

Graphics and Methods

COMP163

*“Civilization is a method of living,
an attitude of equal respect for all
people.”*

Jane Addams

first American woman to be awarded the Nobel Peace Prize

Homework

- The weekly programming assignment is posted on Blackboard
- Program is due by midnight on **Friday**

GUI Methods

- Important methods for a GUI program include
 - main
 - Creates an instance of the class
 - constructor (same name as class)
 - Initializes the GUI
 - Only called once
 - actionPerformed
 - Does what the program is supposed to do
 - Called whenever something is clicked
 - paint (*optional*)
 - Displays graphics
 - Called many times

paint Method

- The Java system calls the paint method when it wants to display the GUI

```
public void paint(java.awt.Graphics peacock ) {  
    peacock.setColor(java.awt.Color.BLUE);  
    peacock.fillRect( 10, 20, 100, 150 );  
}
```

- The paint method may be called many times, particularly when you resize the window

Graphics Class

- The parameter to the paint method of a Frame is an object of the `java.awt.Graphics` class
- There are many methods for Graphics objects to draw simple pictures
- Most paint methods will contain several calls to methods of the Graphics parameter

Graphics Methods

- There are many methods that can be used to draw simple shapes on the screen
- Most Graphics methods do not return a value
- You can set the color to be used to draw a shape. This color will be used for all shapes until you change the color
- *These Graphics methods are **not** effective tools for drawing cool interactive graphics*

Colors

- The `java.awt.Color` class defines colors
- You can define a color using RGB values or you can use a predefined constant
- Some Color constants are:
 - `java.awt.Color.BLUE`
 - `java.awt.Color.GREEN`
 - `java.awt.Color.ORANGE`
 - `java.awt.Color.GRAY`
 - `java.awt.Color.RED`

setColor

- The setColor method of java.awt.Graphics objects sets the color that will be used when drawing shapes

```
bird.setColor(java.awt.Color.BLUE) ;
```

- where **bird** is an object of the type java.awt.Graphics
- Once you set the **color, it stays that color**

Graphics Coordinates

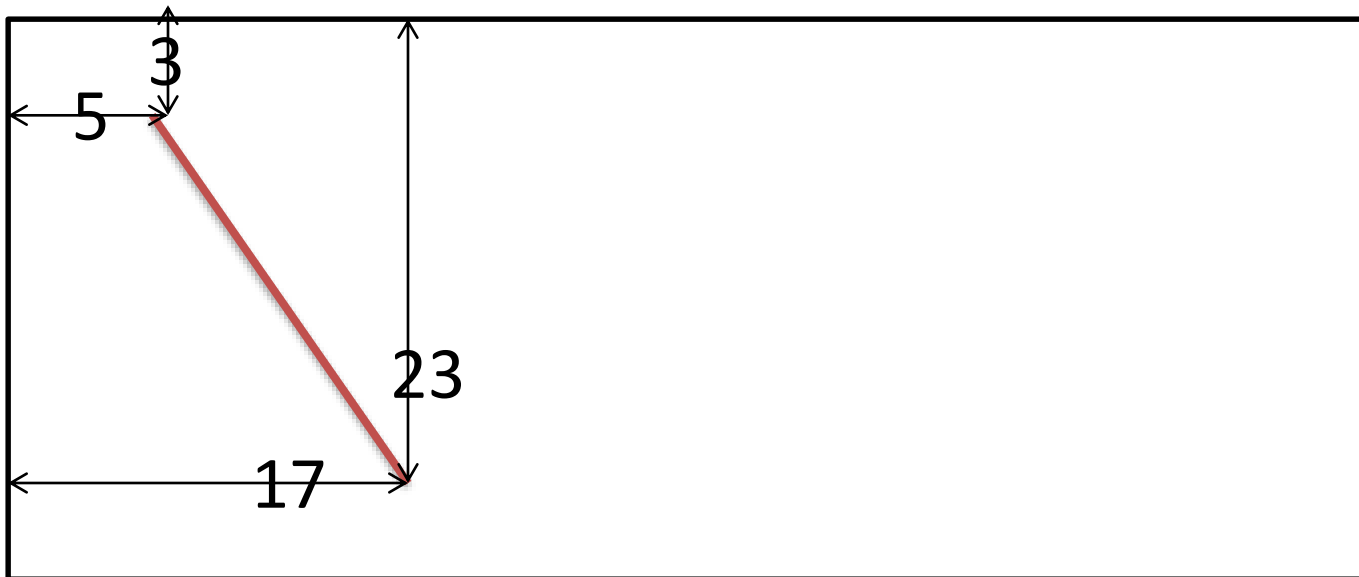
- The screen has a coordinate system with the origin in the upper left corner
- Coordinates are given in pixels (Picture Elements)
- An X coordinate specifies the distance from the left edge
- A Y coordinate specified the distance from the top edge
- The screen size depends on the device

drawLine

- The drawLine method of java.awt.Graphics objects draws a line from x_1, y_1 to x_2, y_2

```
bird.drawLine(5, 3, 17, 23);
```

- The line is drawn using the current color



A. upper left corner

C. upper right corner

Location 0,0 is

D. center

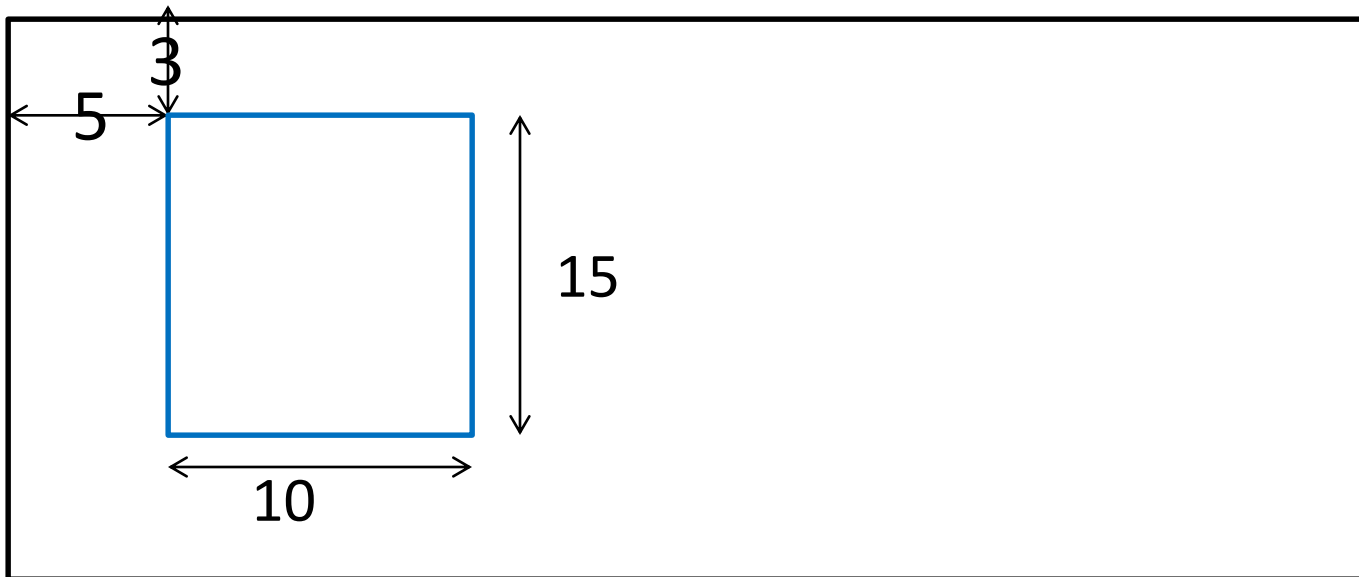
B. lower left corner

drawRect

- The drawRect method of java.awt.Graphics objects draws a rectangle of width and height whose upper left corner is x,y

```
bird.drawRect( x, y, width, height );
```

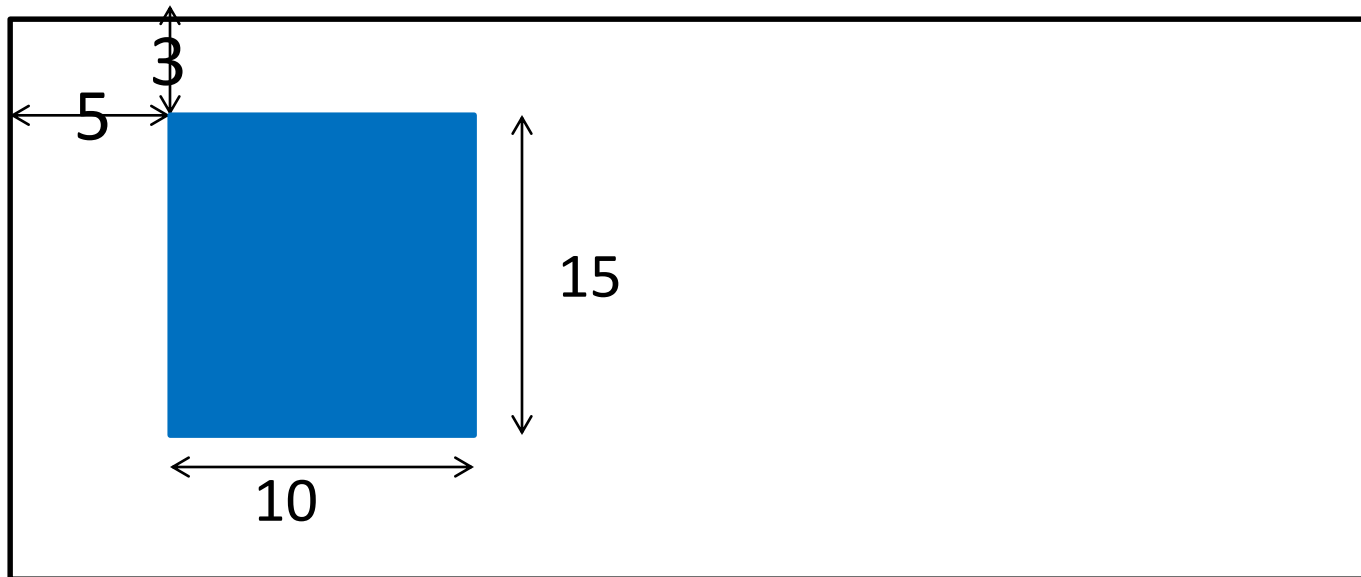
```
bird.drawRect( 5, 3, 10, 15 );
```



fillRect

- The fillRect method of java.awt.Graphics objects draws a rectangle of width and height whose upper left corner is x,y and fills it with the current color

```
bird.fillRect( 5, 3, 10, 15 );
```



Outside and Overlapping

- It is permissible to have a graphics object (such as a rectangle) extend outside the visible window
- The portion outside the window will not be visible
- An object can overlap one another

```
bird.fillRect( 5, 3, 10, 15 );
```

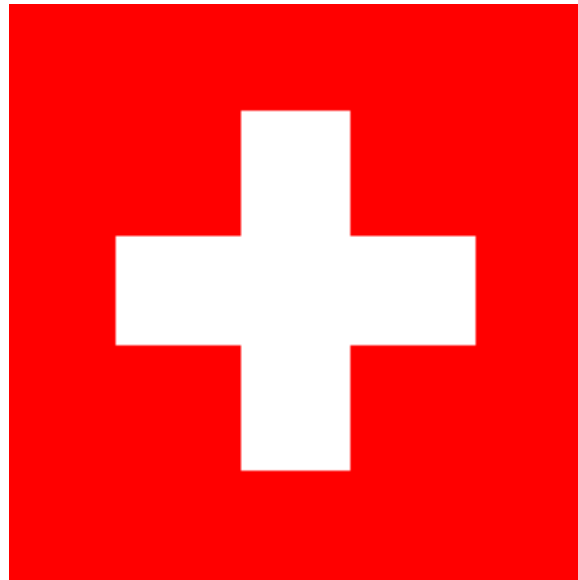
```
bird.setColor( Color.GREEN );
```

```
bird.fillRect( 8, 15, 10, 15 );
```



Graphics Example

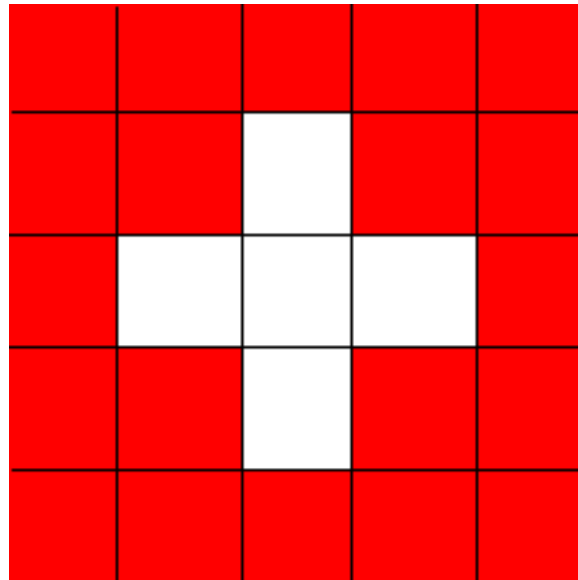
- We want to write a paint method that will draw the flag of Switzerland



- The flag image will be 150 by 150 pixels

Thinking about the picture

- We can draw the flag by setting the background red and drawing two white rectangles



- Think of the flag as a grid of 30 by 30 squares

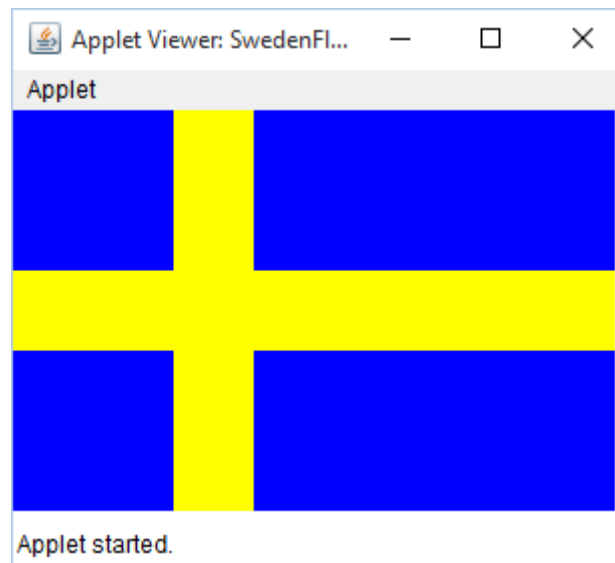
Possible Solution

```
public void paint(java.awt.Graphics canvas) {  
    canvas.setColor(java.awt.Color.RED);  
    canvas.fillRect(0, 0, 150, 150);  
    canvas.setColor(java.awt.Color.WHITE);  
    canvas.fillRect(60, 30, 30, 90);  
    canvas.fillRect(30, 60, 90, 30);  
}
```

Work with the students near you

- Complete this program to draw a Swedish flag that is 300 by 200 pixels

```
public class Sweden extends javax.swing.JFrame {  
    public void paint( java.awt.Graphics page) {
```



Possible Solution

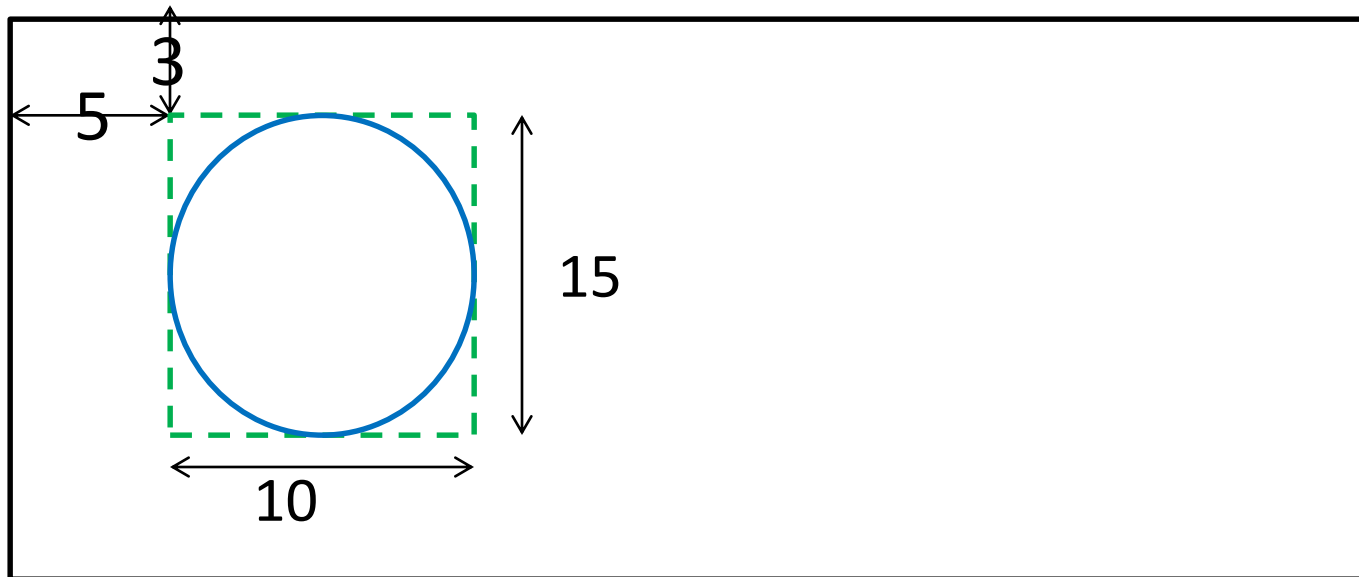
```
public class Sweden extends
    javax.swing.JFrame {
    public void paint(java.awt.Graphics page) {
        page.setColor(java.awt.Color.BLUE);
        page.fillRect(0, 0, 300, 200);
        page.setColor(java.awt.Color.YELLOW);
        page.fillRect(0, 80, 300, 40);
        page.fillRect(80, 0, 40, 200);
    }
}
```

drawOval

- The drawOval method of java.awt.Graphics objects draws a circle or oval to fit in a box of width and height whose upper left is x,y

```
bird.drawOval( x, y, width, height );
```

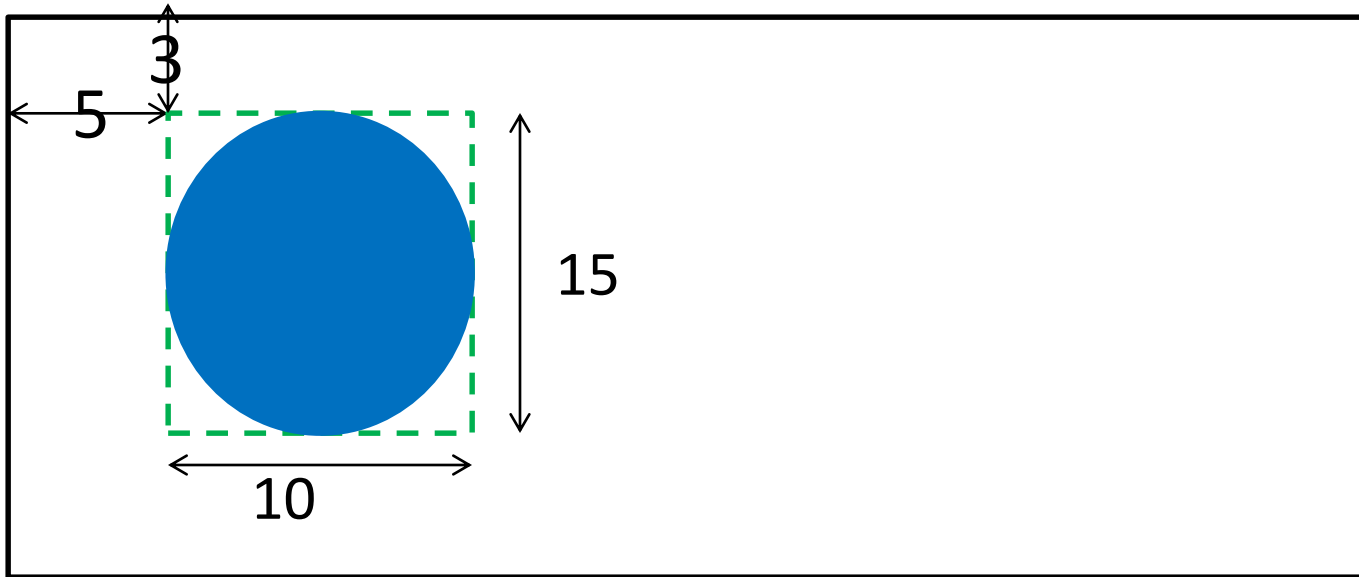
```
bird.drawOval( 5, 3, 10, 15 );
```



fillOval

- fillOval is just like drawOval, but it colors in the circle with the current color

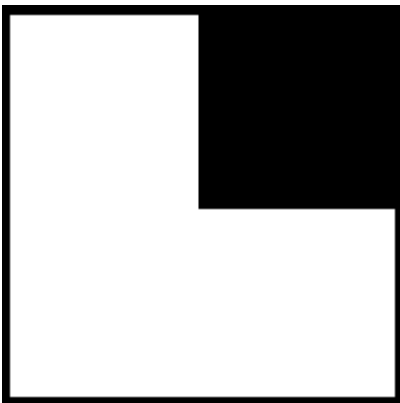
```
bird.fillOval( 5, 3, 10, 15 );
```



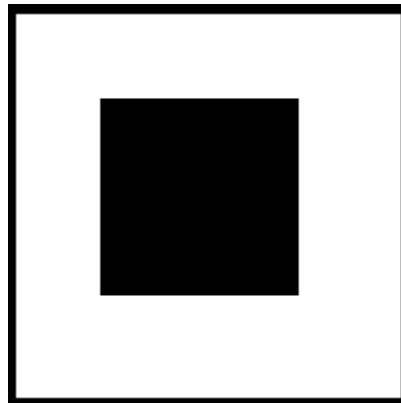
What is the output?

```
public void paint(java.awt.Graphics parrot) {  
    parrot.setColor(java.awt.Color.BLACK);  
    parrot.drawRect( 0, 0, 100, 100);  
    parrot.fillRect( 50, 0, 50, 50 );  
}
```

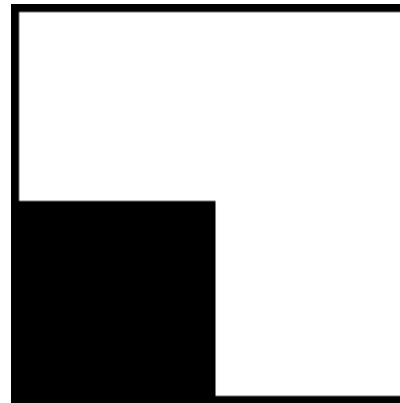
A.



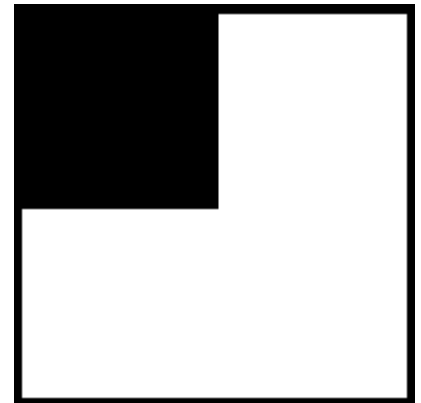
B.



C.



D.

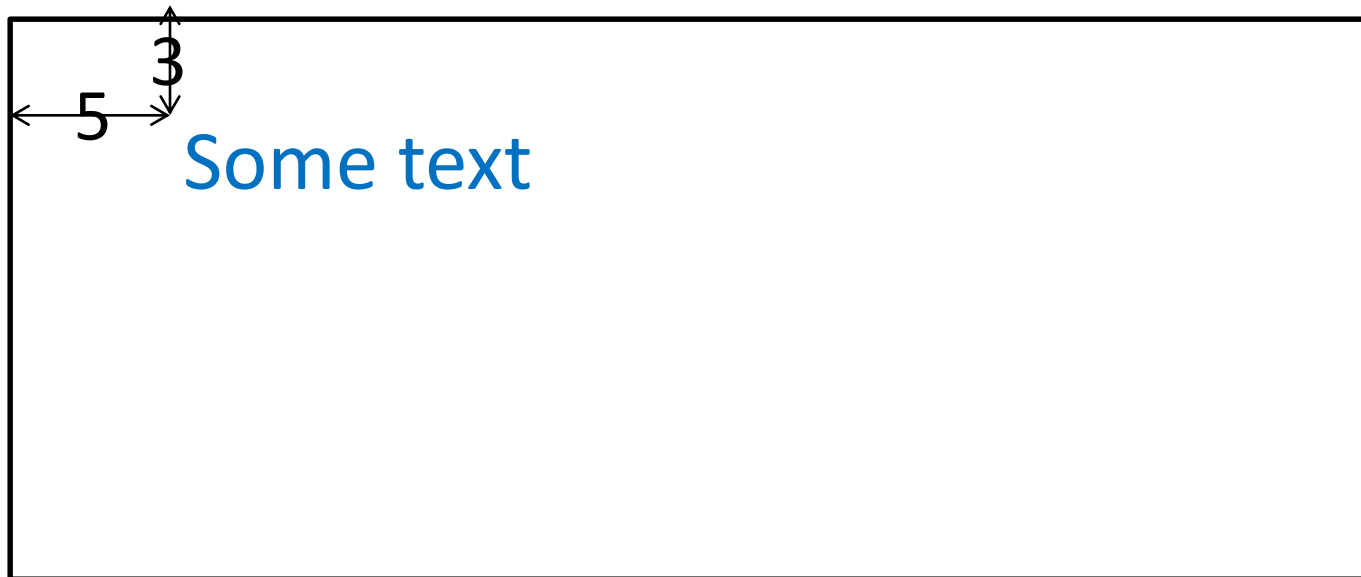


drawString

- The drawString method of java.awt.Graphics objects writes the text of a String starting at the specified x,y location

```
bird.drawString( String, x, y );
```

```
bird.drawString( "Some text", 5, 3 );
```



Example Picture Program

```
public class TestDraw extends javax.swing.JFrame {
    int callCount = 1; // count of the calls to paint

    public void paint( java.awt.Graphics page) {
        System.out.println("Painting " + callCount++);

        page.setColor(java.awt.Color.YELLOW);
        page.fillOval(10, 45, 200, 175 ); // draw head
        page.setColor(java.awt.Color.BLUE);
        page.fillOval(60, 100, 20, 20 ); // draw eyes
        page.fillOval(140, 100, 20, 20 );
        page.setColor(java.awt.Color.BLACK);
        page.drawArc(35, 40, 150, 150, 200, 140 ); //draw smile
        page.drawString("A very simple picture", 50, 240);
    }

    public TestDraw() {
        setSize(300, 300);
        setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        setVisible( true );
    }

    public static void main(String[] unused) {
        new TestDraw();
    }
}
```

Calling Methods

- The graphics programs tend to call many methods
- Most graphics methods do not return a value
- They are used on a line by themselves
- Methods that return a value are often used in an equation

Methods with or without return value

- Value returning method

```
int blue = getBlue( x, y );
```

- void method that does not return a value

```
setBlue( x, y, blue );
```

Calling Methods of Classes

- Static methods are called on a class

Classname.method()

- `Math.cos(dog)` is a static method

We have been using methods

- Homework and lab assignments have used methods

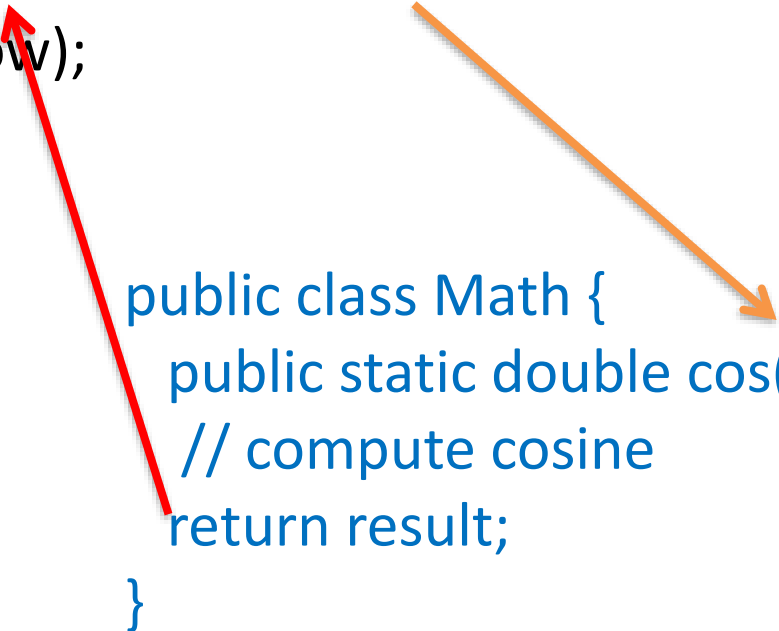
```
cow = 2.0 * Math.cos( theta );
```

- During the execution of your program, it executes the cosine method which returns the result back to your program

Transfer of Execution

```
public class Demo {  
    public static void main( String[ ] stuff) {  
        double theta = 1.047;  
        double cow = 2.0*Math.cos(theta);  
        System.out.println(cow);  
    }  
}
```

```
public class Math {  
    public static double cos(double xyz) {  
        // compute cosine  
        return result;  
    }  
}
```



Calling Methods of Objects

- Non-static methods are called on objects, not classes

```
public void paint(java.awt.Graphics cow ) {  
    cow.setColor(java.awt.Color.PURPLE);  
    cow.fillOval( 10, 25, 7, 7);  
}
```

Class Variables

```
public class Reptile {  
    int skink = 11;  
    int doit( int toad ) {  
        int lizard = skink + toad;  
        return lizard;  
    }  
}
```

- **skink** is an instance variable
- **toad** is a parameter variable
- **lizard** is a local variable

Instance Variables

- An instance variable (a.k.a. field variable, class variable or attribute) hold long term data
- Each object has its own copy of the instance variables
- Objects hold the data of a logical entity

Local Variables

- Local variables are defined in a method
- The values of local variables disappear when the program returns

Parameter Variables

- Parameter variables have an initial value from the calling program
- Parameters are the input to a method
- Changing simple parameter variables does not change the calling program's variables

Making a GUI

1. Create a class that extends JFrame and implements `java.awt.event.ActionListener`
2. Create component objects
3. Specify the layout manager
4. Add the component objects to the container
5. Add an action listener to the components that respond to user input
6. All the program to be visible and stop
7. Create an `actionPerformed` method

Which is a local variable?

```
int aardvark;
```

```
public class Click {
```

```
    int bunny;
```

```
    public void aMethod( int cow ) {
```

```
        int dog;
```

```
    }
```

```
}
```

A. aardvark

B. bunny

C. cow

D. dog

Usual Java GUI

```
public class MyProgram extends javax.swing.JFrame
    implements java.awt.event.ActionListener {
    // declare GUI components here
    public MyProgram() {
        // setup GUI here
    }
    public void actionPerformed(
        java.awt.event.ActionEvent dog){
        // program here
    }
    public static void main(String[] cat) {
        MyProgram fish = new MyProgram();
    }
}
```

GUI Display

- GUI programs usually set the text of a JLabel

// in the class before any methods

```
JLabel hawk = new JLabel();
```

// in the actionPerformed method

```
double dog = Math.sin(cat);
```

```
hawk.setText("The answer is " + dog);
```

If a Frame calls `System.out.println`
the program will

- A. compiler error
- B. run time error
- C. display the data in the usual
place for applications
- D. do nothing

GUI Text Input

- An GUI usually gets text input from a JTextField

// in the class before any methods

```
JTextField kitten = new JTextField();
```

// in the actionPerformed method

```
String bear;
```

```
bear = kitten.getText();
```

String and Numbers

- A String can contain any character on the keyboard, including the numbers

```
String myGrade = "4.0";
```

```
double myScore = 4.0;
```

- You cannot do arithmetic with Strings
- A string containing numerical characters must be converted to a double or an int for use in any calculations

Converting Strings to Numbers

- A String that contains numerical characters can be converted using

```
String deuce = "2.25";
```

```
double cow = Double.parseDouble( deuce );
```

```
String four = "4";
```

```
int goat = Integer.parseInt( four );
```

- These methods will throw an exception if the string does not contain a number

Converting Numbers to Strings

- There are several methods to convert numbers to Strings
- The easy way is to concatenate a number with a string

```
double dog = 47.5;
```

```
String lizard;
```

```
lizard = "The answer is " + dog;
```

Big Computer Science Concepts

- Inheritance
- Methods
- Constructor methods
- Creating objects
- Types of variables
- Strings are different from numbers

Homework

- The weekly programming assignment is posted on Blackboard
- Program is due by midnight on **Friday**