Variables, Types, and Expressions

Announcements

- Karel the Robot due right now.
  - Email: Due Sunday, January 22 at 11:59PM.
- Update to assignment due dates:
  - Assignments 2 – 5 going out one day later.
  - Contact me if this is a problem.
  - Updated syllabus will be posted to the course website.
- Blank Java project available.
  - Play around with Java on your own!

Variables

- A **variable** is a location where a program can store information for later use.

Variables

- A **variable** is a location where a program can store information for later use.

Variables

- A **variable** is a location where a program can store information for later use.

- Each variable has three pieces of information associated with it:

Variables

- A **variable** is a location where a program can store information for later use.

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?

Variables

- A **variable** is a location where a program can store information for later use.

`numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?

Variables

- A **variable** is a location where a program can store information for later use.

`numVoters`

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?

Variables

- A **variable** is a location where a program can store information for later use.

```
int numVoters
```

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?

Variables

- A **variable** is a location where a program can store information for later use.

```
int numVoters
```

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?
  - **Value**: What value does the variable have at any particular moment in time?

Variables

- A **variable** is a location where a program can store information for later use.

- Each variable has three pieces of information associated with it:
  - **Name**: What is the variable called?
  - **Type**: What sorts of things can you store in the variable?
  - **Value**: What value does the variable have at any particular moment in time?

Variables

A **variable** is a location where a program can store information for later use.

Each variable has three pieces of information associated with it:

- **Name**: What is the variable called?
- **Type**: What sorts of things can you store in the variable?
- **Value**: What value does the variable have at any particular moment in time?

Variable Names

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)
  - consist of letters, numbers, and underscores,

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)
  - consist of letters, numbers, and underscores,

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)
  - consist of letters, numbers, and underscores, and
  - aren't one of Java's **reserved words**.

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)
  - consist of letters, numbers, and underscores, and
  - aren't one of Java's **reserved words**.

```
x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
true
C_19_H_14_O_5_S
```

Variable Names

- Legal names for variables
  - begin with a letter or an underscore (_)
  - consist of letters, numbers, and underscores, and
  - aren't one of Java's **reserved words**.

```
x
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
C_19_H_14_O_5_S
```

Variable Naming Conventions

- You are free to name variables as you see fit, but there are conventions.
- Names are often written in **lower camel case:**

capitalizeAllWordsButTheFirst

Variable Naming Conventions

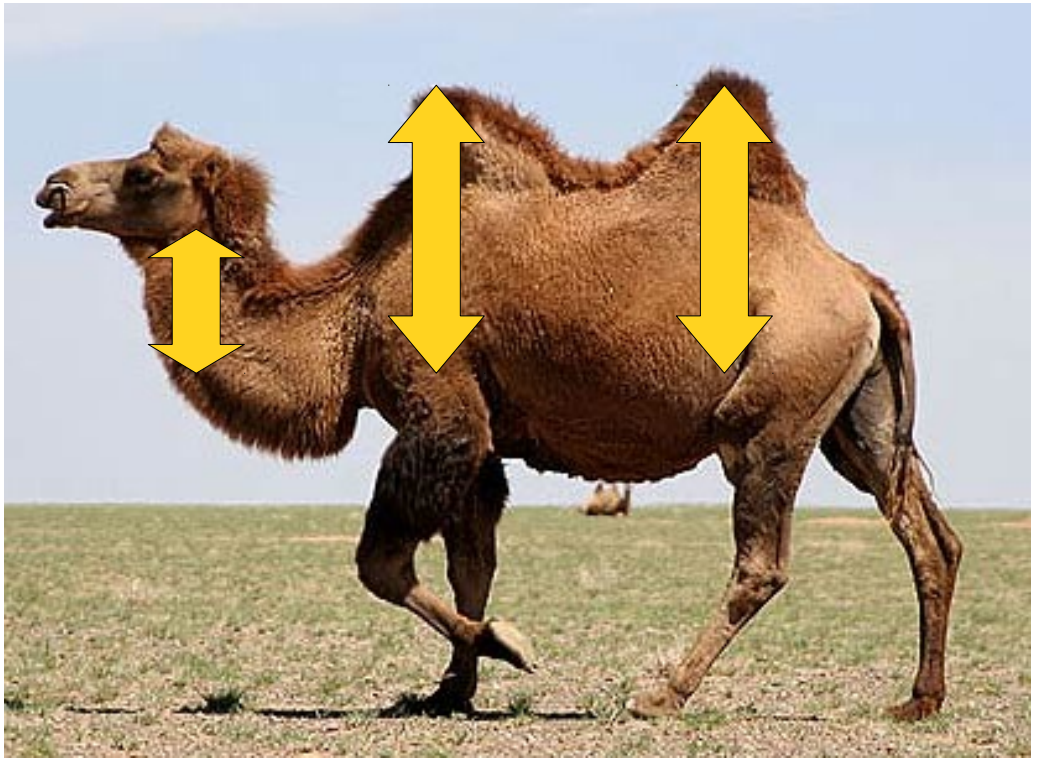
- You are free to name variables as you see fit, but there are conventions.
- Names are often written in **lower camel case:**

capitalizeAllWordsButTheFirst

Variable Naming Conventions

- You are free to name variables as you see fit, but there are conventions.
- Names are often written in **lower camel case:**

capitalizeAllWordsButTheFirst



Variable Naming Conventions

- You are free to name variables as you see fit, but there are conventions.
- Names are often written in **lower camel case:**

  capitalizeAllWordsButTheFirst

- Choose names that describe what the variable does.
  - If it's a number of voters, call it **`numberOfVoters`**, **`numVoters`**, **`voters`**, etc.
  - Don't call it **`x`**, **`volumeControl`**, or **`severusSnape`**

Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:

Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers.

Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers.
  - `double`: Real numbers.

Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers.
  - `double`: Real numbers.



Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers. **(counting)**
  - `double`: Real numbers.



Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers. **(counting)**
  - `double`: Real numbers. **(measuring)**



Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers. **(counting)**
  - `double`: Real numbers. **(measuring)**
  - `char`: Characters (letters, punctuation, etc.)

Types

- The **type** of a variable determines what can be stored in it.
- Java has several **primitive types** that it knows how to understand:
  - `int`: Integers. **(counting)**
  - `double`: Real numbers. **(measuring)**
  - `char`: Characters (letters, punctuation, etc.)
  - `boolean`: Logical true and false.

Values

```
137
```

```
int numVotes
```

```
0.97333
```

```
double fractionVoting
```

```
0.64110
```

```
double fractionYes
```

Declaring Variables

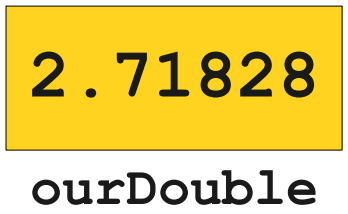
Declaring Variables

```
public void run() {

}
```

Declaring Variables

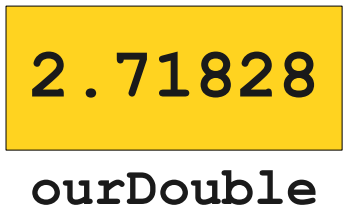
```
public void run() {
    double ourDouble = 2.71828;
}
```

Declaring Variables




```
public void run() {
    double ourDouble = 2.71828;
}
```

Declaring Variables

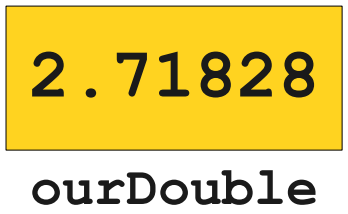



```
public void run() {
    double ourDouble = 2.71828;
}
```

The syntax for declaring a variable with an initial value is

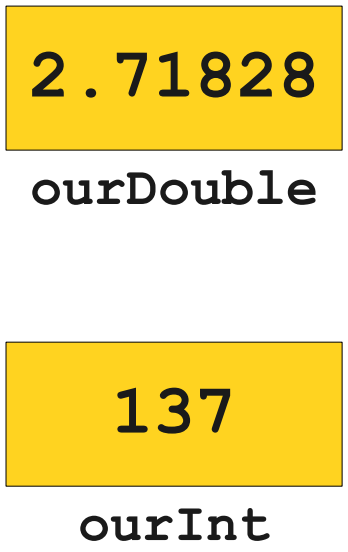
type name = ***value***;

Declaring Variables



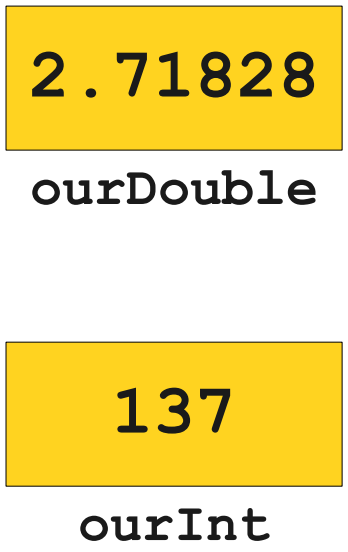

```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

}
```

Declaring Variables




```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

}
```

Declaring Variables




```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;

}
```

Variables can be declared without an initial value:

***type name*;**

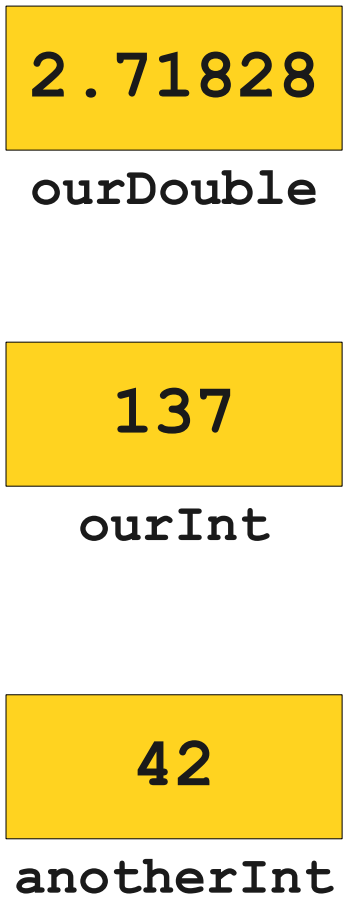
Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;



}
```

Declaring Variables




```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;
}
```

An assignment statement has the form

variable = ***value***;

This stores ***value*** in ***variable***.

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;



}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;

}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;

}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;

}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
    ourInt = 1258;
}
```

Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
    ourInt = 1258;
}
```

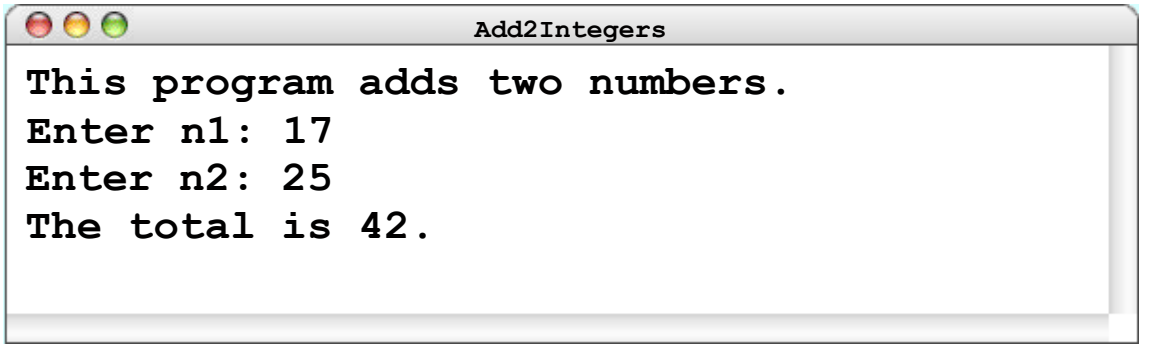
Declaring Variables


```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
    ourInt = 1258;
}
```

The **Add2Integers** Program

```
class Add2Integers extends ConsoleProgram {
   public void run() {
      println("This program adds two numbers.");
      int n1 = readInt("Enter n1: ");
      int n2 = readInt("Enter n2: ");
      int total = n1 + n2;
      println("The total is " + total + ".");
   }
}
```

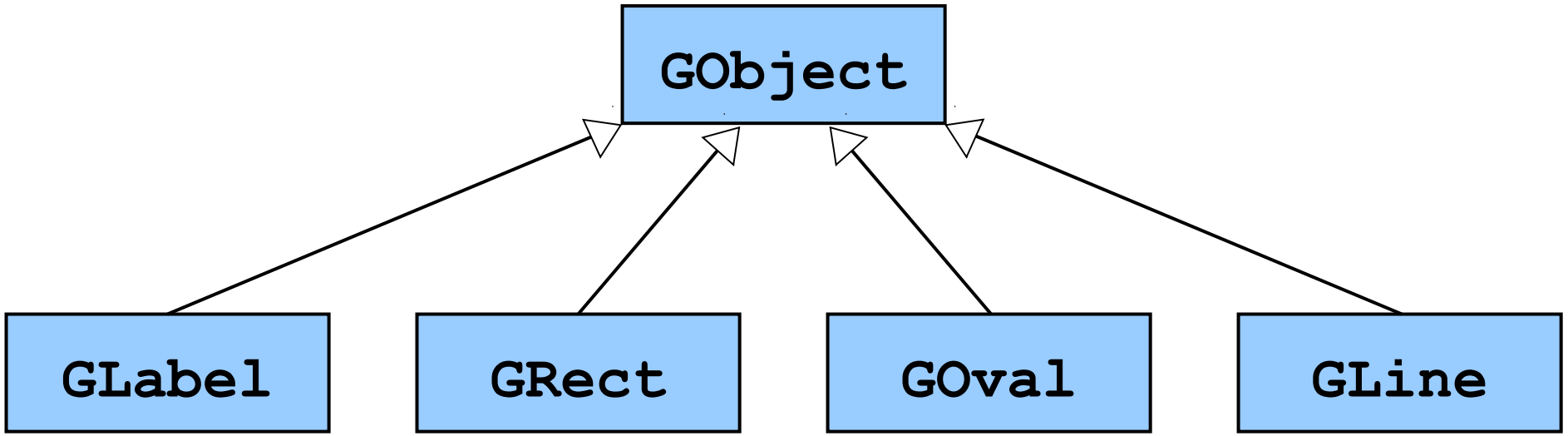
n1	n2	total
17	25	42



Graphic courtesy of Eric Roberts

The `GObject` Hierarchy

The classes that represent graphical objects form a hierarchy, part of which looks like this:



Graphic courtesy of Eric Roberts

Sending Messages to a `GLabel`

```
public class HelloProgram extends GraphicsProgram {
   public void run() {
      GLabel label = new GLabel("hello, world", 100, 75);
      label.setFont("SansSerif-36");
      label.setColor(Color.RED);
      add(label);
   }
}
```

label

hello, world

Graphic courtesy of Eric Roberts

Objects and Variables

- Variables can be declared to hold objects.
- The type of the variable is the name of the class:
  - `GLabel label;`
  - `GOval oval;`
- Instances of a class can be created using the **new** keyword:
  - `GLabel label = new GLabel("Y?", 0, 0);`

Sending Messages

- To call a method on an object stored in a variable, use the syntax

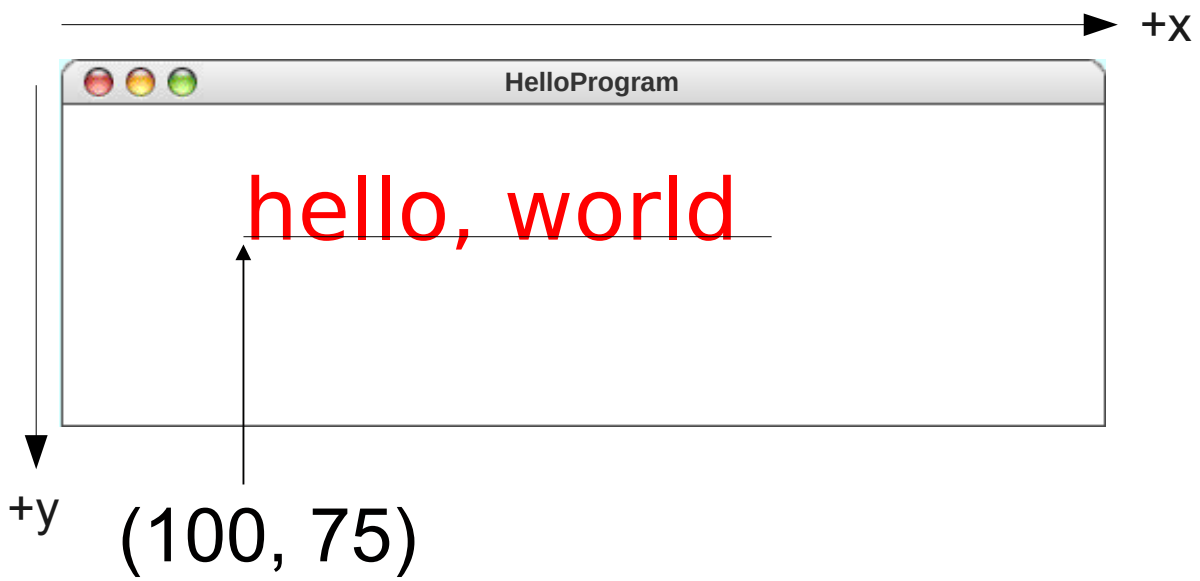
***object*.*method*(*parameters*)**

- For example:

```
label.setFont("Comic Sans-32");
label.setColor(Color.ORANGE);
```

Graphics Coordinates

- Origin is upper left.
- x coordinates increase from left to right.
- y coordinates increase from top to bottom.
- Units are **pixels** (dots on the screen).
- **`GLabel`** coordinates are baseline of first character.



Graphic courtesy of Eric Roberts

Operations on the `GObject` Class

The following operations apply to all `GObjects`:

object`.setColor(`*color*`)` Sets the color of the object to the specified color constant.
object`.setLocation(`*x*`, `*y*`)` Changes the location of the object to the point (*x*, *y*).
object`.move(`*dx*`, `*dy*`)` Moves the object on the screen by adding *dx* and *dy* to its current coordinates.

Standard color names defined in the `java.awt` package:

`Color.BLACK`	`Color.RED`	`Color.BLUE`
`Color.DARK_GRAY`	`Color.YELLOW`	`Color.MAGENTA`
`Color.GRAY`	`Color.GREEN`	`Color.ORANGE`
`Color.LIGHT_GRAY`	`Color.CYAN`	`Color.PINK`
`Color.WHITE`		

Graphic courtesy of Eric Roberts

Operations on the **`GLabel`** Class

Constructor

`new GLabel(`*text*`, `*x*`, `*y*`)`**

Creates a label containing the specified text that begins at the point (*x*, *y*).

Methods specific to the **`GLabel`** class

label`.`**`setFont(`***font*`)`**

Sets the font used to display the label as specified by the font string.

The font is specified as

"*family*`-`*style*`-`*size*"

family is the name of a font family.
style is either **`PLAIN`**, **`BOLD`**, **`ITALIC`**, or **`BOLDITALIC`**.
size is an integer indicating the point size.

Drawing Geometrical Objects

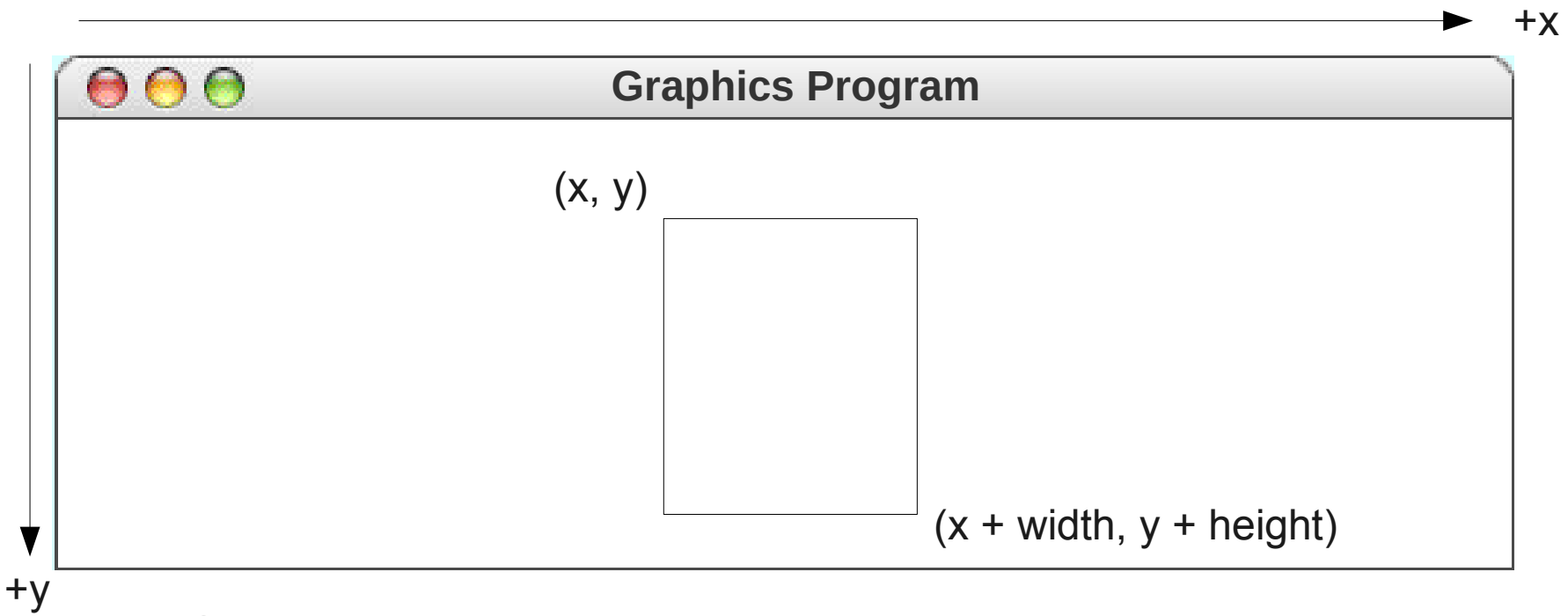
Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

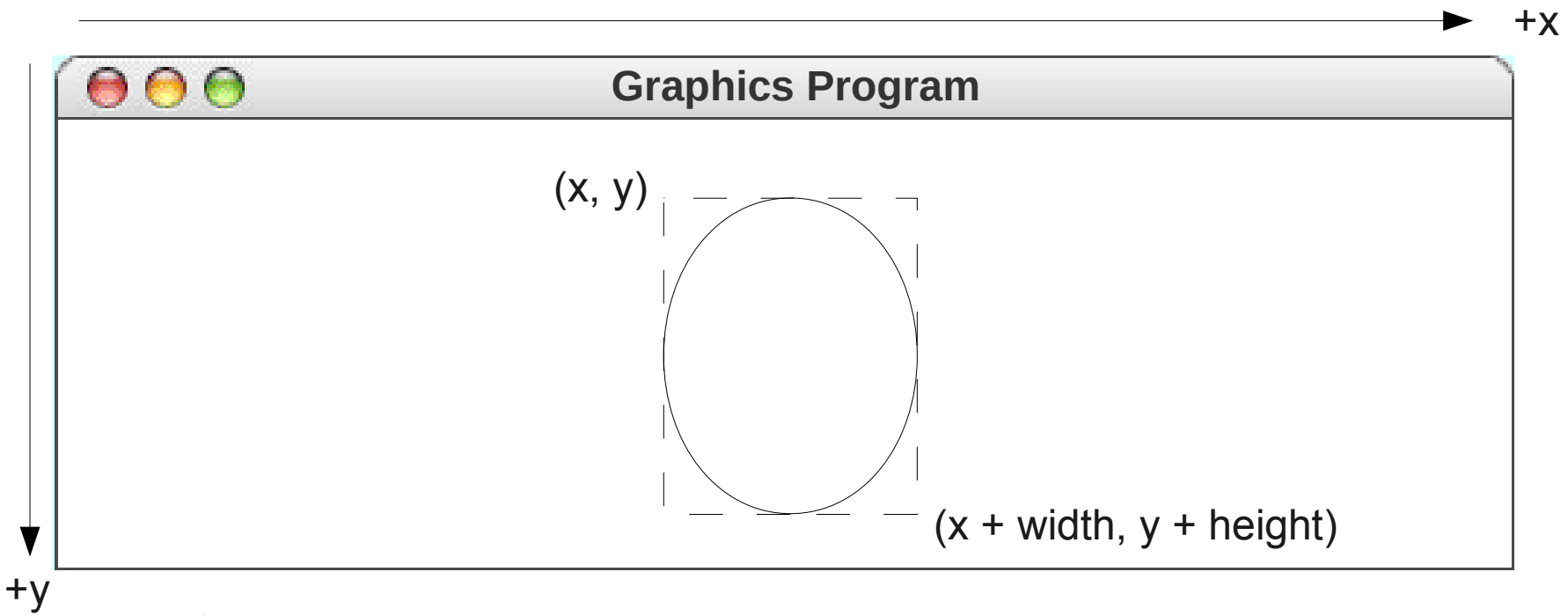
Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates an oval that fits inside the rectangle with the same dimensions.



Graphic courtesy of Eric Roberts

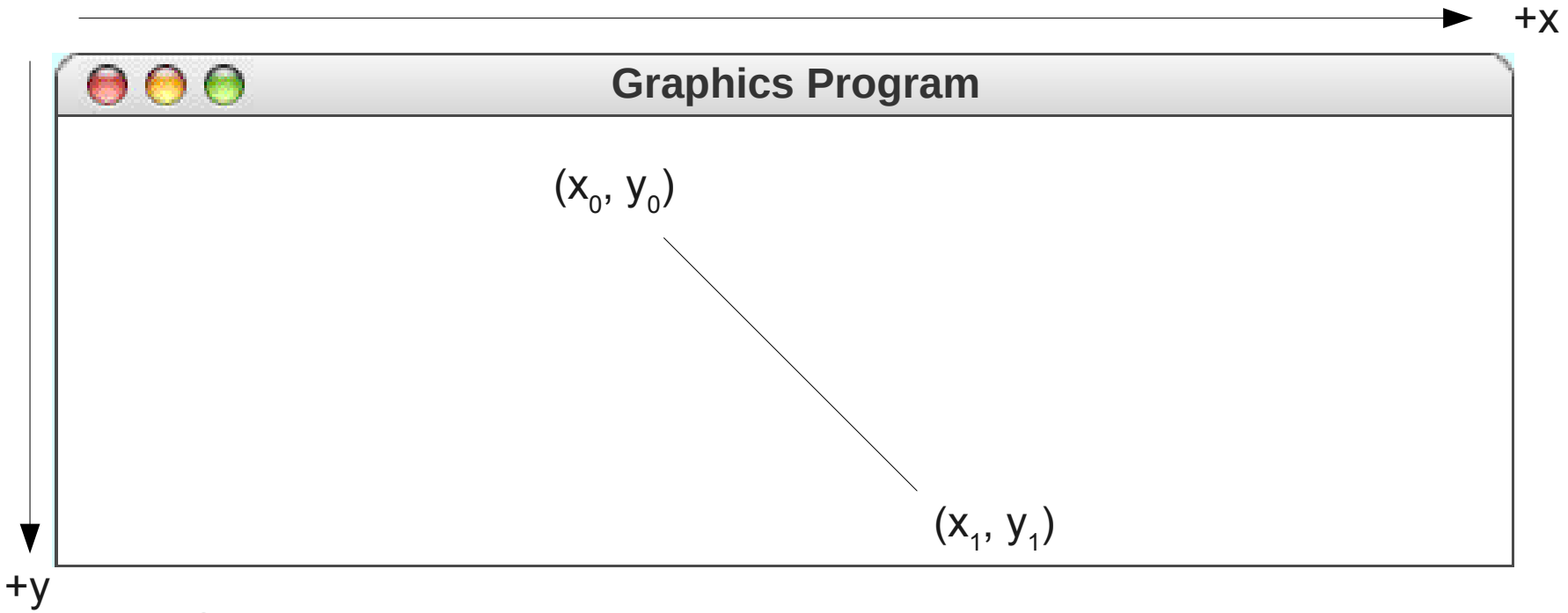
Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`
Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`
Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0`,` y_0`,` x_1`,` y_1`)`
Creates a line extending from (x_0, y_0) to (x_1, y_1).



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0`,` y_0`,` x_1`,` y_1`)`

Creates a line extending from (x_0, y_0) to (x_1, y_1).

Methods shared by the **`GRect`** and **`GOval`** classes

object`.`**`setFilled(`***fill*`)`

If *fill* is **`true`**, fills in the interior of the object; if **`false`**, shows only the outline.

object`.`**`setFillColor(`***color*`)`

Sets the color used to fill the interior, which can be different from the border.

Graphic courtesy of Eric Roberts