Module 1-2

Variables and Data Types

Module 1 Day 2

Can/Do you?

1. … explain the fundamental concepts & components of Java?
2. … read and write code that uses variables?
3. … declare a variable and assign a value to it?
4. … know and use industry accepted naming conventions?
5. … choose an appropriate primitive type to represent varying data?
6. … use arithmetic operators to form mathematical expressions?
7. … explain casting/data conversion, when it occurs, and why it's used?
8. … explain the purpose of literal suffixes and when to use them?
9. … code and test(debug) a simple program in the IDE?
10. … perform simple IDE tasks such as:
    - Understand basic syntax error feedback
    - Use Intellisense

HelloWorld: The Developer's Right of Passage

```
package com.techelevator;

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

Data Type and Operator Quick Reference

Java	Range
`boolean`	`true` or `false`
`byte`	`-127` to `127`
`char`	`\u000` to `\uffff` ('a', 'b', etc.)
`int`	`-2^31` to `2^31`
`float`	`-3.4x10^38` to `3.4x10^38`
`double`	`±5.0 x 10^-324` to `±1.7 x 10^308`
`long`	`-2^63` to `2^63`

3. Arithmetic Operators

Operator	Description	Example
`+`	Adds two operands	15 + 2 = 17
`-`	Subtracts two operands	15 - 2 = 13
`*`	Multiplies two operands	15 * 2 = 30
`/`	Divides two operands	15 / 2 = 7
`%`	Finds the remainder after division	15 % 2 = 1

History of Java

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

Designed as a Write Once, Run Anywhere (WORA) language.

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 freed from the idiosyncrasies of each individual platform! (i.e. pc's, macs, mobile devices, refrigerators, etc) … WORA!

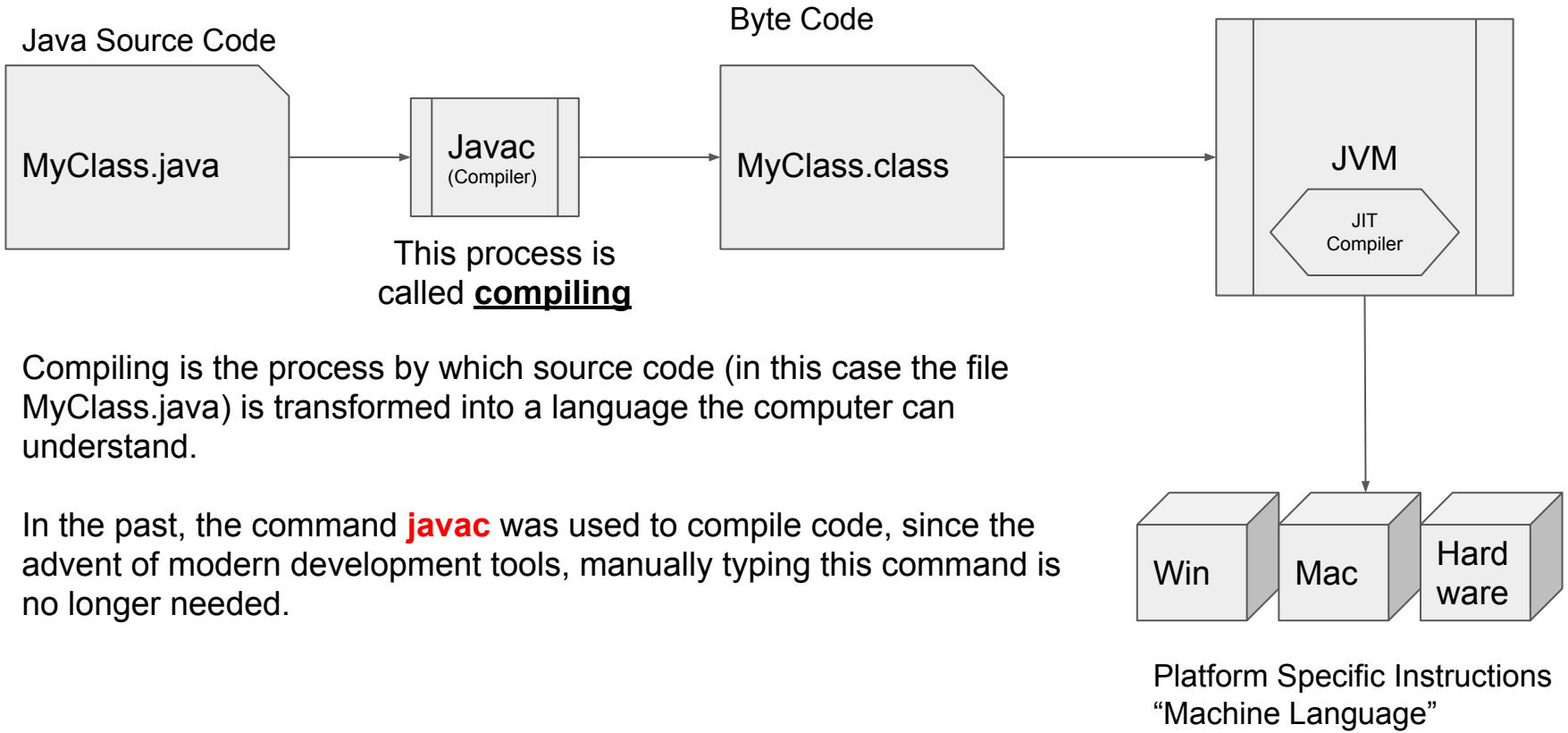
Java Compared

- C/C++
  - Similar Syntax
  - Java eliminated undesirable features
- JavaScript
  - Developed by Netscape in 1995 as Mocha then LiveScript
  - No technical relationship
  - Name is the result of marketing agreement to bundle Sun's Java runtime with the then-dominant browser, Netscape.
- C#
  - Released by Microsoft with .NET platform in 2001
  - Similar features and syntax
  - C# is Microsoft’s answer/response to the release of Java

JVM Architecture

- **JVM (Java Virtual Machine)**
  - A part of the JRE (Java Runtime Environment)
  - Java programs run in a JVM
  - Multiple implementations of JRE to allow WORA/platform independence
- **javac** (pronounced "java-see") is the primary Java compiler included in the Java Development Kit (JDK)
- The javac tool reads Java code (human readable) and compiles it into bytecode class files.
- **Bytecode** is optimized for and interpreted by the JVM

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);
}
```

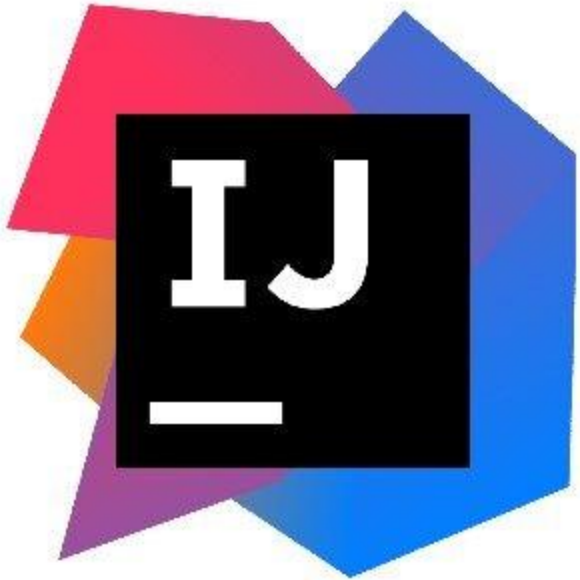
This is what we will write

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


IJ

IDEs & IntelliJ: Why we use them

- Organize code into projects
- Increase Efficiency – faster coding with less effort
  - Immediate feedback for syntax errors
  - Code assistance through **intellisense** (Code Completion in IntelliJ)
- Debugging
  - Allows us to “step through” a program as it is running to inspect variables and memory
- Cannot automatically fix errors, still need knowledge to code efficiently

Ready to Explore?

… Let’s go!

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:

```
public class MyClass {

        public static void main(String[] args) {

                int i = 1;
        }
}
```

This must match the file name. It is the name of the class.

This a method called main… main is a special method that determines what gets run when the program executes

This is a variable called i, which currently stores a value of 1.

All java statements end with a semicolon.

Java Variables: Declaring and Assigning Values.

- In math, a variable is a representation for something that might change.
- In programming, it is thought of as a container
- This is what a java variable declaration looks like:

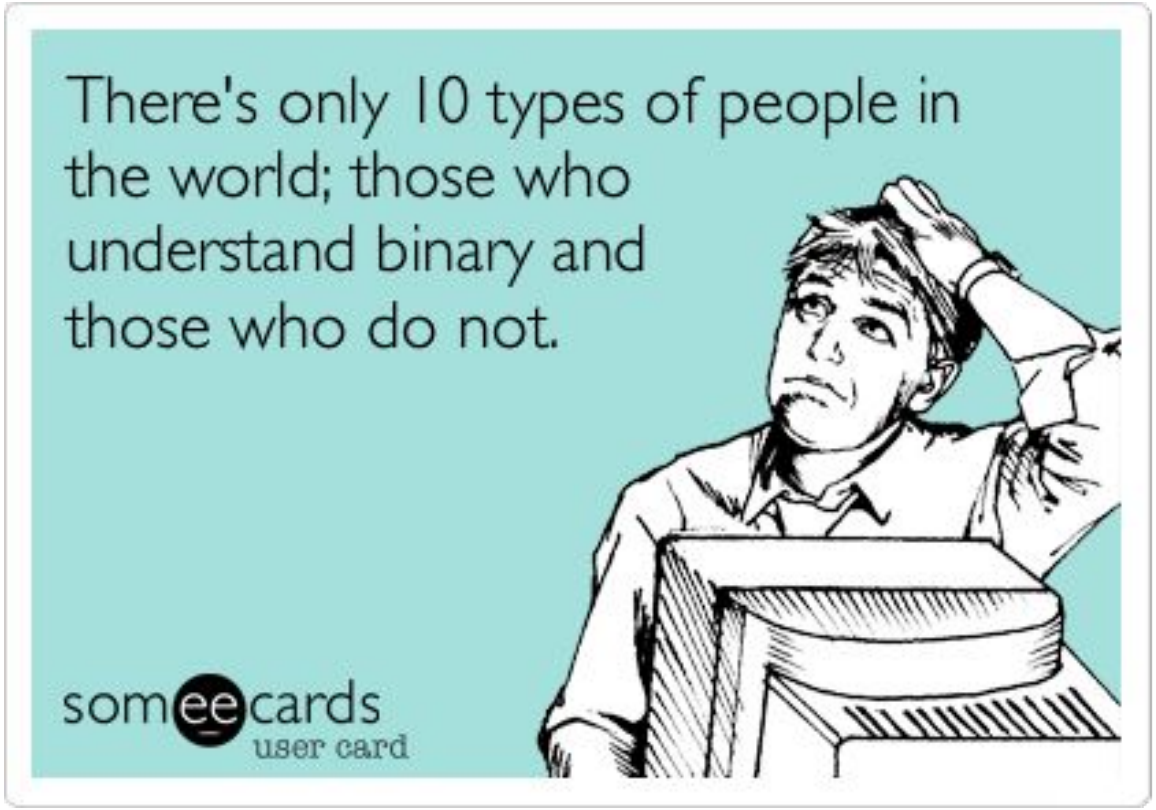
  int i = 0;

  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 (=).

- 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 i.


There's only 10 types of people in the world; those who understand binary and those who do not.
someecards
user card

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’;
String	A bunch or characters. **Note the double quotes**.	String myName = “Horatio”; String mySentence = “This is a beautiful day.”;
boolean	True or false	boolean isItTurnedOn = true; boolean paidBills = false;
byte	Signed small integer storage from -127 to 127	byte eggsInCarton = 11;

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 should 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.

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), **%** (modulo, 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: 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), **%** (modulo, 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 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)

- Anything inside **p**arentheses first.
- **E**xponents
- **M**ultiplication
- **D**ivision
- **A**ddition
- **S**ubtraction

Note that even though the acronym suggests that multiplication outranks division, and addition outranks subtraction, multiplication and division have the same priority, likewise addition and subtraction have the same priority.

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:

- Bytes and Shorts are automatically promoted to an int, when used together. Consider this code:

```
public class MyClass {

        public static void main(String[] args) {

                byte myByte = 4;
                byte myOtherByte = 1;

                byte firstAttempt = myByte + myOtherByte; // Won't work, Java will complain!
        } }
```

The result of myByte and myOtherByte is promoted to an int, it will no longer fit in firstAttempt, which is a byte.

Working with Numbers: Type Conversion

(continued from previous page)...

- We can overcome this problem by doing a cast:

  byte secondAttempt = (byte) (myByte + myOtherByte);

- We can also overcome this problem by making the variable secondAttempt an int:

  int secondAttempt = myByte + myOtherByte;

Working with Numbers: Type Conversion

Two additional rules:

- A long with anything else will become a long.
- A double with anything else becomes a double.

```
public class MyClass {

	public static void main(String[] args) {
		long thisVariable = 4;
		int anotherVariable = 7;

		// This won't work:
		int someVariable1 = thisVariable + anotherVariable;

		// This is perfect:
		long someVariable2 = thisVariable + anotherVariable;
	}}
```

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 combinedName = lastName + “, ” + firstName;
		System.out.println(combinedName);
	}
}
```

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