

# Garfield AP CS

Strings, Graphics

# Strings

- **string**: A sequence of text characters.

```
String name = "text";  
String name = expression;
```

- Examples:

```
String name = "Marla Singer";  
  
int x = 3;  
int y = 5;  
String point = "(" + x + ", " + y + ")";
```

# Strings as parameters

- In RedundantStars, SpaceNeedle, how could Strings as parameters make code more readable?

```
public static void repeat(String str, int count) {  
    for(int i = 1; i <= count; i++) {  
        System.out.print(str);  
    }  
}
```

# Method identifiers

- Method header

- `accessModifier static returnType methodName (parameter list)`

- Method signature

- method name and parameter count and types
- Must be unique
- Can have two methods with the same name as long as parameter list is different: method **overloading**

# Class constants vs. parameters?

- Does the change affect the whole program?
  - That's a class constant
- Is there a piece of code you want to reuse?
  - That's a method
- Can use both in one program

# Objects (hit and run!)

- Entities that contains data and behavior
  - data: variables inside the object
  - behavior: methods inside the object
- Constructing (creating) an object:
  - `Type objectName = new Type(parameters) ;`
- Calling an object's method
  - `objectName.methodName(parameters) ;`

# Graphics

- We will draw graphics in Java using 3 kinds of objects:
  - `DrawingPanel`: A window on the screen. Not part of Java; provided by UW
  - `Graphics`: A "pen" to draw shapes/lines on a window.
  - `Color`: Colors in which to draw shapes.

# DrawingPanel



*"Canvas" objects that represents windows/drawing surfaces*

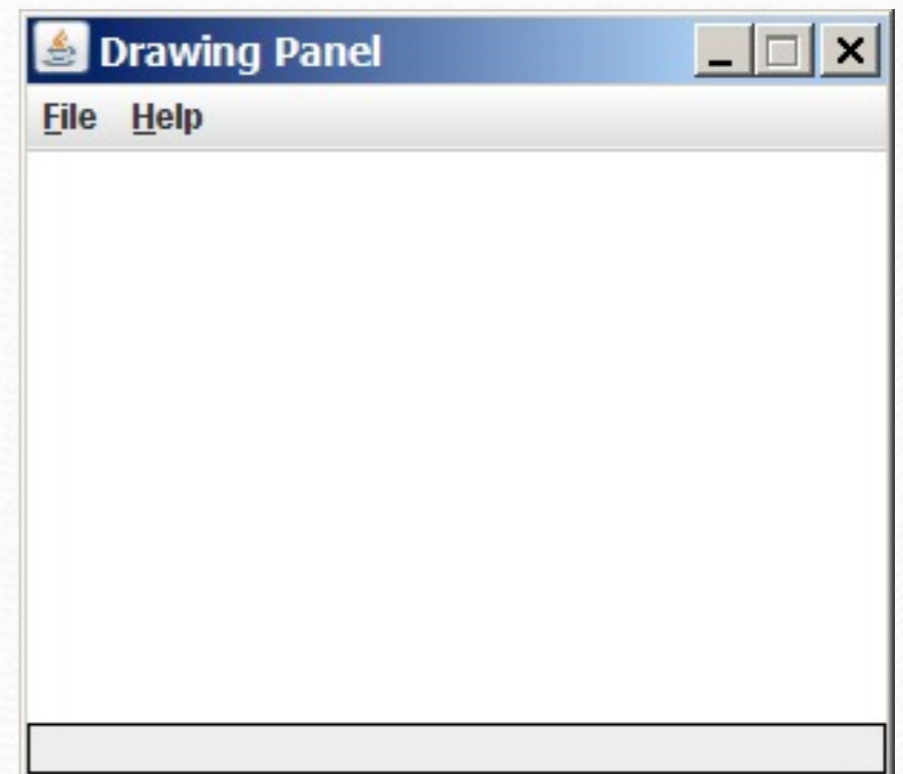
- To create a window:

```
DrawingPanel name = new DrawingPanel (width, height);
```

**Example:**

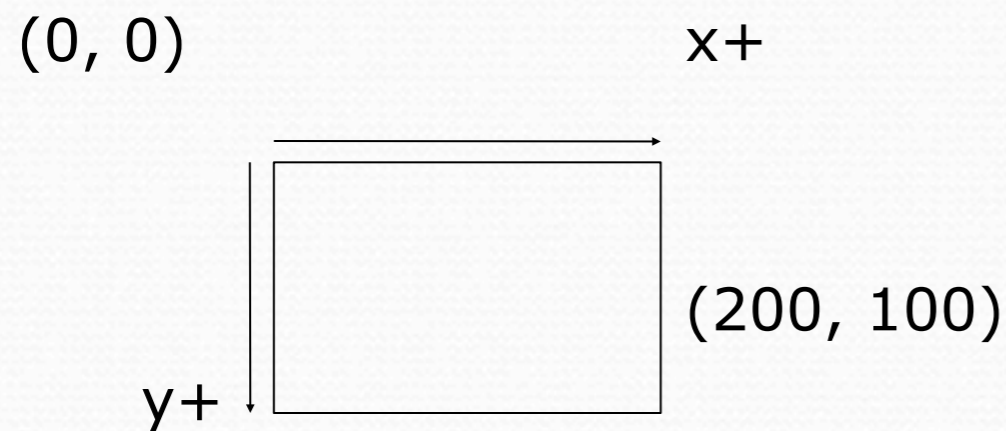
```
DrawingPanel panel = new DrawingPanel(300, 200);
```

- The window has nothing on it.
  - We can draw shapes and lines on it using another object of type `Graphics`.



# Coordinate system

- Each  $(x, y)$  position is a *pixel* ("picture element").
- $(0, 0)$  is at the window's top-left corner.
  - $x$  increases rightward and the  $y$  increases downward.
- The rectangle from  $(0, 0)$  to  $(200, 100)$  looks like this:



# Graphics



*"Pen" objects that can draw lines and shapes*

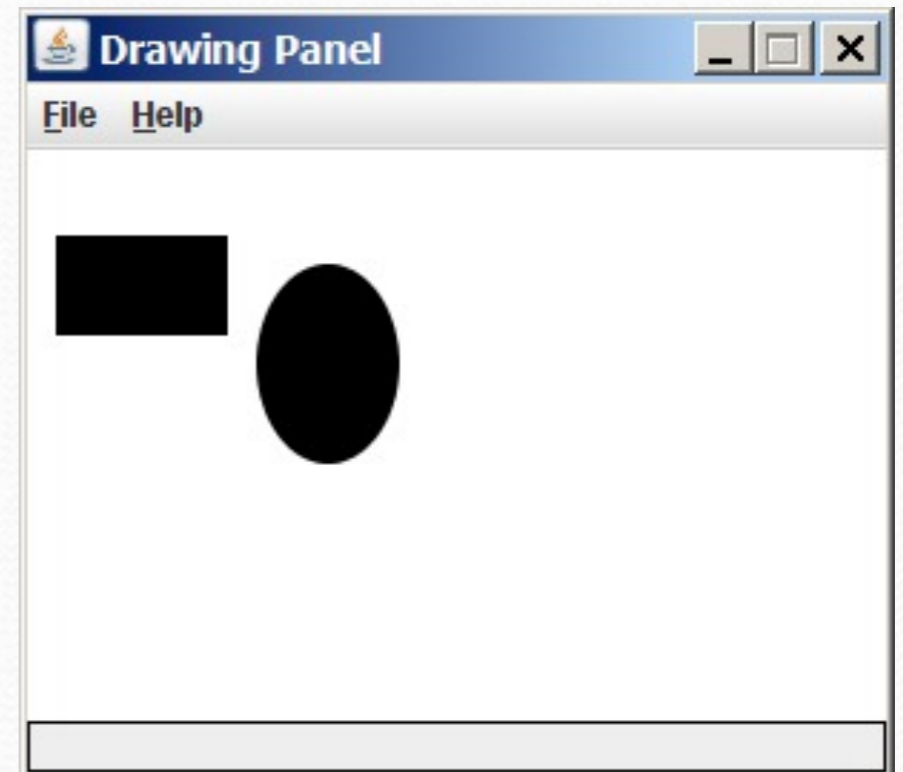
- Access it by calling `getGraphics` on your `DrawingPanel`.

```
Graphics g = panel.getGraphics();
```

- Draw shapes by calling methods on the `Graphics` object.

```
g.fillRect(10, 30, 60, 35);
```

```
g.fillOval(80, 40, 50, 70);
```



# Java class libraries, import

- **Java class libraries:** Classes included with Java's JDK.
  - organized into groups named *packages*
  - To use a package, put an *import declaration* in your program.
- **Syntax:**

```
// put this at the very top of your program
import packageName.*;
```
- `Graphics` is in a package named `java.awt`

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

  - In order to use `Graphics`, you must place the above line at the very top of your program, before the `public class` header.

# Graphics methods

Method name	Description
<code>g.drawLine(<b>x1</b>, <b>y1</b>, <b>x2</b>, <b>y2</b>);</code>	line between points $(x1, y1)$ , $(x2, y2)$
<code>g.drawOval(<b>x</b>, <b>y</b>, <b>width</b>, <b>height</b>);</code>	outline largest oval that fits in a box of size $width * height$ with top-left at $(x, y)$
<code>g.drawRect(<b>x</b>, <b>y</b>, <b>width</b>, <b>height</b>);</code>	outline of rectangle of size $width * height$ with top-left at $(x, y)$
<code>g.drawString(<b>text</b>, <b>x</b>, <b>y</b>);</code>	text with bottom-left at $(x, y)$
<code>g.fillOval(<b>x</b>, <b>y</b>, <b>width</b>, <b>height</b>);</code>	fill largest oval that fits in a box of size $width * height$ with top-left at $(x, y)$
<code>g.fillRect(<b>x</b>, <b>y</b>, <b>width</b>, <b>height</b>);</code>	fill rectangle of size $width * height$ with top-left at $(x, y)$
<code>g.setColor(<b>Color</b>);</code>	set Graphics to paint any following shapes in the given color

# Your tasks

- Rewrite RedundantStars using parameterized methods
- Finish writing the parameterized methods from last week's handout
- Play with Graphics