Writing GUIs

GEEN163

“Programming is not a zero-sum game. Teaching something to a fellow programmer doesn't take it away from you. I'm happy to share what I can, because I'm in it for the love of programming.”

John Carmack

lead programmer of Doom, Quake, Rage

Homework

- The Balloon size homework is due tonight at midnight
- Upload your .java file to Blackboard
- See the tutors in Cherry Hall 124 if you have any questions or see Dr. Williams

Form Teams

- All students in GEEN163 must form a team of 3 or 4 students
- Team members should sit next to each other during all lectures
- Everyone should provide a list of their team member's full names on Blackboard by Wednesday, September 18
- You earn 10 points towards the next homework for providing the names
- Everyone must agree to be on the team

Exam

- The first exam in GEEN163 will be Monday, September 23
- The exam will cover everything since the beginning of the course
- You are allowed one 8½ by 11" page of notes
- The exam counts for 17% of your total grade

Step by Step

- Programming involves a series of step
- In general, most programs follow a similar pattern
  - Read the input data
  - Calculate the answer
  - Display the result
- GUI programs have an initialization step to format the user interface

Standard GUI Framework

```
import javax.swing.*;
public class KmMile extends JApplet implements
java.awt.event.ActionListener {
     // declare GUI object here
public void init() {
    setSize(150, 75);
    java.awt.Container pane = getContentPane();
    BoxLayout where =
          new BoxLayout(pane, BoxLayout.Y_AXIS);
    setLayout( where );
      // init stuff here
}
public void actionPerformed(
            java.awt.event.ActionEvent thing) {
     // purposeful calculation here
```

GUI parts

There are two basic parts to a GUI

- Initialize the interface
- Calculate and display the results

The size of screen objects in Java are measured in

A. inches
B. centimeters
C. pixels
D. mickeys
E. percents

Creating the Interface

The first part of the program creates the graphical interface

- GUI objects are defined
- Initial values are specified
- GUI objects are added to the content pane
- ActionListeners are added to objects

GUI objects

- What GUI objects do you need?
- Consider how you want the GUI to look
- For each object, you need to
  - Declare a variable of the proper type
  - Add the variable to the content pane in init

Example GUI Initialization

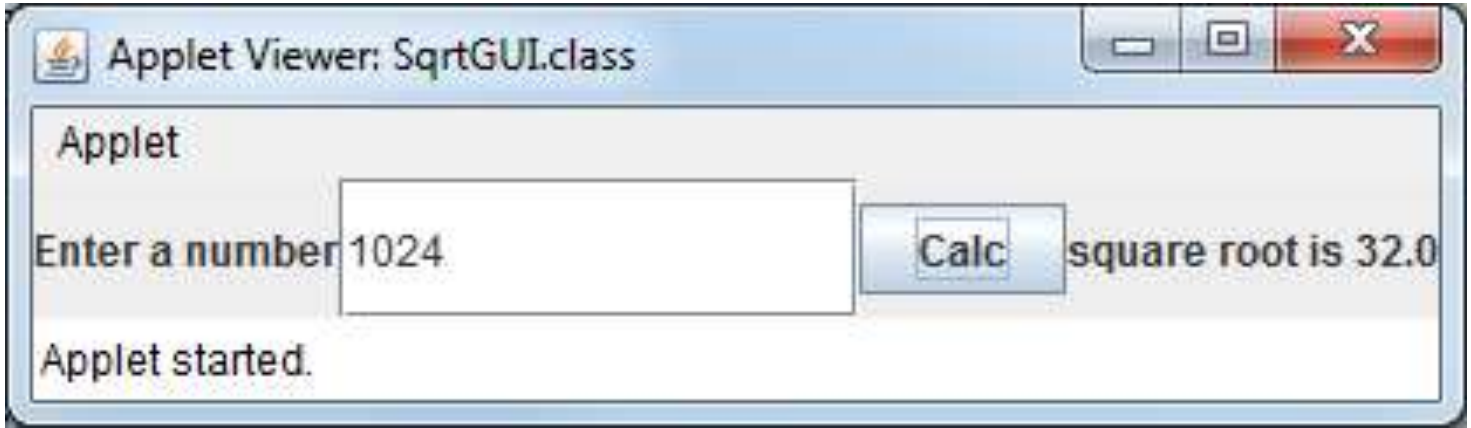
```
public class …
    JLabel   instruct =
           new JLabel("Enter kilometer");
    JTextField inKm   = new JTextField();
    JButton  goButton =
           new JButton("Convert");
    JLabel   answer   = new JLabel();

public void init() {
           …
    pane.add( instruct );
    pane.add( inKm );
    pane.add( goButton );
    pane.add( answer );
```

Unique Names

- Avoid duplicating names. Each variable should have a unique name
- There are zillions of possible names
- Do not name a JTextField and a double the same name

Work Together in Teams

- Consider a program that displays the square root of a number
- What GUI objects does this program need?\
- Write the variable declarations in the order they should be added to the content pane



Possible Solution

```
JLabel enterMsg =
                    new JLabel("Enter a number");
JTextField inStuff = new JTextField();
JButton    button  = new JButton("Calc");
JLabel     result  = new JLabel();
```

Action Part

- The actionPerformed method is called whenever the user does something
- The real purpose for the program is run in the actionPerformed method
- The actionPerformed method can be called many times if the user presses a button many times

actionPerformed

There are three general parts to actionPerformed

- Get the input
- Calculate the answer
- Display the results

Example actionPerformed

```
public void actionPerformed( … ) {
    // get the input data
   String textKm = inKm.getText();
   double km = Double.parseDouble( textKm );
    // calculate the answer
   double miles = km * MILEPERKM;
    // display the answer
   answer.setText( miles + " miles");
}
```

GUI input

- GUI programs get their input from JTextFields
- Use the **`getText()`** method of a JTextField

- If you want to erase the input in the JTextField, use the setText method to set the text to null after you get the input

```
myInputField.setText( null );
```

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 inData = keyboard.nextDouble();
double cow = Double.parseDouble( inData );
String four = “4”;
int goat = Integer.parseInt( four );
```

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

Inputting Numbers

- The getText() method returns a String
- If a number is entered in a JTextField, it will have to be converted to an int or double

```
JTextField gekko = new JTextField();
                    // in the actionPerformed method
String dragon = gekko.getText();
double lizard = Double.parseDouble( dragon );
```

Changing a Component's Text

- A useful method is setText It can be used to change the text of any object that has text, such as JButtons, JLabels and JTextfields

```
javax.swing.JLabel  msg   =
        new javax.swing.JLabel("Do something");

msg.setText("Don't do something");
```

GUI Output

- When a program with a GUI wants to display a result, it usually puts the value in a JLabel object.

```
JLabel aardvark = new JLabel();
                  …
double answer = some equation;
aardvark.setText("The answer is "+answer);
```

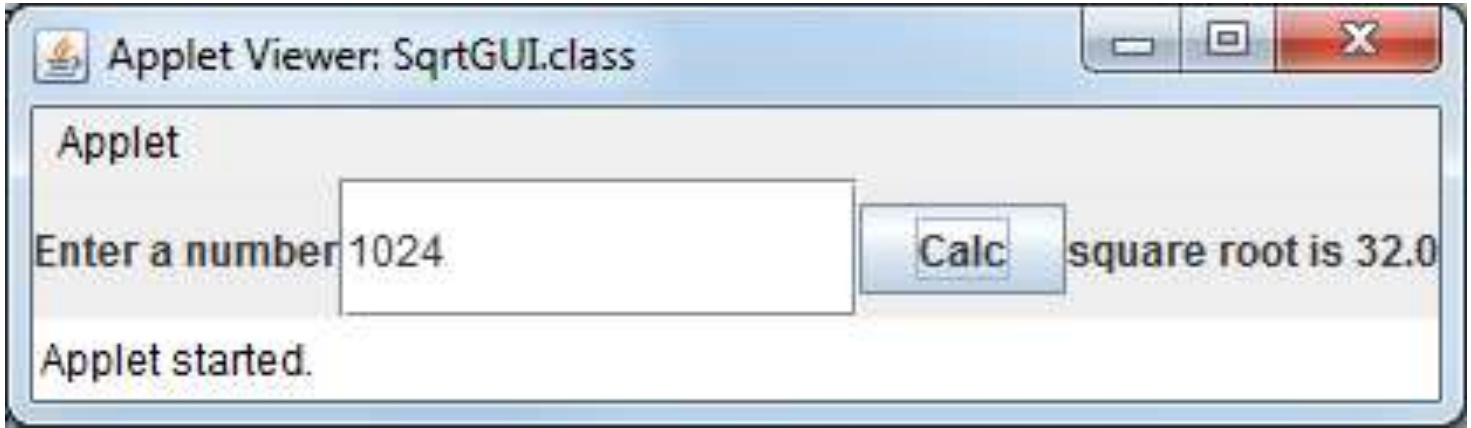
Change of Thinking

- Applets do not have a main method
  - The init method is used to initialize the GUI
- Input is usually from JTextfield objects
  - **`java.util.Scanner`** is not used
- Output is usually to JLabel objects
  - **`System.out.println`** is not used

Work Together

- Write the action performed method for the square root program
- The GUI objects are:

```
JLabel enterMsg =
                    new JLabel("Enter a number");
JTextField inStuff = new JTextField();
JButton    button  = new JButton("Calc");
JLabel     result  = new JLabel();
```



Possible Solution

```
String numText = inStuff.getText();
double num = Double.parseDouble( numText );
double root = Math.sqrt( num );
result.setText("square root is " + root);
```

paint Method

- The web browser calls the paint method when it wants to display graphics

```
public void paint( 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 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.YELLOW

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

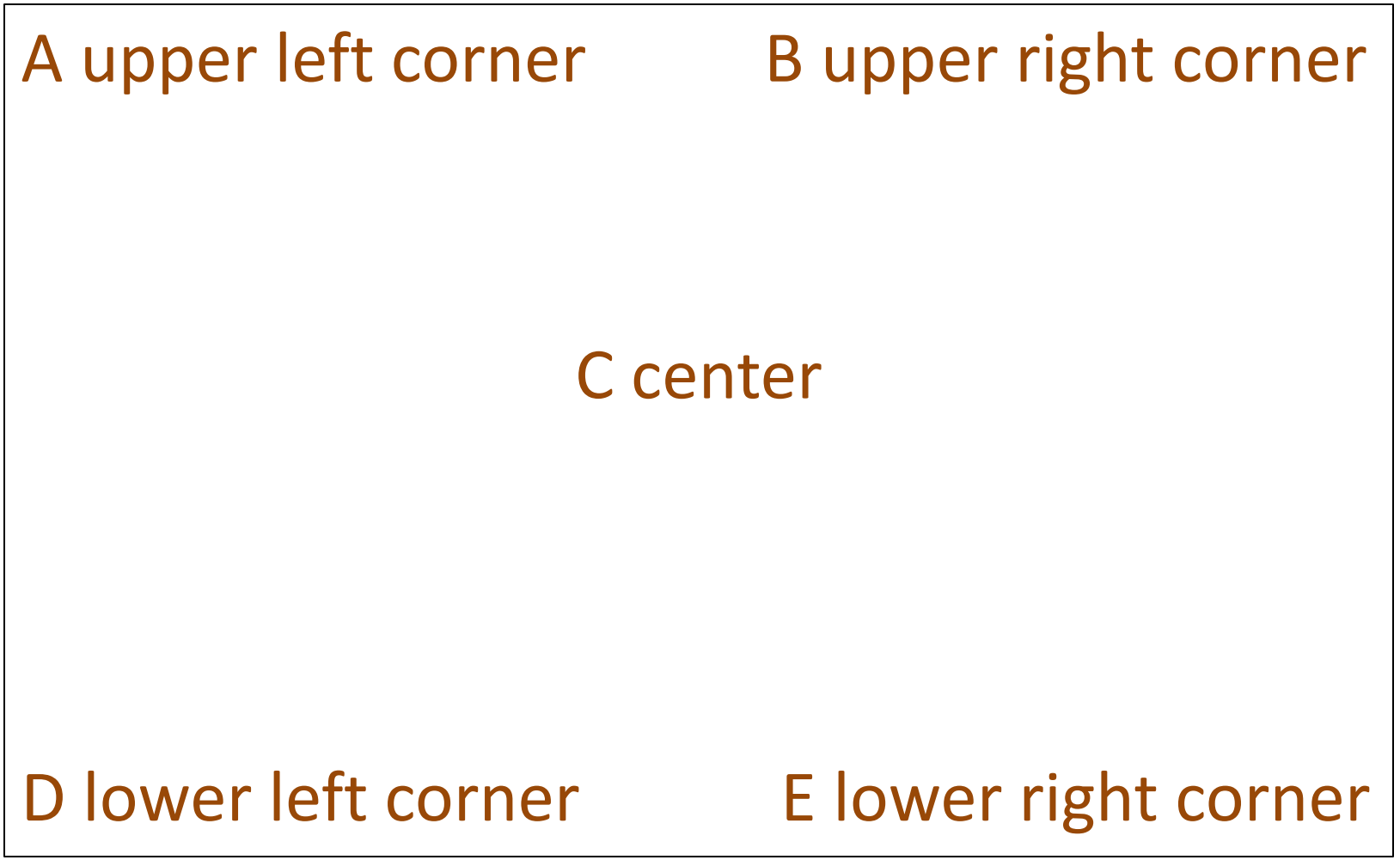
setColor

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

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

- where **`gobj`** is an object of the type java.awt.Graphics

Location 0,0 is

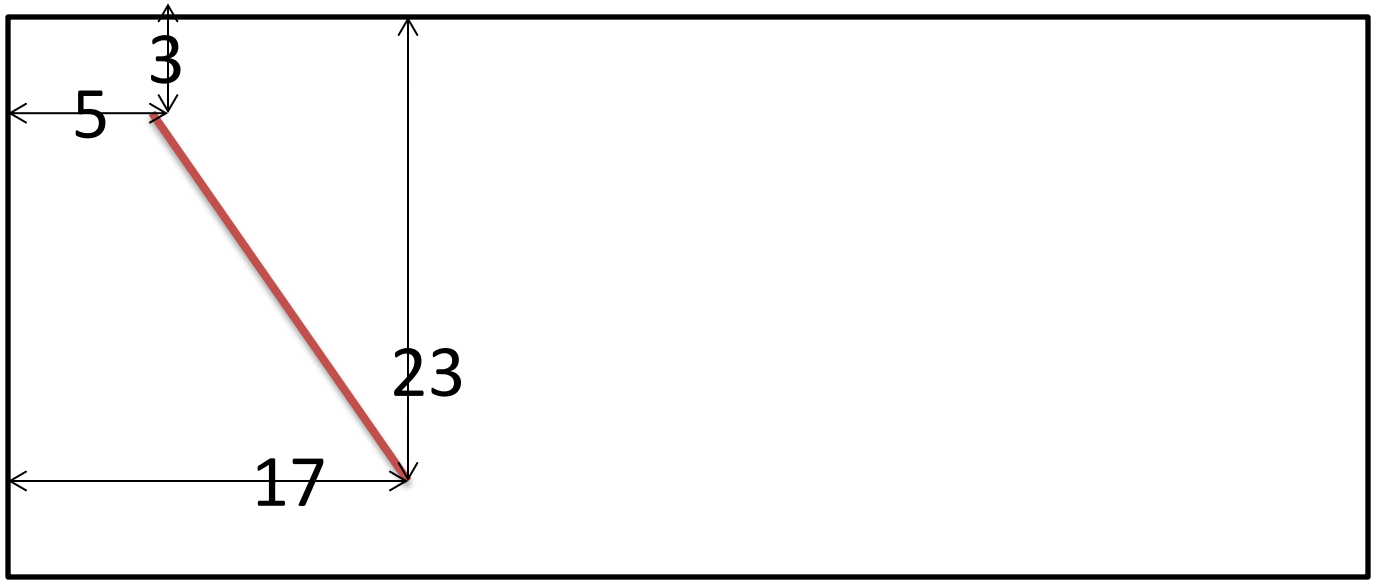


drawLine

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

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

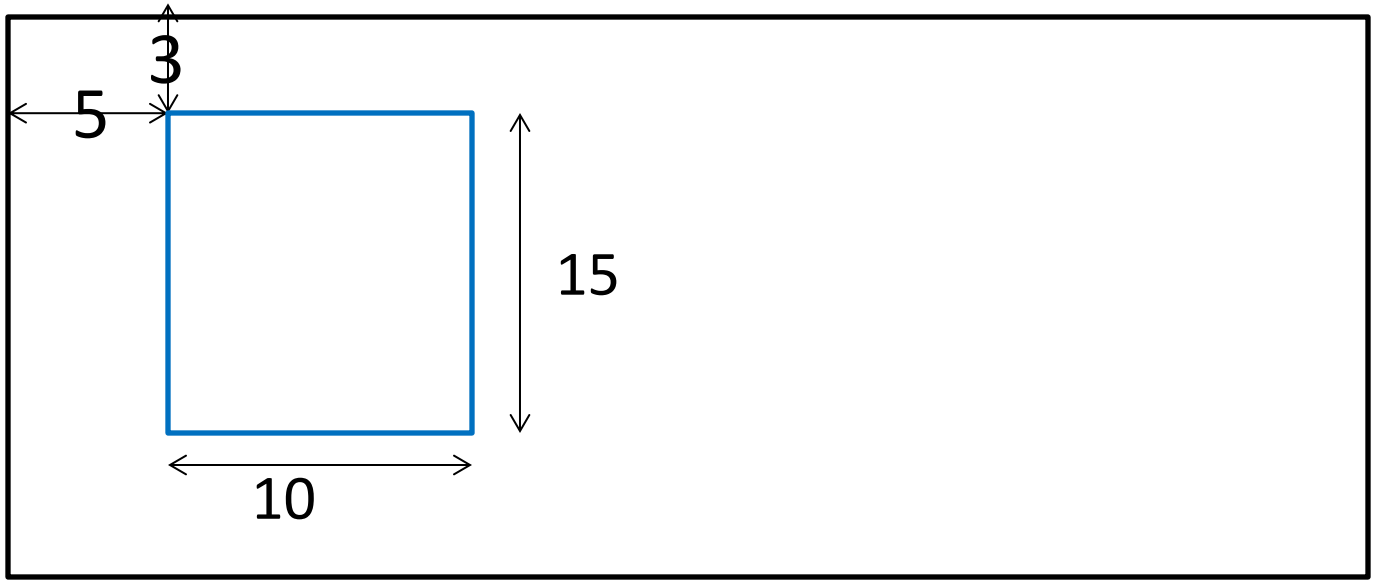
- The line is drawn using the current color



drawRect

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

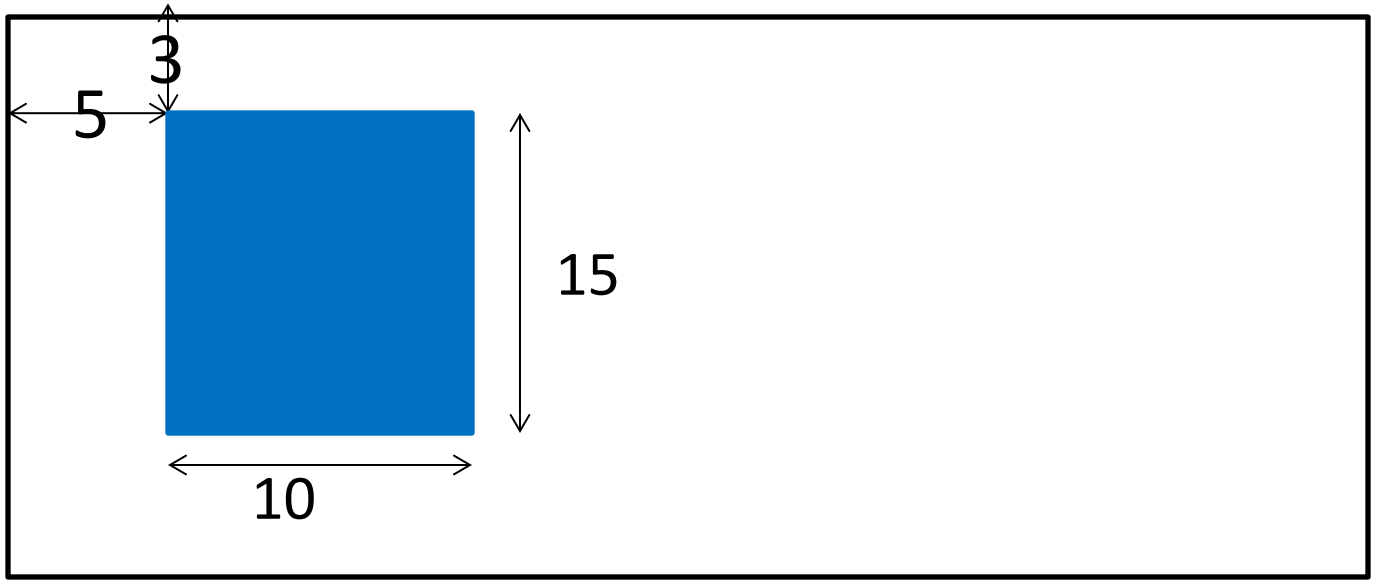
```
gobj.drawRect( x, y, width, height);
gobj.drawRect( 5, 3, 10, 15 );
```



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

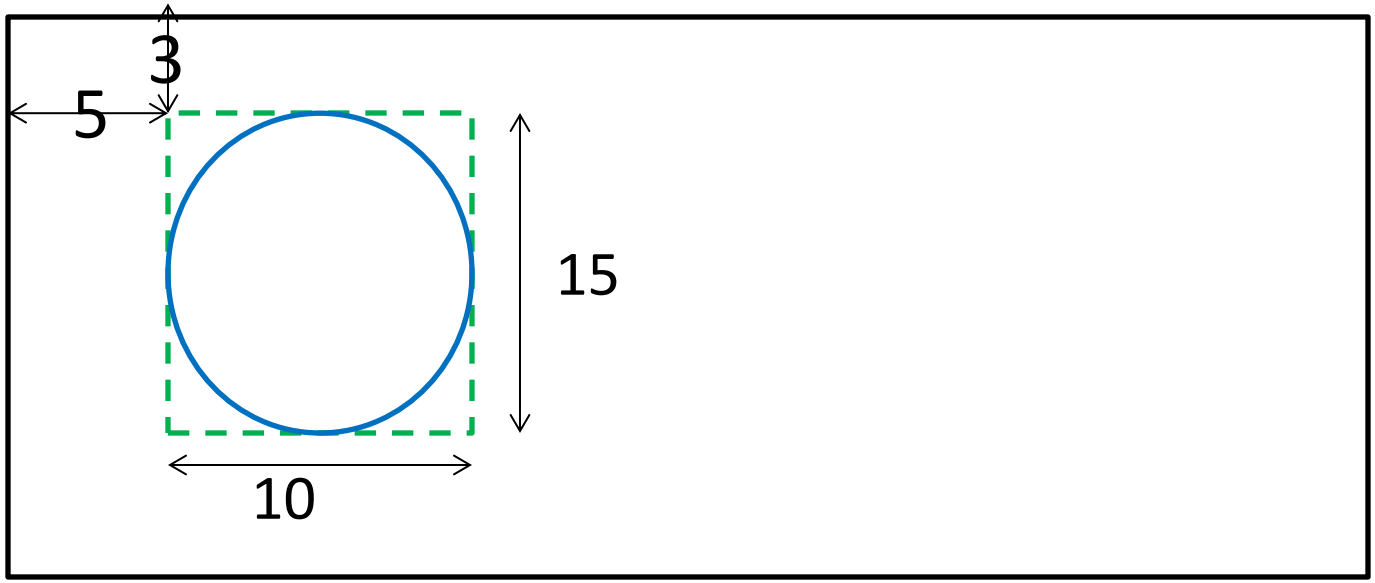
```
gobj.fillRect( 5, 3, 10, 15 );
```



drawOval

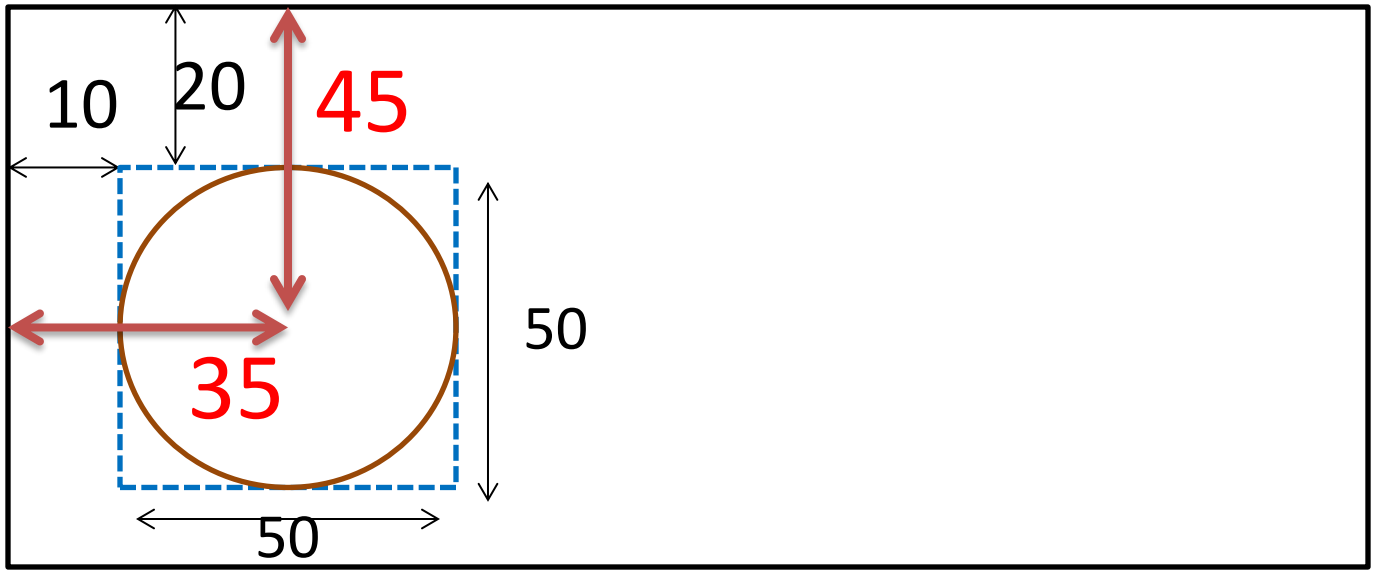
- The drawOval method of java.awt.Graphics objects draws a circle or oval to fit in a box of width and height whose upper left is x,y

```
gobj.drawOval( x, y, width, height );
gobj.drawOval( 5, 3, 10, 15 );
```



Most Difficult Question

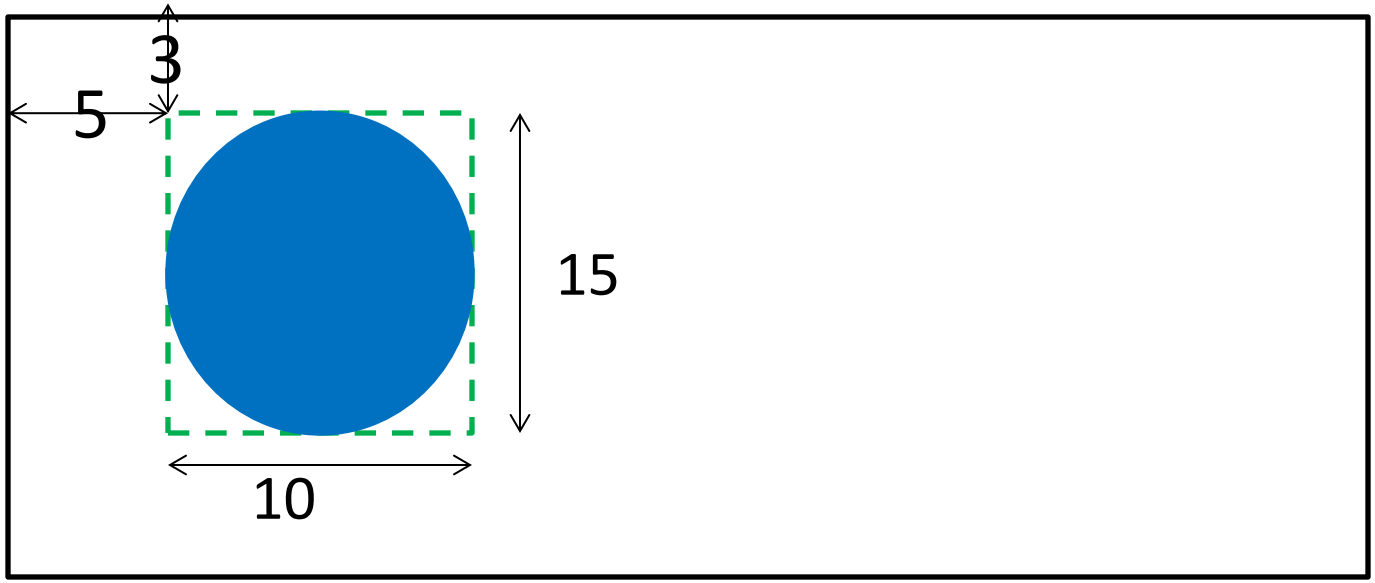
- What are the coordinates of the center of a circle created by drawOval(10, 20, 50, 50)?



fillOval

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

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



drawArc

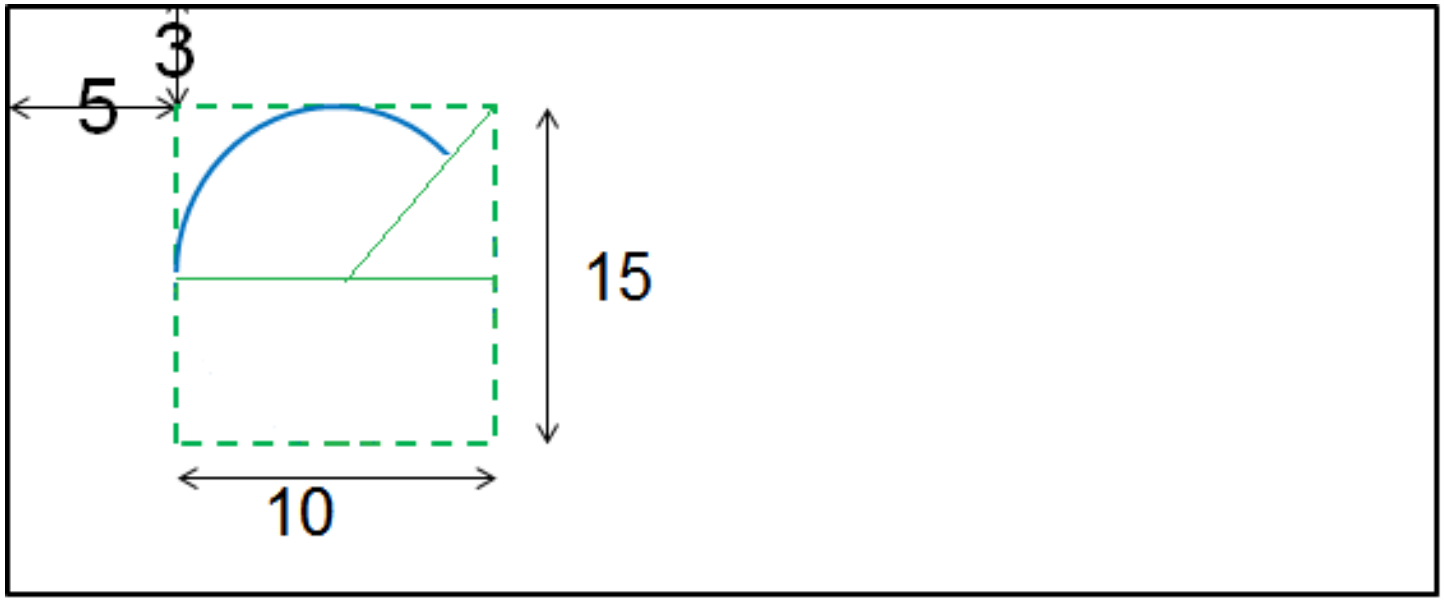
- The drawArc method of java.awt.Graphics objects draws part of an oval from start angle for arc angle degrees
- Angles are in degrees not radians
- Zero degrees in the 3 o'clock position

```
gobj.drawArc( x, y, width, height,
              startAngle, arcAngle );
```

drawArc

- The drawArc method of java.awt.Graphics objects draws part of an oval from

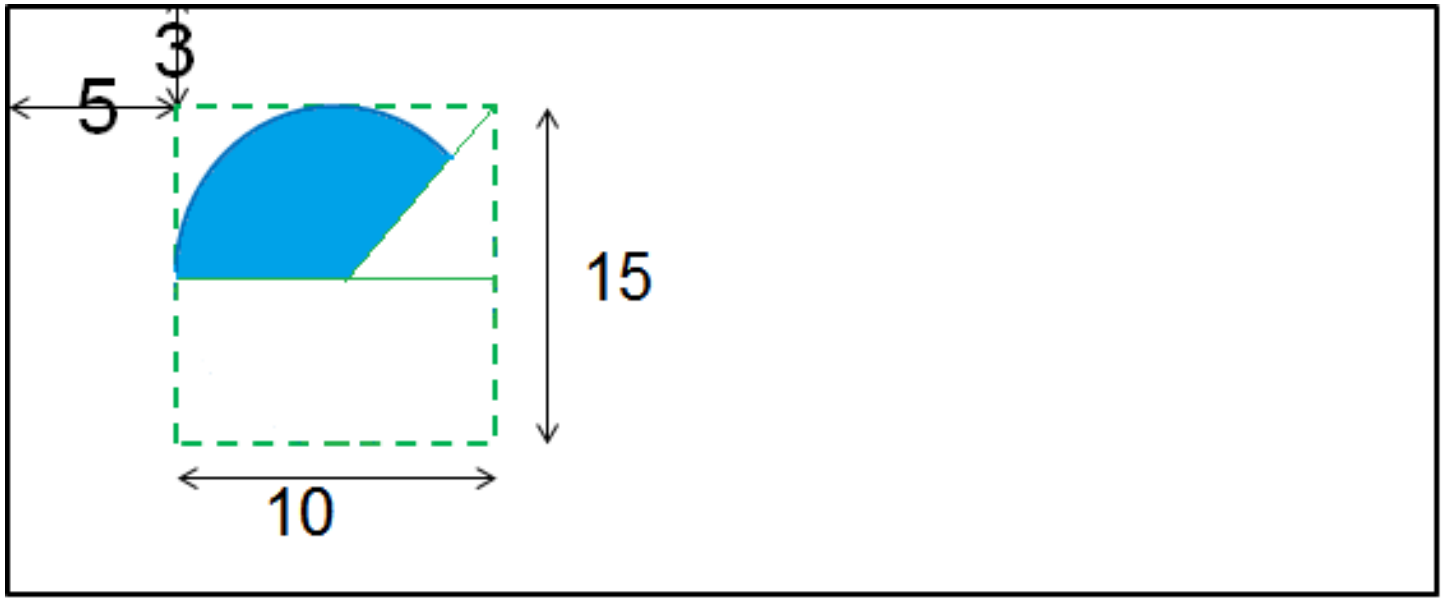
```
gobj.drawArc( 5, 3, 10, 15, 45, 135);
```



fillArc

- The fillArc method is just like the drawArc method except it fills the arc with the current color

```
gobj.fillArc( 5, 3, 10, 15, 45, 135);
```



drawString

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

```
gobj.drawString( String, x, y);
gobj.drawString("Whatever", 5, 3 );
```

drawString is a

A. static method
B. dynamic method
C. both
D. neither
E. cannot be determined

Drawing Pictures

- You can draw simple pictures using the methods of the Graphics
- If shapes overlap, the shape from the later method call will be on top

Example Picture Applet

```
public class Artist extends javax.swing.JApplet {
    int count = 0;
    public void init() {
        setSize( 250, 250 );
    }
    public void paint( java.awt.Graphics elephant) {
        elephant.setColor(java.awt.Color.BLUE);
        elephant.fillRect( 10, 20, 100, 150 );
        count = count + 1;
        elephant.setColor(java.awt.Color.RED);
        elephant.drawString( "paint called " + count
                                    + "times", 5, 185);
    }
}
```

Form Teams

- All students in GEEN163 must form a team of 3 or 4 students
- Team members should sit next to each other during all lectures
- Everyone should provide a list of their team member's full names on Blackboard by Wednesday, September 18
- You earn 10 points towards the next homework for providing the names
- Everyone must agree to be on the team

Homework

- The Balloon size homework is due tonight at midnight
- Upload your .java file to Blackboard
- See the tutors in Cherry Hall 124 if you have any questions or see Dr. Williams

Exam

- The first exam in GEEN163 will be Monday, September 23
- The exam will cover everything since the beginning of the course
- You are allowed one 8½ by 11" page of notes
- The exam counts for 17% of your total grade