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.

- 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?

```c
int numVoters
```
Variables

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

```java
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.

```c
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.

  ```
  137 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?
Variable Names

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

Examples:
- 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 (_)

\[
\begin{align*}
x & \quad \text{x} \\
7\text{thHorcrux} & \quad \text{7thHorcrux} \\
\text{Harry Potter} & \quad \text{Harry Potter} \\
\text{noOrdinaryRabbit} & \quad \text{noOrdinaryRabbit} \\
\text{lots\_of\_underscores} & \quad \text{lots\_of\_underscores} \\
\text{LOUD\_AND\_PROUD} & \quad \text{LOUD\_AND\_PROUD} \\
\text{that\'s\_ACoolName} & \quad \text{that\'s\_ACoolName} \\
\text{true} & \quad \text{true} \\
\text{C\_19\_H\_14\_O\_5\_S} & \quad \text{C\_19\_H\_14\_O\_5\_S}
\end{align*}
\]
Variable Names

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

x
7thHorcrux
Harry Potter
noOrdinaryRabbit
lots_ofUnderscores

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,

\[
\begin{align*}
x & \quad w \\
7\text{thHorcrux} & \quad \text{LOUD\_AND\_PROUD} \\
\text{Harry Potter} & \quad \text{that'sACoolName} \\
\text{noOrdinaryRabbit} & \quad \text{true} \\
\text{lots\_of\_underscores} & \quad \text{C\_19\_H\_14\_O\_5\_S}
\end{align*}
\]
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 7thHorcrux
Harry Potter Harry-Potter
noOrdinaryRabbit noOrdinaryRabbit
lots_of_underscores lots_of_underscores
w LOUD_AND_PROUD LOUD_AND_PROUD
that'sACoolName that'sACoolName
true true
C_19_H_14_O_5_S 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   w
LOUD_AND_PROUD

noOrdinaryRabbit
lots_of_underscores   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

- `int numVotes = 137`
- `double fractionVoting = 0.97333`
- `double fractionYes = 0.64110`
Declaring Variables
Declaring Variables

```java
public void run() {
}
```
Declaring Variables

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

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

The syntax for declaring a variable with an initial value is

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

```
2.71828
ourDouble
```
Declaring Variables

```java
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;
}
```
Declaring Variables

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

2.71828
ourDouble

137
ourInt
Declaring Variables

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

    int anotherInt;
}
```
Declaring Variables

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

    int anotherInt;
}
```
Declaring Variables

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

    int anotherInt;

    Variables can be declared without an initial value:
    type name;
}
```
Declaring Variables

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

    int anotherInt;
    anotherInt = 42;
}
```
Declaring Variables

```java
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

\[ \text{variable} = \text{value}; \]

This stores \text{value} in \text{variable}. 
Declaring Variables

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
}

2.71828
ourDouble

137
ourInt

42
anotherInt
Declaring Variables

```java
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;
    int anotherInt;
    anotherInt = 42;
    ourInt = 13;
}
```
Declaring Variables

```java
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;
    int anotherInt;
    anotherInt = 42;
    ourInt = 13;
}
```
Declaring Variables

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;
}
```
Declaring Variables

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;
}
```
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;
}
public void run() {
    double ourDouble = 2.71828;
    int ourInt = 137;

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
}

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
}
Declaring Variables

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
}
```
Declaring Variables

```java
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

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

    int anotherInt;
    anotherInt = 42;

    ourInt = 13;
    ourInt = ourInt + 1;

    anotherInt = ourInt;
    ourInt = 1258;
}
```
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 + ".");
    }
}

<table>
<thead>
<tr>
<th>n1</th>
<th>n2</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>25</td>
<td>42</td>
</tr>
</tbody>
</table>

This program adds two numbers.
Enter n1: 17
Enter n2: 25
The total is 42.
The GObject Hierarchy

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

![Diagram showing the GObject hierarchy with classes GLLabel, GRect, GOval, and GLine.]
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);
    }
}

Sending Messages to a GLabel

hello, world

hello, world

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

  \texttt{object.method(parameters)}

- For example:

  \begin{verbatim}
  label.setFont("Comic Sans-32");
  label.setColor(Color.ORANGE);
  \end{verbatim}
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.
Operations on the **GObject** Class

The following operations apply to all **GObjects**:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setColor(color)</code></td>
<td>Sets the color of the object to the specified color constant.</td>
</tr>
<tr>
<td><code>object.setLocation(x, y)</code></td>
<td>Changes the location of the object to the point ((x, y)).</td>
</tr>
<tr>
<td><code>object.move(dx, dy)</code></td>
<td>Moves the object on the screen by adding (dx) and (dy) to its current coordinates.</td>
</tr>
</tbody>
</table>

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

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

Graphic courtesy of Eric Roberts
Operations on the `GLabel` Class

**Constructor**

```java
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**

```java
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.

Graphic courtesy of Eric Roberts
Drawing Geometrical Objects
Drawing Geometrical Objects

**Constructors**

```java
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

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

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

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

Creates an oval that fits inside the rectangle with the same dimensions.
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(x₀, y₀, x₁, y₁)
    Creates a line extending from (x₀, y₀) to (x₁, y₁).
```
# Drawing Geometrical Objects

## Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>new GRect(x, y, width, height)</code></td>
<td>Creates a rectangle whose upper left corner is at ((x, y)) of the specified size</td>
</tr>
<tr>
<td><code>new GOval(x, y, width, height)</code></td>
<td>Creates an oval that fits inside the rectangle with the same dimensions.</td>
</tr>
<tr>
<td><code>new GLine(x_0, y_0, x_1, y_1)</code></td>
<td>Creates a line extending from ((x_0, y_0)) to ((x_1, y_1)).</td>
</tr>
</tbody>
</table>

## Methods shared by the GRect and GOval classes

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setFill(true)</code></td>
<td>If <code>true</code>, fills in the interior of the object; if <code>false</code>, shows only the outline.</td>
</tr>
<tr>
<td><code>object.setFillColor(color)</code></td>
<td>Sets the color used to fill the interior, which can be different from the border.</td>
</tr>
</tbody>
</table>