**Java Applets**

Reference: Gaddis pages 880-894

Intended not to run as standalone program but by

You’ve probably seen one running before!

/\* An applet shell by Anderson, Franceschi \*/

import javax.swing.JApplet;

import java.awt.Graphics; //part of Abstract Window Toolkit

**public class ShellApplet extends JApplet**

{

// declare variables here needed throughout

public void **init**( )

{

// initialize data here

}

public void **paint**( Graphics g )

{

super.paint( g );

// include graphics code here

}

}

public class ShellApplet extends JApplet

* JApplet is a class with many methods.
* “extends” means that our class is a of JApplet and will those methods

public void init( ) and public void paint(Graphics g)

* These are the 2 standard methods of an applet.
* ***init*** is used to initialize variables (optional)
* ***paint*** is automatically called and sent an automatically generated Graphics object
* THERE IS NO main METHOD!

**Running an Applet**

How to run it in JGrasp:

Run code as an Applet by going to Build menu, then “Run As Applet”

How to run in a web browser:

You need to embed the name of the compiled Applet in HTML code—see Ch.14

**What code goes in the “paint” method?**

We’ll use Java Graphics

What does the below code do??

g.drawRect (50, 50, 40, 40); // square

g.drawRect (60, 80, 225, 30); // rectangle

g.drawOval (75, 65, 20, 20); // circle

g.drawLine (35, 60, 100, 120); // line

g.drawString ("Out of clutter, find simplicity.", 110, 70);

g.drawString ("-- Albert Einstein", 130, 100);

**How does it work?**

If you are using graphics in an applet, the *paint* method is automatically called and a Graphics object is automatically created when the applet runs

public void **paint**( Graphics g )

{

super.paint( g );

// include graphics code here

}

Be sure to import BOTH the java.awt.Graphics and the javax.swing.JApplet classes

The java.awt package has many predefined classes available for the production of Graphics

Graphical objects are positioned using an x-y coordinate system where

(0,0) x increases🡪 (400,0)

y

i

n

c

↓

(0,400) (400,400)

**Some Methods (see page 881 for a full list)**

drawLine (int xStart, int yStart, int xEnd, int yEnd)

* **g**.drawLine ( 10, 10, 50, 10 );

drawRect ( int x, int y, int width, int length)

* **g**.drawRect ( 200, 100, 50, 100 );

drawOval(int x, int y, int width, int height)

* **g**. drawOval(100, 100, 50, 100);

All can be modified to fill in:

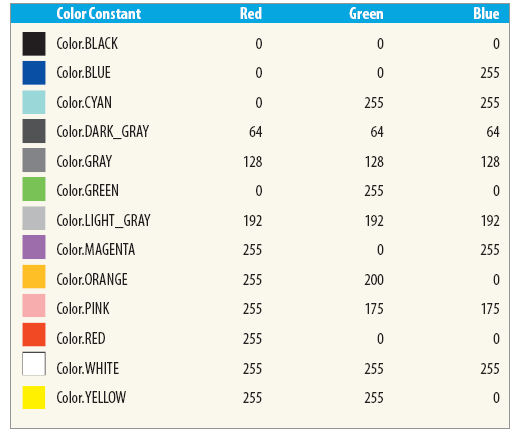
**Color**

There are 13 static constants for color

Everything you draw after setting a color will be in that color, until you set a new color

g.setColor (Color.BLUE);

g.drawOval (100, 200, 150, 80);



MCj02807400000[1]Mix Your Own Color!

Every color is composed of red, green and blue components in (r, g, b) order.

Each component is in the range 0..255

Color myColor = new Color (0, 0, 255); // what’s the color?

g.setColor (myColor);

g.fillOval ( 100, 200, 80, 150);

Color myColor = new Color (255, 255, 0 ); // what’s the color?

g.setColor (myColor);

g.drawOval ( 100, 200, 80, 150);

public class AstronautHead extends JApplet

{

public void paint( Graphics g )

{

super.paint( g );

// instantiate a custom color

Color spacesuit = new Color( 195, 175, 150 );

int sX = 100; // the starting x position

int sY = 25; // the starting y position

// helmet

g.setColor( spacesuit );

g.fillOval( sX + 60, sY, 75, 75 );

g.setColor( Color.LIGHT\_GRAY );

g.fillOval( sX + 70, sY + 10, 55, 55 );

// face

g.setColor( Color.DARK\_GRAY );

g.drawOval( sX + 83, sY + 27, 8, 8 );

g.drawOval( sX + 103, sY + 27, 8, 8 );

g.drawLine( sX + 97, sY + 35, sX + 99, sY + 43 );

g.drawLine( sX + 97, sY + 43, sX + 99, sY + 43 );

g.drawOval( sX + 90, sY + 48, 15, 6 );

}

}