# Applets and Graphics

## Introduction

- One of the most important features of Java is its ability to draw graphics.
- We can write Java applets that can draw lines, figures of different shapes, images, and text in different fonts, styles, and colours.
- Every applet has its own area on the screen known as *canvas*, where it creates display.
- Java coordinate system has the origin (0,0) in the upper-left corner. Positive x values are to the right and +ve y values to the bottom. The values of (x,y) are in pixels.

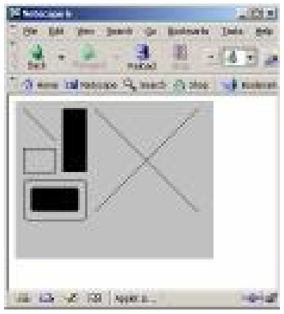
## Graphics Class: Methods include

- drawArc() – draws a hollow arc
- drawLine() – draws a straight line
  - g.drawLine(x1,y1,x2,y2)
- drawOval() - draws a hollow oval
  - g.drawLine(x,y,width, height)
  - If width and height are same, it draws a circle
- drawPolygon() - draws a hollow polygon
- drawRect() - draws a hollow rectangle
  - g.drawLine(x,y,width,height)
- drawRoundRect() - draws a hollow round cornered rectangle
- drawString() – display a text string
- fillArc()  - draw a filled arc
- fillOval()
- fillPolygon()
- fillRect()
- fillRoundRect()
- getColor – retrieve the current drawing colour
- getFont
- setColor – sets the drawing color
- setFont
- More at http://java.sun.com/products/jdk/1.2/docs/api/java/awt/Graphics.html

## Drawing Lines and Rectagles

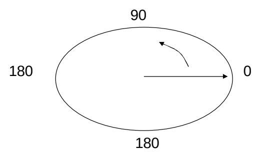
```
import java.awt.*;
import java.applet.*;

public class LineRect extends Applet
{
        public void paint(Graphics g)
        {
                g.drawLine(10,10,50,50);
                g.drawRect(10,60,40,30);
                g.fillRect(60,10,30,80);
                g.drawRoundRect(10,100,80,50,10,10);
                g.fillRoundRect(20,110,60,30,5,5);
                g.drawLine(100,10,230,140);
                g.drawLine(100,140,230,10);
        }
}
```

## Output of LineRect applet



## Drawing arc.

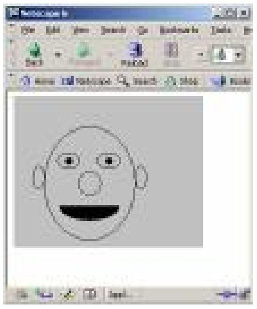
- **drawArc**(int x, int y, int width, int height, int startAngle, int arcAngle)
- Draws the outline of a circular or elliptical arc covering the specified rectangle.
- Java considers 3 O'clock as 0 degree position and degree increases in anti-clock wise direction.



## Drawing a Happy Face Applet

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

public class Face extends Applet
{
    public void paint(Graphics g)
    {
        g.drawOval(40,40,120,150);          //Head
        g.drawOval(57,75,30,20);            //Left eye
        g.drawOval(110,75,30,20);           //Right eye
        g.fillOval(68,81,10,10);            //Pupil (left)
        g.fillOval(121,81,10,10);           //Pupil (right)
        g.drawOval(85,100,30,30);           //Nose
        g.fillArc(60,125,80,40,180,180);    //Mouth
        g.drawOval(25,92,15,30);            //Left ear
        g.drawOval(160,92,15,30);            //Right ear
    }
}
```

## ☺ Output!



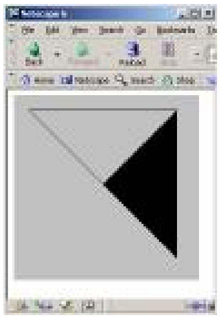
## Drawing Polygons

- Polygons are shapes with many sides. A polygons may be defined as a set of connected lines. The end of first line is the beginning of second line, and so on,
- Syntax:
  - drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
  - Draws a closed polygon defined by arrays of x and y coordinates.

## Polygon example code

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

public class Poly extends Applet
{
    int x1[]={20,120,220,20};
    int y1[]={20,120,20,20};
    int n1=4;
    int x2[]={120,220,220,120};
    int y2[]={120,20,220,120};
    int n2=4;

    public void paint(Graphics g)
    {
        g.drawPolygon(x1,y1,n1);
        g.fillPolygon(x2,y2,n2);
    }
}
```

## Polygon output



## Drawing Bar Charts and Reading Parameters passed via HTML

- Applets can be designed to display bar charts, which are commonly used in comparative analysis of data.
- For example, the Table below shows annual turnover of a company during the period 2000-2003.

| Year | 2000 | 2001 | 2002 | 2003 |
|---|---|---|---|---|
| Turnover | 110 | 150 | 100 | 170 |

- These values can be passed via HTML file as PARAM attributes.

## Bar chart applet program…

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

public class BarChart extends Applet
{
        int n=0;
        String label[];
        int value[];
        public void init()
        {
                try
                {
                        n=Integer.parseInt(getParameter("columns"));
                        label=new String[n];
                        value=new int[n];
                        label[0]=getParameter("label1");
                        label[1]=getParameter("label2");
                        label[2]=getParameter("label3");
                        label[3]=getParameter("label4");

                        value[0]=Integer.parseInt(getParameter("c1"));
                        value[1]=Integer.parseInt(getParameter("c2"));
                        value[2]=Integer.parseInt(getParameter("c3"));
                        value[3]=Integer.parseInt(getParameter("c4"));
                }catch(NumberFormatException e){}
        }
```

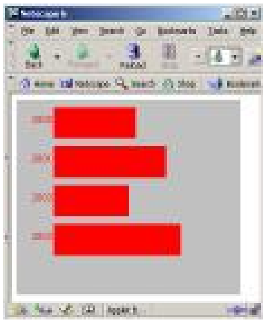
## Bar chart applet program.

```
        public void paint(Graphics g)
         {
                for(int i=0;i<n;i++)
                {
                        g.setColor(Color.red);
                        g.drawString(label[i],20,i*50+30);
                        g.fillRect(50,i*50+10,value[i],40);
                }
         }
}
```

## HTML file – BarChar.html

```
<HTML>
        <APPLET
                CODE=BarChart.class
                WIDTH=300
                HEIGHT=250>

        <PARAM NAME="columns"
   VALUE="4">
```

## Output



## Summary

- Java's *Graphics* class supports many methods that enable us to draw many types of shapes. These methods can be used put together visual display and graphical illustrations.
- Java provide many more capabilities (such as Swings), we are not able cover all of them. For more info, please refer to Sun Java 2 document.
- Future lectures will mostly focus on UML!