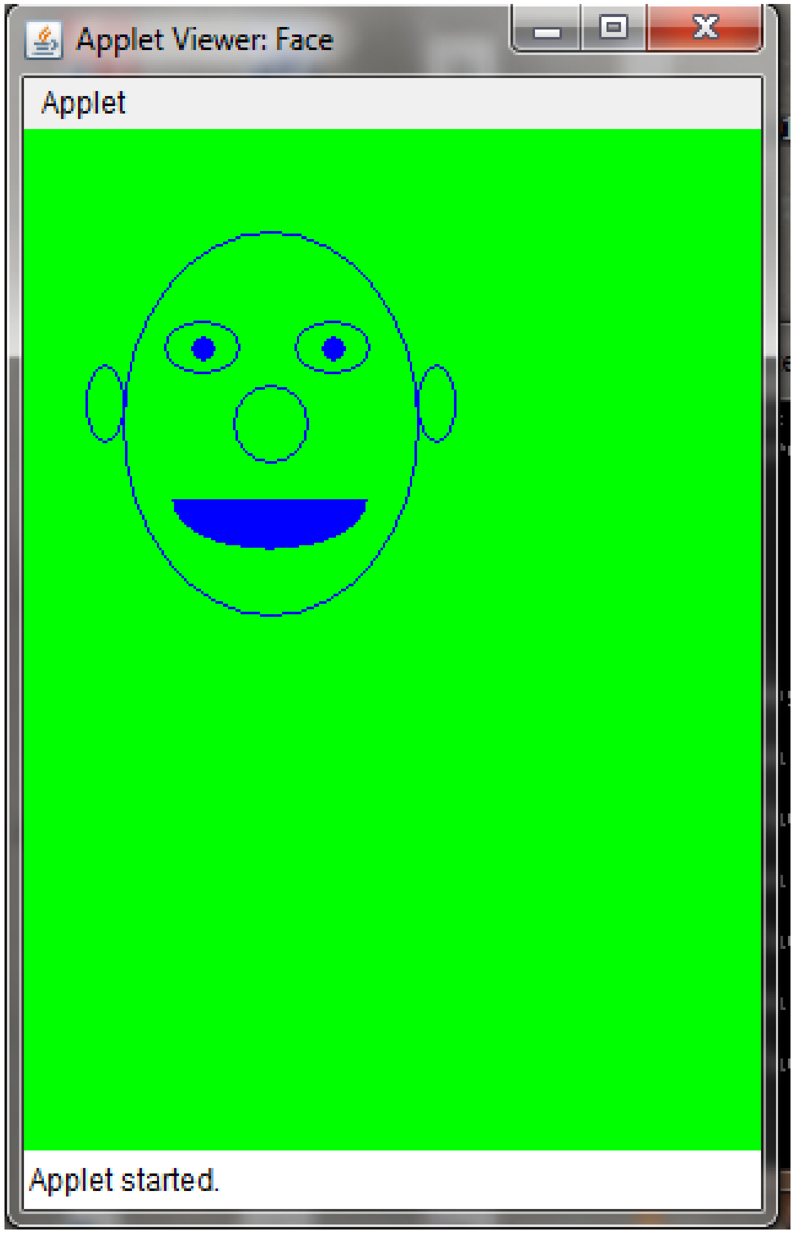
```
g.setColor(Color.red);
g.setColor(Color.green);
```

<u>Requesting Repainting</u>

An applet writes to its window only when its *update()* or *paint()* method is called.

```
void repaint()
void repaint(int left,int top,int width,int height)
```

```
import java.awt.*;
import java.applet.*;
/* <applet code="Face" width=300
      height=400>
</applet>*/
public class Face extends Applet
{public void init()
{
  setBackground(Color.green);
  setForeground(Color.blue);
}
public void paint(Graphics g)
{
//g.setColor(Color.red);
g.drawOval(40,40,120,150);
g.drawOval(57,75,30,20);
g.drawOval(110,75,30,20);
g.fillOval(68,81,10,10);
g.fillOval(121,81,10,10);
g.drawOval(85,100,30,30);
g.fillArc(60,125,80,40,180,18);
g.drawOval(25,92,15,30);
g.drawOval(160,92,15,30);
}}
```



Applet started.

Applet Questions

- Program to draw Circle, Rectangle, Line in an Applet.
- Program to find maximum of three numbers using AWT.
- Find the percentage of marks obtained by a student in 5 subjects. Display a happy face if he secures above 50% or a sad face if otherwise.
- Using 2D graphics commands in an Applet, construct a house. On mouse click event, change the color of the door from blue to red.

- Implement a simple calculator using AWT components.
- Develop a program that has a Choice component which contains the names of shapes such as rectangle, triangle, square and circle. Draw the corresponding shapes for given parameters as per user’s choice.
- Develop a program to handle all mouse events and window events
- Develop a program to handle Key events.

AWT Controls

Controls

- Components that allow a user to interact with an application.

Layout manager

- Automatically positions components within a container.

The AWT supports the following types of controls:

Labels	Lists
Push buttons	Scroll bars
Check boxes	Text Fields

To include a control in a window,add it to the window

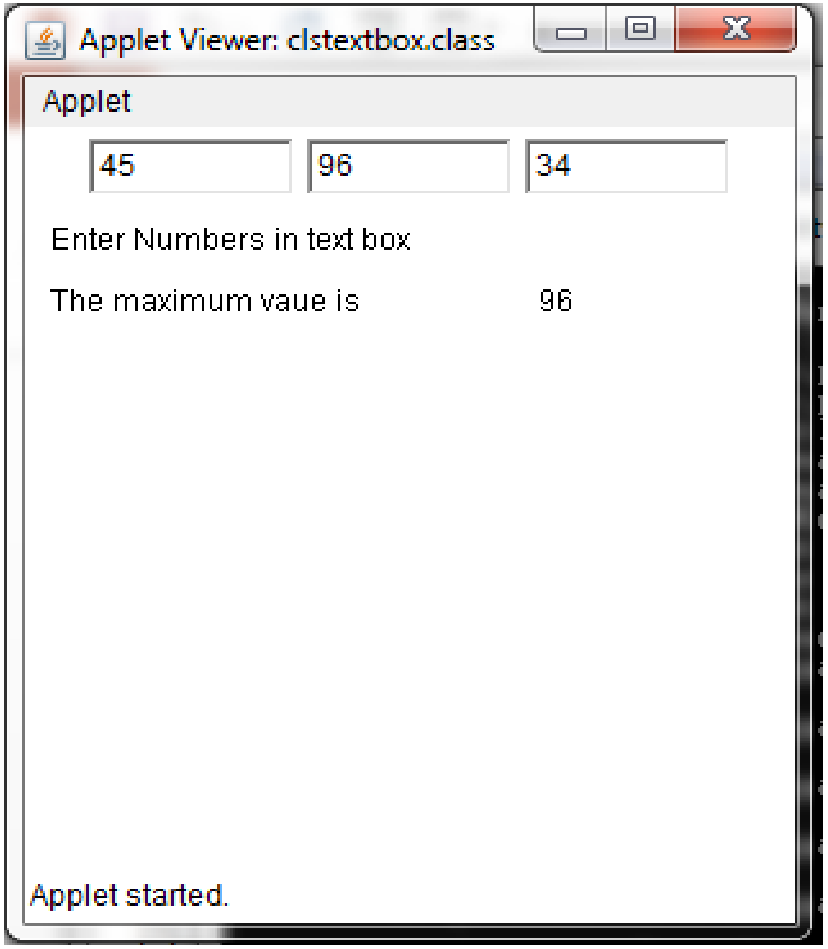
add(…….)

to remove a control

remove(…..)

to remove all controls

removeAll()



```
Enter Numbers in text box

The maximum vaue is                96
```

Applet started.

```
import java.awt.*;
import java.applet.*;
/*<applet code=clstextbox.class width=300  height=300>
</applet>
*/
public class clstextbox extends Applet
{
    TextField txt1,txt2,txt3;
 public void init()
 {
  txt1= new TextField(8);
  txt2= new TextField(8);
  txt3= new TextField(8);
  add(txt1);
  add(txt2);
  add(txt3);
 }
```

```
public void paint(Graphics g)
 {
  int x=0,y=0,z=0;
  String s1,s2,s3;
  g.drawString("Enter Numbers in text box",10,50);
  s1= txt1.getText();
  s2=txt2.getText();
  s3=txt3.getText();
  x=Integer.parseInt(s1);
  y=Integer.parseInt(s2);
  z=Integer.parseInt(s3);
  int w=(x>y? x:y);
  int max=(w>z? w:z);
  g.drawString("The maximum vaue is ",10,75);
  g.drawString(String.valueOf(max),200,75);
```

Buttons

- Most widely used control
- Contains a label and generates an event when it is pressed
- Objects of Button

Button()

Button(String str)

void setLabel(String str)

String getlabel()

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="ButtonDemo" width=250 height=150>
</applet>  */
public class ButtonDemo extends Applet implements
    ActionListener
{   String msg=" ";
    Button yes,no;
 public void init()
 {  yes= new Button("Yes");
    no= new Button("No");
  add(yes);
  add(no);
  yes.addActionListener(this);
  no.addActionListener(this);  }
public void actionPerformed(ActionEvent ae)
 {
  String str=ae.getActionCommand();
  if (str.equals("Yes"))
     {msg=" You have pressed Yes Button";}
  else
     {msg=" You have pressed No Button";}
   repaint();
 }
public void paint(Graphics g)
 {
  g.drawString(msg,10,75);
 }
}
```

Check Boxes

- Control that is used to turn an option on or off
- Consists of a small box that can either contain a check mark or not.

Checkbox()

Checkbox(String str)

Checkbox(String str,boolean on)

Checkbox(String str,boolean on,CheckboxGroup cbGroup)

Checkbox(String str,CheckboxGroup cbGroup, boolean on)

getState() – to retrieve the current state of a checkbox

boolean getState()

To set its state, setState()

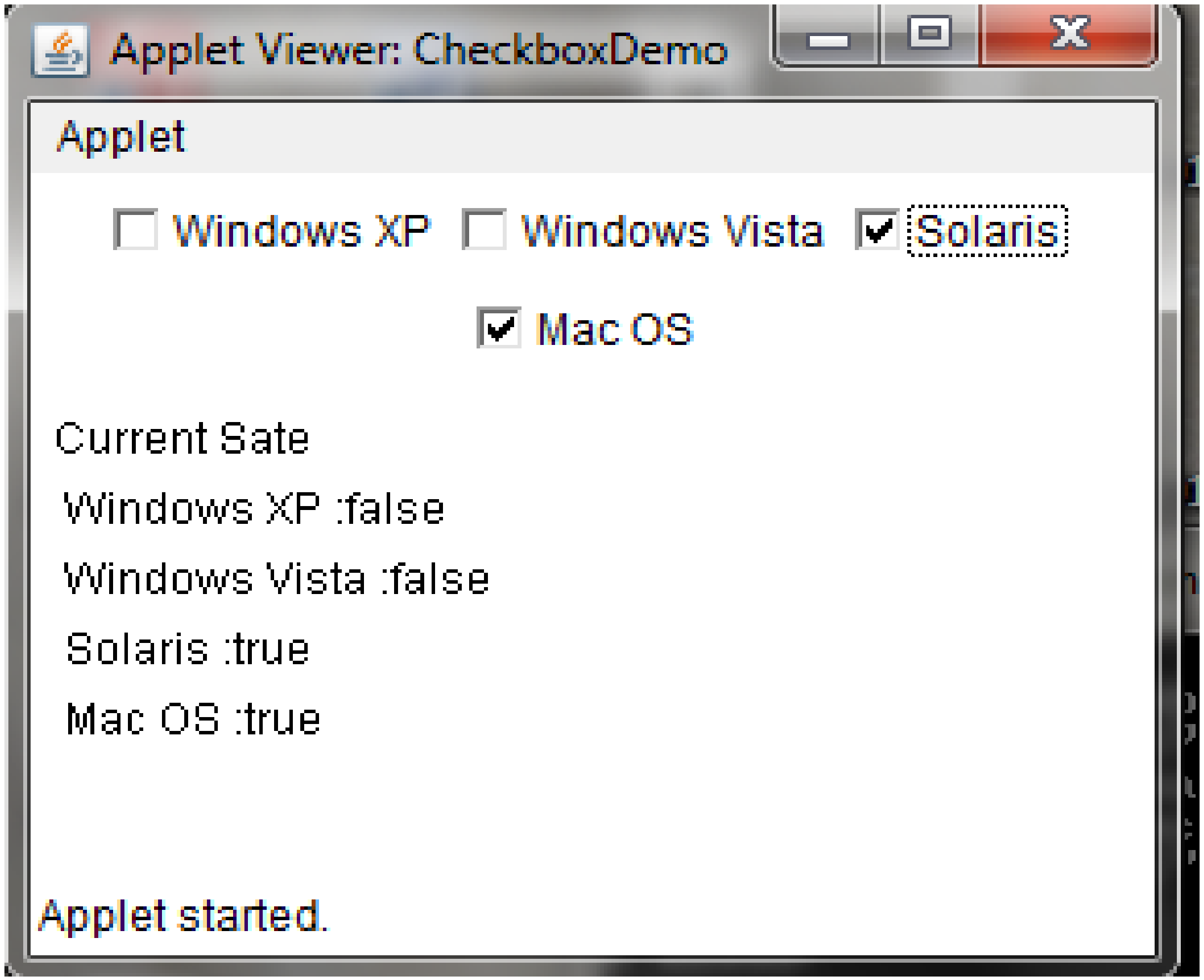
```
void setState(boolean on)
```

To obtain the current label associated with a check box getLabel()

```
String getLabel()
```

To set the Label, setLabel()

```
void setLabel(String str)
```



```
Current Sate
 Windows XP :false
 Windows Vista :false
 Solaris :true
 Mac OS :true
```

Applet started.

public class CheckboxDemo **extends Applet** **implements ItemListener**

```
Checkbox winxp, winvista, solaris, mac;
public void init()
{
winxp=new Checkbox("Windows XP",null,true);
winvista=new Checkbox("Windows Vista");
solaris =new Checkbox("Solaris");
mac = new Checkbox("Mac OS");
add(winxp);
add(winvista);
add(solaris);
add(mac);

winxp.addItemListener(this);
winvista.addItemListener(this);
solaris.addItemListener(this);
mac.addItemListener(this);
}
```

```
public void itemStateChanged(ItemEvent ie)
{
repaint();
}

public void paint(Graphics g)
{msg= "Current Sate";
 g.drawString(msg,6,80);
msg=" Windows XP :"+winxp.getState();
g.drawString(msg,6,100);
msg=" Windows Vista :"+winvista.getState();
g.drawString(msg,6,120);
msg=" Solaris :"+solaris.getState();
g.drawString(msg,6,140);
msg=" Mac OS :"+mac.getState();
g.drawString(msg,6,160);
}
```

CheckboxGroup

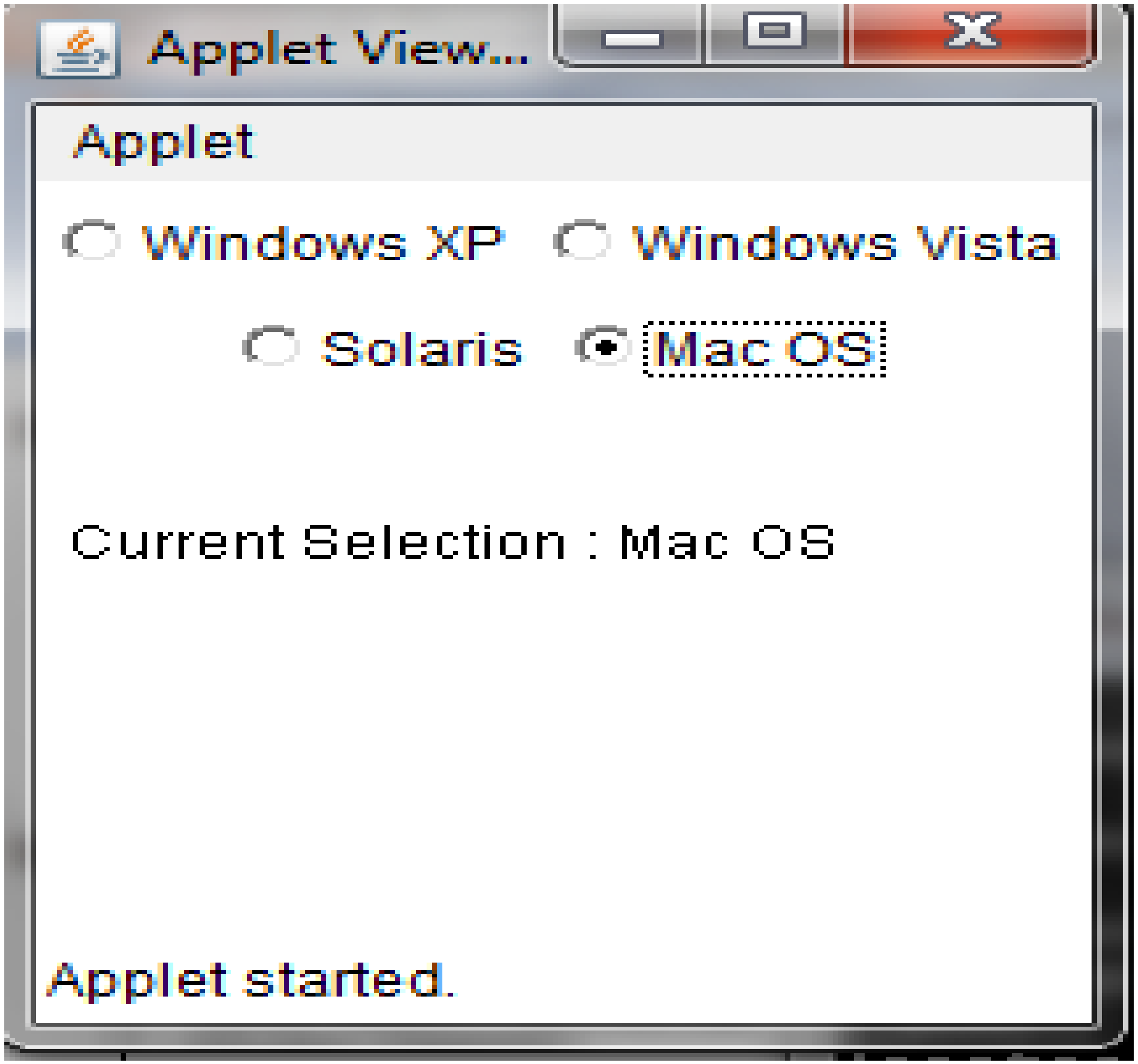
- Radio buttons

- A set of mutually exclusive check boxes in which one and only one checkbox in the group can be checked at any one time.

Checkbox getSelectedCheckbox()

- To determine which checkbox in a group is currently selected
- To set a checkbox setSelectedCheckbox()

void setSelectedCheckbox(Checkbox which)



Current Selection : Mac OS

Applet started.

public class CBGroup extends Applet implements ItemListener

```
Checkbox winxp,winvista,solaris, mac;
CheckboxGroup cbg;
public void init()
{
cbg=new CheckboxGroup();
winxp=new Checkbox("Windows XP",cbg,true);
winvista=new Checkbox("Windows Vista",cbg,false);
solaris =new Checkbox("Solaris",cbg,false);
mac = new Checkbox("Mac OS",cbg,false);
add(winxp);
add(winvista);
add(solaris);
add(mac);
winxp.addItemListener(this);
winvista.addItemListener(this);
solaris.addItemListener(this);
mac.addItemListener(this); }
```

```
public void itemStateChanged(ItemEvent ie)
{
repaint();
}
public void paint(Graphics g)
{
   msg= "Current Selection : ";
   msg +=cbg.getSelectedCheckbox().getLabel();
   g.drawString(msg,6,100);
}
}
```

Choice Controls

- To create a pop-up list of items from which the user may choose.
- To add a selection to the list, call add()

void add(String name)

To determine which item is currently selected

getSelectedItem() **String getSelectedItem()**

getSelectedIndex() **int getSelectedIndex()**

To obtain the nuber of items in the list

getItemCount() int getItemCount()

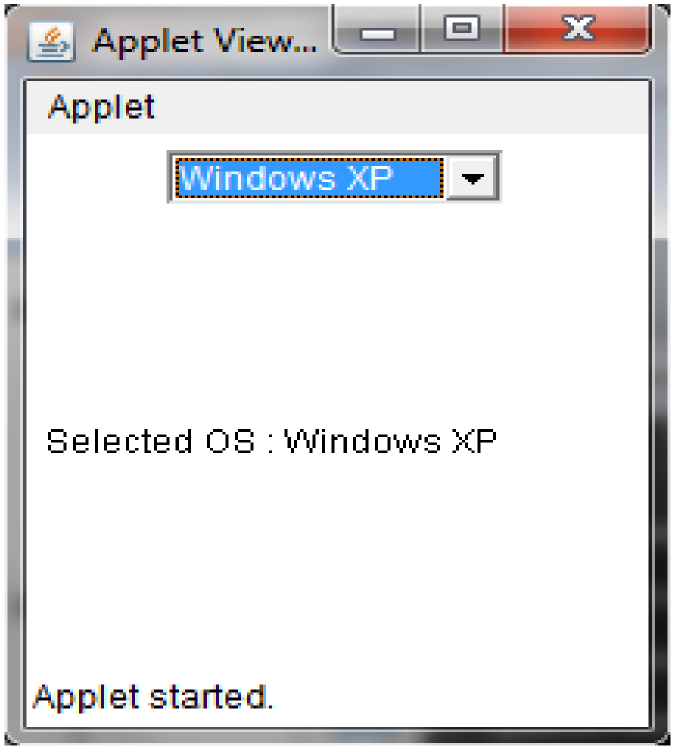
To set the currently selected item ,use select() method

void select(int index)

void select(String name)

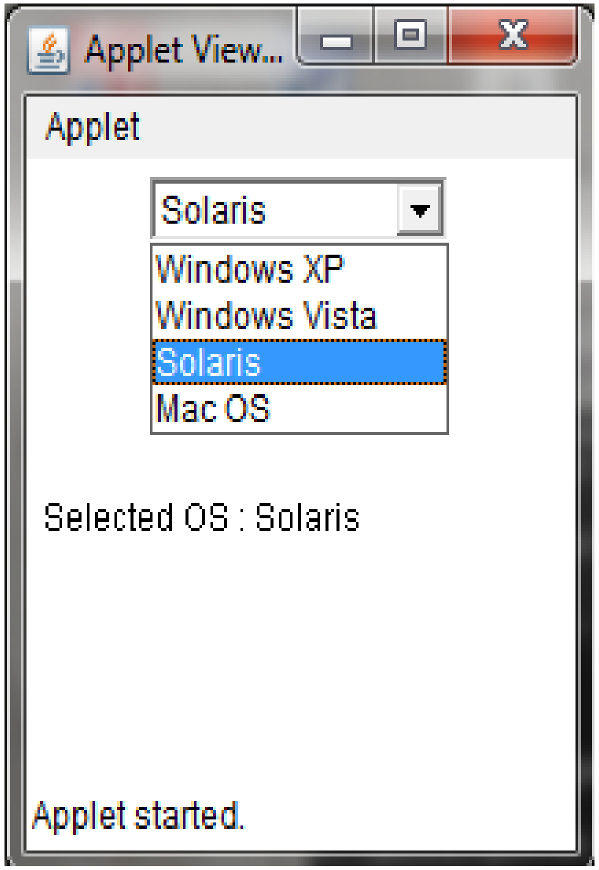
To obtain the name associated with the item at a particular index position –getItem()

String getItem(int index)



Selected OS : Windows XP

Applet started.



Selected OS : Solaris

Applet started.

```
public class ChoiceDemo extends Applet implements ItemListener
{String msg=" ";
Choice OS;
public void init()
{ OS= new Choice();
   OS.add("Windows XP");
   OS.add("Windows Vista");
   OS.add("Solaris");
   OS.add("Mac OS");
   add(OS);
   OS.addItemListener(this); }
public void itemStateChanged(ItemEvent ie)
{repaint(); }
public void paint(Graphics g)
{  msg= "Selected OS : ";
    msg +=OS.getSelectedItem();
   g.drawString(msg,6,120); }}
```

Event Handling

- Applets are event-driven programs that use a graphical user interface to interact with the user.

- java.awt.event

- Main Event classes and Interfaces used by the AWT.

- Event handling is based on the *delegation event model*.

Delegation Event Model

- A source generates an event and sends it to one or more listeners.
- Listener waits until it receives an event.
- Once an event is received, the listener processes the event and returns.
- Listener must register with a source in order to receive an event notification.

Event – an object that describe a state change in a source.

Source(event source) – an object that generates an event.

- may generate more than one type of event.

- Source must register listeners to receive notifications about a specific type of event.

public void add*Type*Listener(*Type*Listener el)

Listener – an object that is notified when an event occurs.

1. Must have been registered with one or more sources to receive notification about specific types of events.
2. Implement methods to receive and process these notifications.

Event classes

ActionEvent class

- Generated when a button is pressed, a list item is double-clicked or menu item is selected.

  String getActionCommand( )

  - returns the label on the button

ItemEvent class

- Generated when a checkbox or a list item is clicked or when a ckeckable menu item is selected or deselected.

AdjustmentEvent class

- Generated by a scroll bar

FocusEvent class

- Generated when a component gains or loses input focus.

KeyEvent class

- Generated when keyboard input occurs.

MouseEvent class

TextEvent class – generated by text fields and text areas when characters are entered by a user

Event Listener Interfaces

ActionListener Interface

void actionPerformed(ActionEvent ae)

- invoked when an action event occurs.

AdjustmentListener Interface

void adjustmentValueChanged(AdjustmentEvent ae)

- invoked when an adjustment event occurs.

FocusListener Interface

void focusGained(FocusEvent fe)

void focusLost(FocusEvent fe)

ItemListener Interface

void itemStateChanged(ItemEvent ie)

- invoked when the state of an item changes.

KeyListener Interface

void keyPressed(KeyEvent ke)

void keyReleased(KeyEvent ke)

void keyTyped(KeyEvent ke)

The MouseListener Interface

void mouseClicked(MouseEvent me)

void mouseEntered(MouseEvent me)

void mouseExited(MouseEvent me)

void mousePressed(MouseEvent me)

void mouseReleased(MouseEvent me)

The MouseMotionListener Interface

void mouseDragged(MouseEvent me)

void mouseMoved(MouseEvent me)

TextListener Interface

void textChanged(TextEvent te)