Starting Off in Java

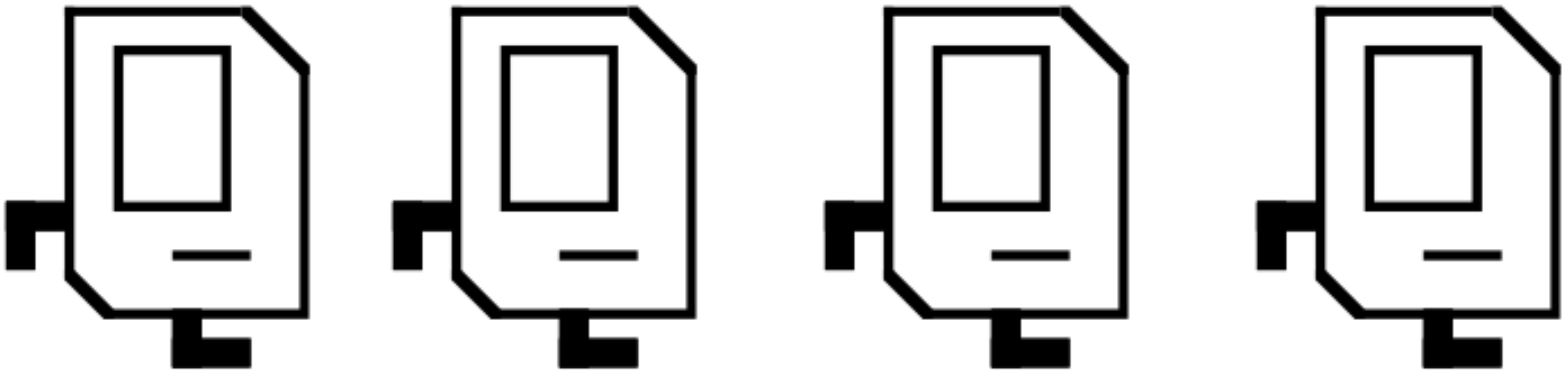
Announcements

- Programming Assignment 1 is due this Friday.
  - Recommendation: Try to complete a first draft of all four programs by this Wednesday so that you have time to debug and test them at the end of the week.
- Section signups closed yesterday at 5PM; section assignments will be announced soon.
  - Missed section signups? Section signups will reopen on Tuesday afternoon, though with more limited options.
- LaIR hours start tonight! Open 6PM – midnight from Sunday through Thursday.
  - Located on the second floor of Tresidder.

Outline for Today

- **Programming in Java**
  - What does Java look like outside of Karel?
- **Variables, Types, and Values**
  - Storing information for later.
- **Expressions**
  - Mathematical operations in Java.
- **Graphics**
  - Oooh! Shiny!

A Farewell to Karel



Welcome to Java!

What is Java?

- Java is an industrial programming language used to build large applications.
- Used in web servers, Android phones, desktop applications, etc.
- Extremely common; easily one of the most popular programming languages in use today.

Transitioning to Java

- The Karel code that you've written so far is perfectly legal Java code.
- However, there are many key aspects of the Java programming language we haven't yet touched on.
- The remainder of this class will focus on those more general Java features and how to make the most use of them.

Our First Java Program

Dissecting our Program

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summa
      println("The sum of th
   }
}
```

The boilerplate code here looks similar to a Karel program, but there are some differences. Notice that we're now extending `ConsoleProgram` and that the import is different.

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

This is the ***run method***. As with Karel programs, our Java programs start here.

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summa
      println("The sum of those numbers is " + sum);
   }
}
```

Each of these lines of code is called a ***statement***. As with Karel commands, each statement ends with a semicolon.

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values fr
      int n1 = readInt("Ent
      int n2 = readInt("Ent

      // Compute their sum
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

The **println** method (**print line**) displays a line of text on the screen. The quoted text in the parentheses is the ***argument*** to the method.

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the su
      println("The sum of
   }
}
```

These statements are called ***variable declarations***. They allow us to give names to quantities (here, the first two numbers entered and their sum).

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

These are comments.
The /* … */ comments still work in Java; this is another option.

Working with Variables

- The previous program declared three ***variables***: `n1`, `n2`, and `sum`.
- In the previous example, we used these variables to keep track of values that the user entered and to store information for later on.
- Variables are very important in Java, so we'll start with a quick overview of how to use them.

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
7thBookInTheSeries
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
C_19_H_14_O_5_S
```

Variable Names

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

```
x
7thBookInTheSeries
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
C_19_H_14_O_5_S
```

Variable Names

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

```
x
~~7thBookInTheSeries~~
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
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
~~7thBookInTheSeries~~
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
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
~~7thBookInTheSeries~~
~~Harry Potter~~
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
~~that'sACoolName~~
void
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
7thBookInTheSeries
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
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
7thBookInTheSeries
Harry Potter
noOrdinaryRabbit
lots_of_underscores
```

```
w
LOUD_AND_PROUD
that'sACoolName
void
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 some standard conventions.
- Names are often written in ***lower camel case***:

```
capitalizeAllWordsButTheFirst
```

Variable Naming Conventions

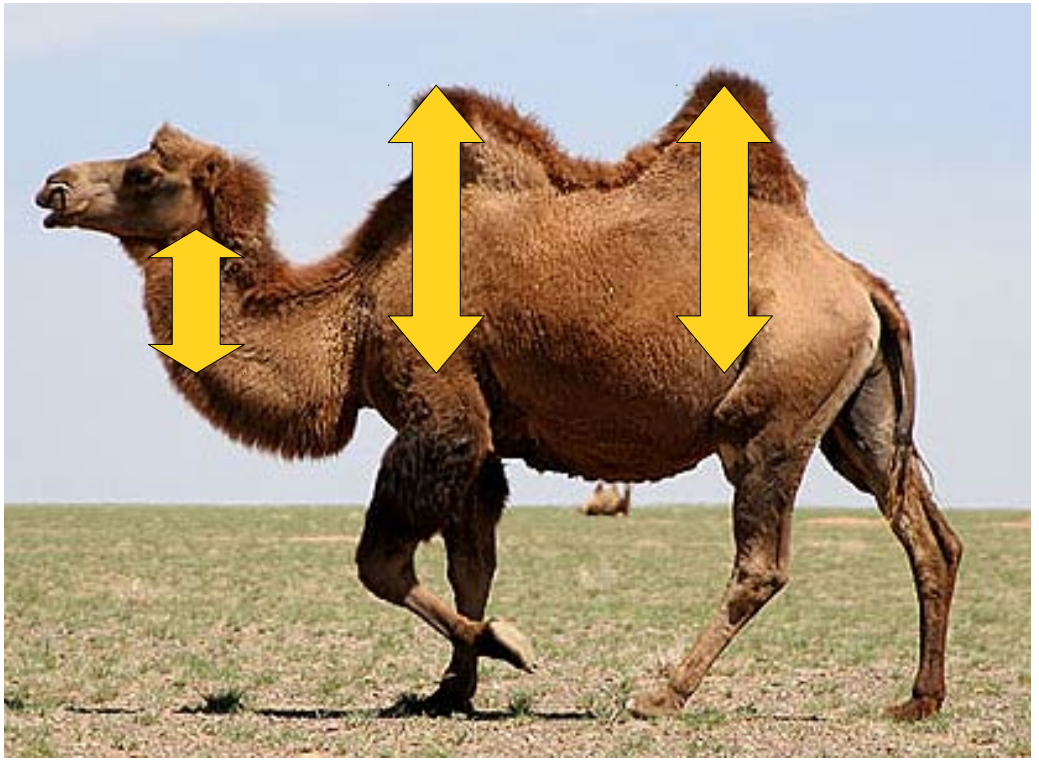
- You are free to name variables as you see fit, but there are some standard conventions.
- Names are often written in ***lower camel case***:

```
capitalizeAllWordsButTheFirst
```

Variable Naming Conventions

- You are free to name variables as you see fit, but there are some standard 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 some standard 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)***
  - `boolean`: Logical true and false.
  - (Plus a few more)

Values

```
137
```

```
int numVotes
```

```
0.97333
```

```
double fractionVoting
```

```
0.64110
```

```
double fractionYes
```

Declaring Variables

- In Java, before you can use a variable, you need to ***declare*** it so that Java knows the name, type, and value.
- The syntax for declaring a variable is

type name = ***value***;

- For example:
  - `int numVotes = 137;`
  - `double pricePerPound = 0.93;`

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

Reading Values

- You can prompt the user for a value by using the `readInt` and `readDouble` methods.
- For example:

```
int numBunnies = readInt("How many bunnies? ");

double weight = readDouble("Each bunny weighs? ");
```

- Notice that there's a space at the end of each of the prompts – we'll see why in a second.

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

```
import acm.program.*;

public class AddTwoIntegers extends ConsoleProgram {
   public void run() {
      println("This program adds two integers.");

      // Read two values from the user.
      int n1 = readInt("Enter first integer: ");
      int n2 = readInt("Enter second integer: ");

      // Compute their sum.
      int sum = n1 + n2;

      // Print out the summation
      println("The sum of those numbers is " + sum);
   }
}
```

Expressions

- Variables and other values can be used in ***expressions***.
- Some familiar mathematical operators:
  - `+` (addition)
  - `-` (subtraction)
  - `*` (multiplication)
  - `/` (division)

Operator Precedence

- Java's mathematical operators have the following precedence:

```
()      (highest)
* /
+ -     (lowest)
```

- Operators of equal precedence are evaluated left-to-right.

Fun with Division

The Mod Operator

- The special operator `%` (called the ***modulus operator*** or ***mod operator***) computes the remainder of one value divided by another.
- *a* `%` *b* is pronounced "*a* mod *b*."
- For example:
  - `15 % 3 = 0`
  - `14 % 8 = 6`
  - `21 % 2 = 1`
  - `14 % 17 = 14`

Rounding Down

- In Java, dividing two `int`s will divide and then round down.
- For example, this will print 3:

```
int value = 7 / 2;

println("The value is " + value);
```

- This might be a bit weird, but there's a good reason for it.

Sharing Cookies


She got more
than me!


Cookies for everyone!

Dividing Doubles

- In Java, dividing two `int`s will divide and then round down.
- Dividing two `double`s will do the division correctly.
- If either operand is a `double`, the division will be done correctly.
- For example, to compute the average of two `int`s `n1` and `n2`, you could write

```
double average = (n1 + n2) / 2.0;
```