Module 1-2

Variables and Data Types

Objectives

- identify and explain fundamental concepts and components of Java Framework
- read and write code that uses variables
- declare a variable and assign a value to a variable
- know and use industry acceptable naming conventions for variables
- choose the appropriate primitive data type to represent different kinds of data in a program
- utilize arithmetic operators to form mathematical expressions
- explain the concept of data conversion (or casting), when it occurs, why it's used
- explain the purpose and use of literal suffixes and why they should be used
- code and test a simple Java using an Integrated Development Environment (IDE).
- perform simple tasks in an IDE such as:
    - Organizing code into projects
    - Understand feedback on syntax errors
    - Utilize the "intellisense" feature of an IDE to assist in code development

History of Java

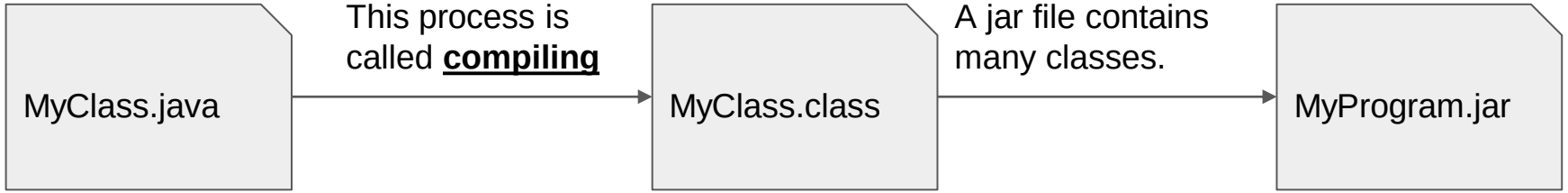
Java is an object oriented language (you will learn what this means later!) developed by Sun Microsystems in 1995. It was originally created by James Gosling. Sun Microsystems was acquired by Oracle Corporation in 2010.

It it is one of the most widely used languages. It consistently ranks in the top 2 in terms of popularity. (Source: https://stackify.com/popular-programming-languages-2018/).

Java Features

- It shares similar syntax with C/C++
- It can be used to create desktop, mobile, and web applications.
- Unlike other languages, Java is not run natively on a given device, it is instead executed in a Java Virtual Machine (JVM) / Java Runtime Environment (JRE).... think of this as a virtual computer running inside your computer.
- Advantages of this model:
  - Oracle (not you :)) is responsible for the lifecycle of the JRE / JVM.
  - Developers are therefore freed from the idiosyncrasies of each individual platform! (i.e. pc's, macs, mobile devices, refrigerators, etc)
  - WORA – write once, run anywhere

Java Development Workflow



Compiling is the process by which source code (in this case the file MyClass.java) is transformed into a language the computer can understand.

In the past, the command **javac** was used to compile code, since the advent of modern development tools, manually typing this command is no longer needed.

Java Bytecode Example

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

This is what we will write

A Java compiler might translate the Java code above into byte code as follows, assuming the above was put in a method:

```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic       #84; // Field java/lang/System.out:Ljava/io/PrintStream;
34:  iload_1
35:  invokevirtual   #85; // Method java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

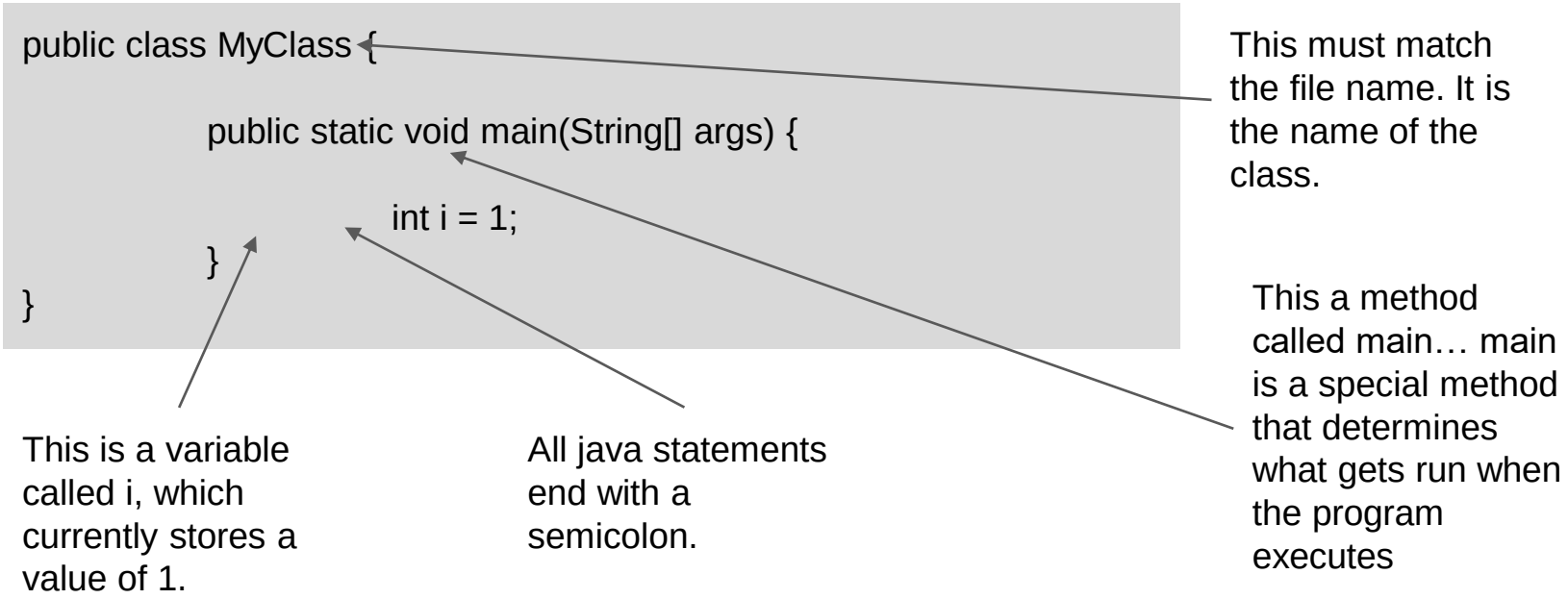
This is what the java translates it to

Source: https://en.wikipedia.org/wiki/Java_bytecode

Java Program Structure

- The main unit of organization in Java is a class. Here is a simple example:

  This file must be called MyClass.java. Its contents are as follows:



Hello World Program

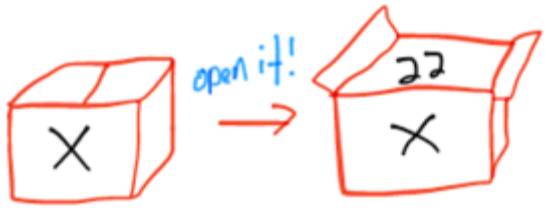
```
package com.techelevator;

public class HelloWorld {
    public static void main(String[] args) {
        // Prints out Hello World
        System.out.println("Hello World");
    }
}
```

Java Variables

- A variable is a representation for something that might change.

  Consider the math formula: $c^2 = a^2 + b^2$

  We don’t know for a fact what a, b, or c are, they can take different values depending on the situation. Therefore, a, b, and c are like containers that could take on different values at different times. These containers are called **variables**.



Data Types

- In order for computer to store data, must know what type it is holding
- All data is represented by bits
  - Bit is a switch
  - 2 states – on or off
- Byte
  - Amount of computer storage needed to store 1 character
  - 8 bits
- ASCII
  - American Standard Code for Information Interchange
  - 1 Byte – 256 different combinations
- Unicode
  - Universal Character encoding
  - 2 Bytes – 65,536 different combinations (16 bit)
  - 32 bits also available

Java Variables: Declaring and Assigning Values.

- This is what a java variable declaration looks like:

  int i = 0;

  Here, we have declared a variable called i of type integer, we have also given it an initial value of zero. Assigning values is accomplished using equals (=), the assignment operator.

- Consider this:

  ```
  int i = 0;
  i = 1;
  ```

  Here, we have declared a variable called i of type integer, we gave it an initial value of zero, but then changed the value of the variable to 1.

Java Variables: Data Types.

Data Type	Description	Example
int	Integers (whole numbers)	int i = 1;
double	Decimals	double x = 3.14;
float	Also decimals, but older format. Avoid, use doubles instead.	float x = 3.14f; // Note that to declare a float you must explicitly state that the number is a float by appending the f.
char	One character. **Note the single quotes**.	char myChar = ‘a’;
boolean	True or false	boolean isItTurnedOn = true; boolean paidBills = false;
String	A bunch or characters. **Note the double quotes**.	String name = “Minnie”; String fullName = name + “ Mouse”;

Java Variables: Rules & Conventions

Conventions:

- Ideally, variables should be named using “Camel Case.” Examples of variable names: *playerOneScore*, *cityTemperature*, *shirtSize,* etc. (See the pattern?)
- Ideally, variables never start with an upper case.
- Variable names should be descriptive and of reasonable length.

Rules:

- Variables can begin with an underscore (_), a dollar sign ($), or a letter.
- Subsequent characters can be letters, numbers, or underscores.
- Variable names cannot be the same as java keywords.

Let’s code!

Working with Numbers: Basic Operators

- Math operators can be used to perform basic arithmetic between two numeric variables (From the previous slides, variables of type int, float, and double are examples).
- These are the basic operators: **+** (addition), **-** (subtraction), ***/*** (division), **%** (modulus, aka remainder).
- The basic operators can be combined with the assignment operator to store result calculations, for example: **int i = 4 + 6**;

Working with Numbers: Order of Operations

For now, order of operations is almost the same as normal arithmetic: **P**lease **E**xcuse **M**y **D**ear **A**unt **S**ally (this will get more complex as we introduce more Java concepts)

Category	**Operators**
parenthesis	do first
multiplicative	* or / or %
additive	+ or -
assignment	=

Working with Numbers: Example 1

- Express the following English statement in Java: I paid for an item that cost $8.50 with a $10.00 bill, how much change would I get in return?

```
public class MyClass {

		public static void main(String[] args) {

				double price = 8.50;
				double payment = 10.00;
				double change = payment - price;
				System.out.println(change);
		}
}
```

Working with Numbers: Example 2

- Let’s try something a little bit more complex: We can convert degrees in Fahrenheit to Celsius using the following formula: $(T_F - 32) \times (5/9) = T_C$. How much is 98.6 degrees Fahrenheit in Celsius?

```
public class MyClass {

        public static void main(String[] args) {

                double tempInF = 98.6;
                double tempInC = (tempInF -32.0) * (5.0/9.0);
                System.out.println(tempInC);
        }
}
```

Working with Numbers: Type Conversion

ints, doubles and floats can be used together in the same statement, but Java will apply certain rules:

- Mixed mode expressions are automatically promoted to the higher data type (in this case, a double):

```
public class MyClass {

        public static void main(String[] arg

                int myInt = 4;
                double myDouble = 2.14;

                int firstAttempt = myInt - myDouble; // Won't work, Java will complain!
        } }
```

The result of myInt and myDouble is promoted to a double, it will no longer fit in firstAttempt, which is a int.

Working with Numbers: Type Conversion

(continued from previous page)...

- We can overcome this problem by doing a cast:

  int secondAttempt = (int) (myInt - myDouble);

- We can also overcome this problem by making the variable secondAttempt a double:

  double secondAttempt = myInt - myDouble;

Working with Numbers: Type Conversion

Remember this problem?

```
public class MyClass {

		public static void main(String[] args) {

				double tempInF = 98.6;
				double tempInC = (tempInF -32.0) * (5/9);
				System.out.println(tempInC);
		}
}
```

Note that instead of 5.0/9.0, it is now 5/9, and when run, the result is 0.0. Using the rules we just discussed, can you figure out why?

Combining Strings

The plus sign can also be used with Strings:

```
public class MyClass {

        public static void main(String[] args) {

                String firstName = “Carl”;
                String lastName = “Jung”;
                String comma = "
                String combinedName = lastName + “, ” + firstName;
                System.out.println(combinedName);
        }
}
```

The following code will print ***Jung, Carl***. This process is known as **concatenation**.