More about GUIs

GEEN163

"The best programmers are not marginally better than merely good ones. They are an order-of-magnitude better, measured by whatever standard: conceptual creativity, speed, ingenuity of design, or problem-solving ability."

Randall E. Stross

Short Week

There will be no class on:

- Wednesday, April 12 – Honor's Convocation
- Friday, April 14 – Good Friday

Quizzes Next Week

- There will be a quiz in recitation next week, either Monday, April 17 or Friday, April 21

- There will be a lab quiz next week, Thursday, April 20

Interesting Objects

Many GUI program can be created with a few simple components

- **JLabel** – Displays text in the GUI
- **JButton** – Creates a button you can press
- **JTextField** – Creates an input text box

- All of these objects are from the package javax.swing

How do you make an array of 5 buttons for a GUI?

A. JButton[5] gazelle = new JButton[];

B. JButton gazelle = new JButton(5);

C. JButton[] gazelle = new JButton[5];

D. JButton[] gazelle = { JButton(), JButton(), JButton(), JButton(), JButton() };

E. You cannot make an array of GUI objects

Array of Objects

- If you create an array of objects, you need to create the objects in the array

```
for (int cow = 0; cow < 5; cow++) {
        gazelle[cow] = new JButton("Go");
        pane.add( gazelle[cow] );
}
```

Implementing ActionListener

- Implementing the ActionListener interface makes the program respond to GUI input
- Programs implementing ActionListener must have an actionPerformed method

```
public class MyApplet
        extends javax.swing.JFrame
        implements java.awt.event.ActionListener {
```

Interfaces

- A Java interface has a list of method headers that a program must implement
- ActionListener is an interface that requires a program to include an actionPerformed method
- Programmers can create their own interfaces
  - This is a topic for GEEN165

Other Interfaces

- There are many interfaces in the standard Java libraries
- The Comparable interface requires implementing classes to have a compareTo method

```
public int compareTo( MyClass other )
```

- Returns a negative, zero, or positive integer as this object is less than, equal to, or greater than the other object

The Comparable interface is implemented by which class?

A. Math
B. Scanner
C. String
D. Only classes that you write

Generics

- Sometime when you create an object or interface in Java, you must specify the class it will work with
- When you used a HashTable you had to specify the type of the key and data

```
Hashtable<String, WordData> dict =
                    new Hashtable<>(18000);
```

- You must specify the type for Comparable

```
public class Mine implements Comparable<Mine>
```

Adding a Listener

- The method addActionListener should be called on each GUI component that can do something
- Do not call addActionListener on components that do not do anything, like JLabels

```
javax.swing.JButton  frog = new
                        javax.swing.JButton("Go");
frog.addActionListener( this );
```

Taking Action

- When a user clicks an object with an ActionListener, Java calls the method

```
public void actionPerformed(
                java.awt.event.ActionEvent   rabbit) {
    // do something here
}
```

- This method should do what the user expects to happen when the button is pressed

Getting the Text

- You can get the text from a GUI component with the getText() method
- This is useful to get the text a person entered in the JTextField

```
javax.swing.JTextField  llama = new
                                javax.swing.JTextField();
// in the actionPerformed method
String answer = llama.getText();
```

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");
```

Who Dunit?

- Many GUIs have more than one active object
- You can determine where the action was taken by using the **getSource()** method of the event
- The result of **getSource()** should be compared to each GUI object

```
public void actionPerformed(
                    java.awt.event.ActionEvent   frog ) {
    if ( frog.getSource() == GUIobject ) {
```

Example Use of getSource()

```
JButton delete = new JButton("kill");
JButton save   = new JButton("allow");
        - - - - - - - - - - - - - - - - - - - - - - - - -
public void actionPerformed(
                java.awt.event.ActionEvent   toad) {
    if (toad.getSource() == delete ) {
        // delete something
    } else {
        // don't delete anything
    }
```

Looping in GUI Programs

- When you want a user to enter multiple values in a GUI program, you probably do **not** need a for loop or a while loop
- Each time the user enters a name, it will call actionPerformed

What will select the pressed button?

```
JButton deer = new JButton();
public void actionPerformed(
                    java.awt.event.ActionEvent  bear)
```

A. `if( bear == deer )`
B. `if( bear.getSource() == JButton )`
C. `if( bear.getSource() == deer )`
D. `if( deer == JButton.getSource() )`
E. `if( this == bear.getSource() )`

setVisible

- Most GUI components have a setVisible method

  setVisible(boolean)

- If the boolean parameter is true, the object shows, if false it does not show
- Calling **`setVisible(true)`** makes the frame appear onscreen

Disabling GUI Components

- Individual GUI components (i.e. JButtons) can be made invisible or disabled
- A disabled component is gray and cannot be selected

```
JButton myButton = new JButton("OK");
myButton.setEnabled(false);
```

- This will make the button gray and unusable

setTitle

- The setTitle method of a JFrame sets the text in the upper border of the window

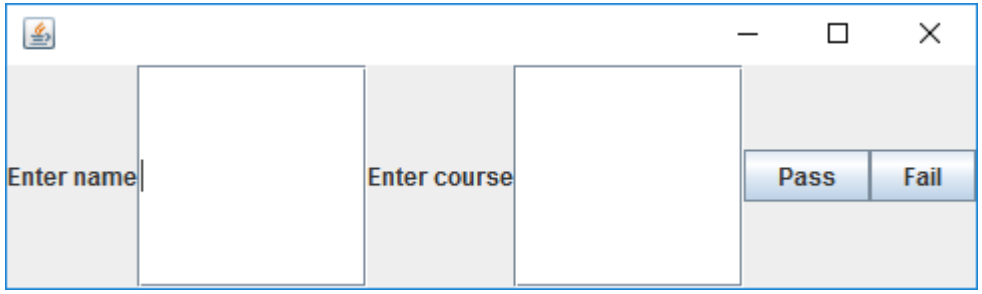
```
setTitle("My GUI program");
```

pack

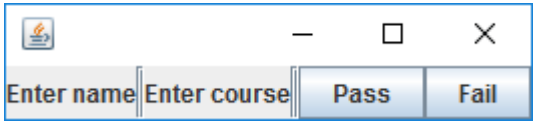
- The pack method sizes the frame so that all its contents are at their preferred sizes
- When you use the pack method, you do not have to call setSize for the frame
- If you use absolute positioning of components instead of a layout manager, you do not need to call pack

pack() Issues

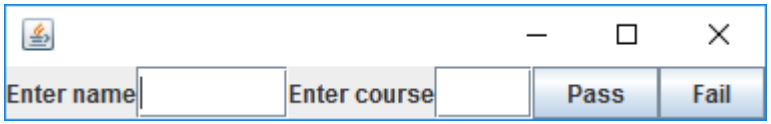
- Using setSize and not pack()



- Using pack()



- Using pack() with spaces in JTextfields



Sample GUI Constructor

```
JTextField      inStuff     = new JTextField("          ");
JButton         theButton   = new JButton("OK");
public MyGUI( ) {
    java.awt.Container pane = getContentPane();
    BoxLayout where = new BoxLayout(pane, BoxLayout.Y_AXIS);
    setLayout(where);
    pane.add(inStuff);
    pane.add(theButton);
    theButton.addActionListener(this);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    pack();
}
```

Method Review

- Follow the flow of execution. Execution starts in the main method
- Constructors can be used to initialize objects

How can you call the `chew()` method on all objects??

Assume you have an array cow that contains 12 objects of the class Bovin which has a method called chew()

```
for (int i = 0; i < 12; i++) {
  A.      cow.chew(i);
  B.      cow.chew[i];
  C.      cow[i].chew();
  D.      cow.chew()[i]
  E.      Bovine.chew(i)
```

Creating Arrays of Objects

- Often you will create an array and put the objects in the array as you create them
- Consider a program that will create an array of WordData objects using data from a file

```
WordData[] centaur = new WordData[1000];
```

- The array can hold 1000 objects, but it is initially empty with all elements equal to null

Example Class

```
public class WordData {
        private String sphinx;
        private int  count = 1;
        public WordData( String aWord ) {
                sphinx = aWord;
        }
        public String getWord() {
                return sphinx;
        }
        public boolean match( String mermaid ) {
                return sphinx.equals( mermaid );
        }
```

Putting Data in the Array

```
WordData[] centaur = new WordData[1000];
int  numObj = 0;
java.io.File griffin = new java.io.File("stuff.txt");
Scanner dragon = new Scanner( griffin );
while ( dragon.hasNext() ) {
    String unicorn = dragon.next();
    centaur[numObj] = new WordData(unicorn);
    numObj++;
}
```

Incrementing in an Index

```
WordData[] centaur = new WordData[1000];
int  numObj = 0;
java.io.File griffin = new java.io.File("stuff.txt");
Scanner dragon = new Scanner( griffin );
while ( dragon.hasNext() ) {
    String unicorn = dragon.next();
    centaur[numObj++] = new WordData(unicorn);
}
```

Finding Things in an Array

- Programs may need to search an array of objects for one that matches an instance variable
- If the data in the object is private, you will have to use a method to access it

Search Objects with Get

```
String target = "special";
for (int roc = 0; roc < numObj; roc++) {
        if (target.equals(centaur[roc].getWord()) {
                System.out.println("found it");
                break;
        }
}
```

Search Objects with match()

```
String target = "special";
for (int roc = 0; roc < numObj; roc++) {
        if (centaur[roc].match( target )) {
                System.out.println("found it");
                break;
        }
}
```

Graphics Coordinates

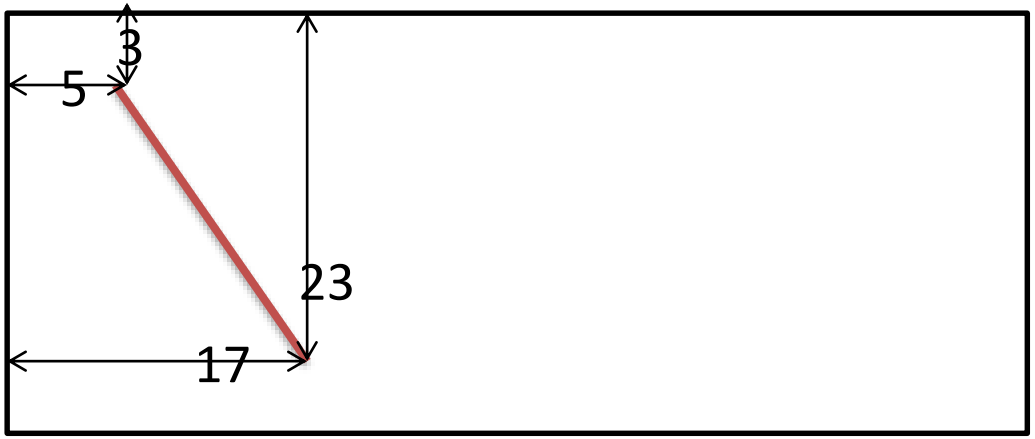
- The screen has a coordinate system with the origin in the upper left corner
- Coordinates are given in pixels (Picture Elements)
- An X coordinate specifies the distance from the left edge
- A Y coordinate specified the distance from the top edge
- The screen size depends on the device

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

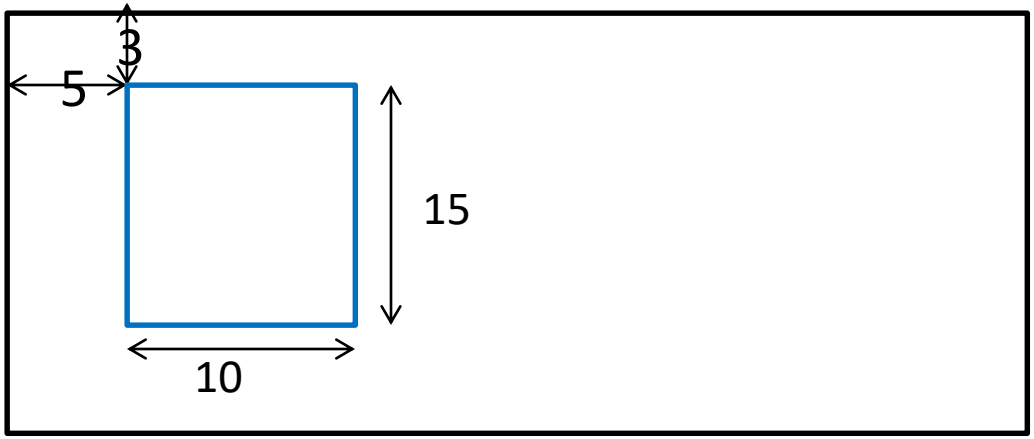
D.
center

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

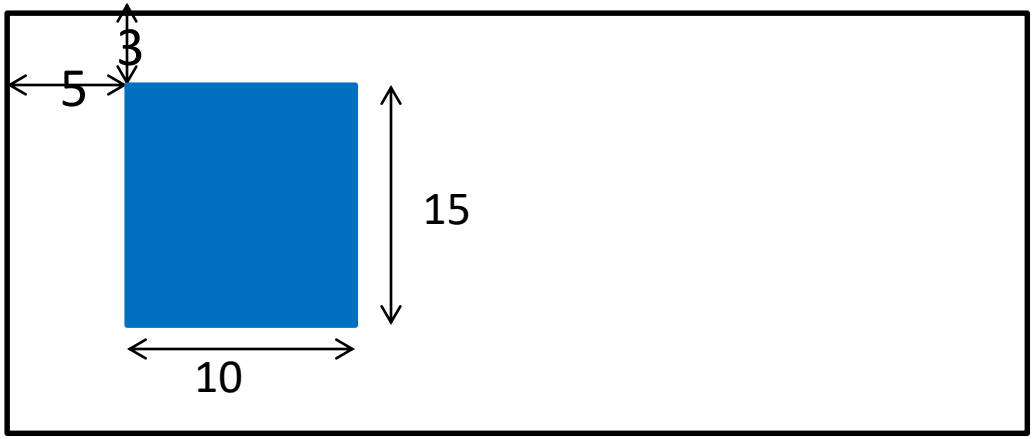
```
bird.drawRect( x, y, width, height);
bird.drawRect( 5, 3, 10, 15 );
```



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

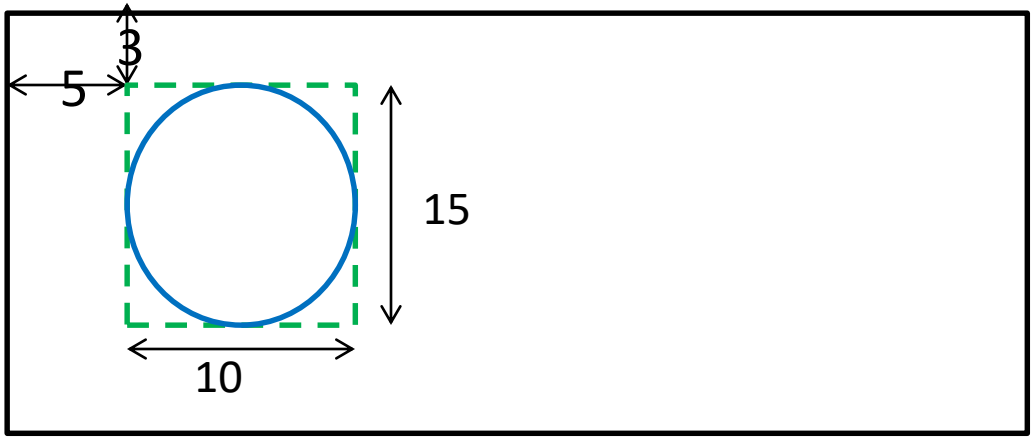
- It is permissible to have a graphics object (such as a rectangle) extend outside the visible window
- The portion outside the window will not be visible
- An object can overlap one another

```
bird.fillRect( 5, 3, 10, 15 );
bird.setColor( Color.GREEN );
bird.fillRect( 8, 15, 10, 15 );
```

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

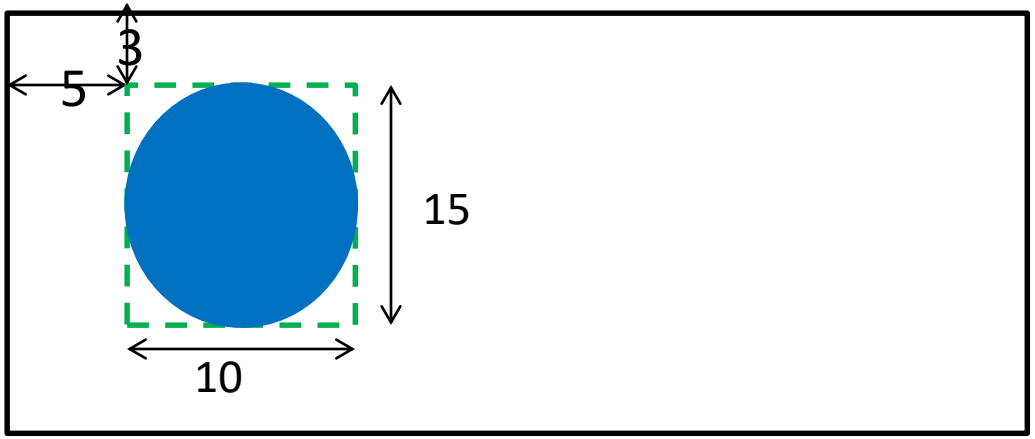
```
bird.drawOval( x, y, width, height );
bird.drawOval( 5, 3, 10, 15 );
```



fillOval

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

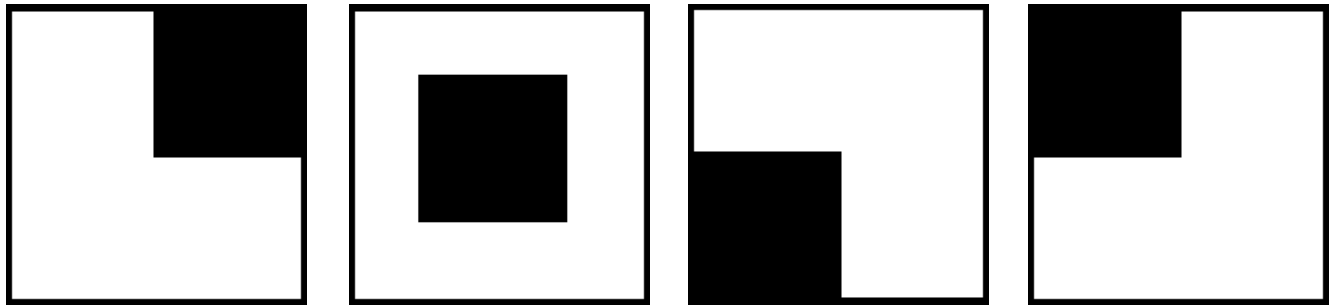
```
bird.fillOval( 5, 3, 10, 15 );
```



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.



drawPolygon

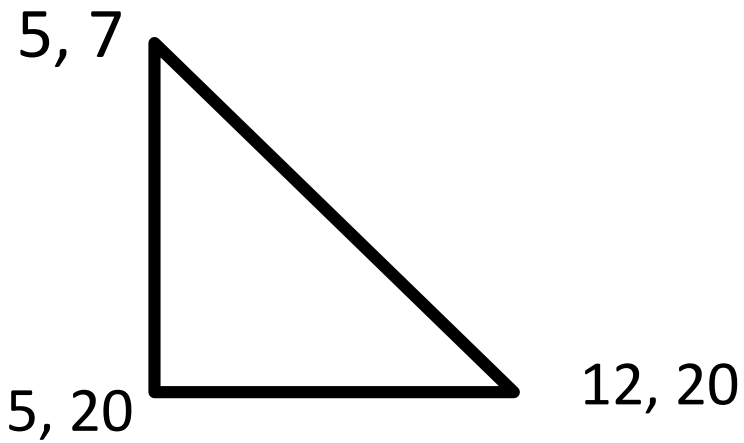
- The drawPolygon draws lines between the points specified by an array

```
drawPolygon(int[] xPoints,
    int[] yPoints, int nPoints)
```

- where there are nPoints values in both arrays
- Lines are drawn from (xPoints[0], yPoints[0]) to (xPoints[1], yPoints[1]) and from there to (xPoints[2], yPoints[2]) and so forth
- A line is drawn between the last point and the first point

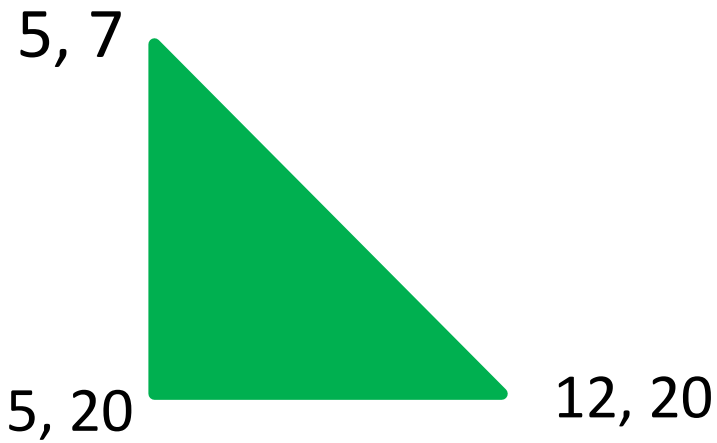
drawPolygon Example

```
int[] xs = { 5, 12,  5};
int[] ys = { 7, 20, 20};
bird.drawPolygon( xs, ys, 3 );
```



fillPolygon Example

```
int[] xs = { 5, 12,  5};
int[] ys = { 7, 20, 20};
bird.fillPolygon( xs, ys, 3 );
```



Short Week

There will be no class on:

- Wednesday, April 12 – Honor's Convocation
- Friday, April 14 – Good Friday

Quizzes Next Week

- There will be a quiz in recitation next week, either Monday, April 17 or Friday, April 21

- There will be a lab quiz next week, Thursday, April 20