
LETS LEARN JAVA
GET SET GO

Training Plan - Basics

Topics to be covered
Introduction to Java Difference between Java & C++ Setting up JDK Environment & IDE Understanding ECLIPSE IDE – perspectives, views First program, Understanding the program, How to compile, How to Execute How to format code, Keyboard Shortcuts in ECLIPSE and Configuring Save Actions Different ways of printing data to console. Various String output and formatting options What are variables. How they are named What are the data types Getting input from console and printing it What are methods. How method works and what are arguments and return statements

What is Java? And why is it so popular

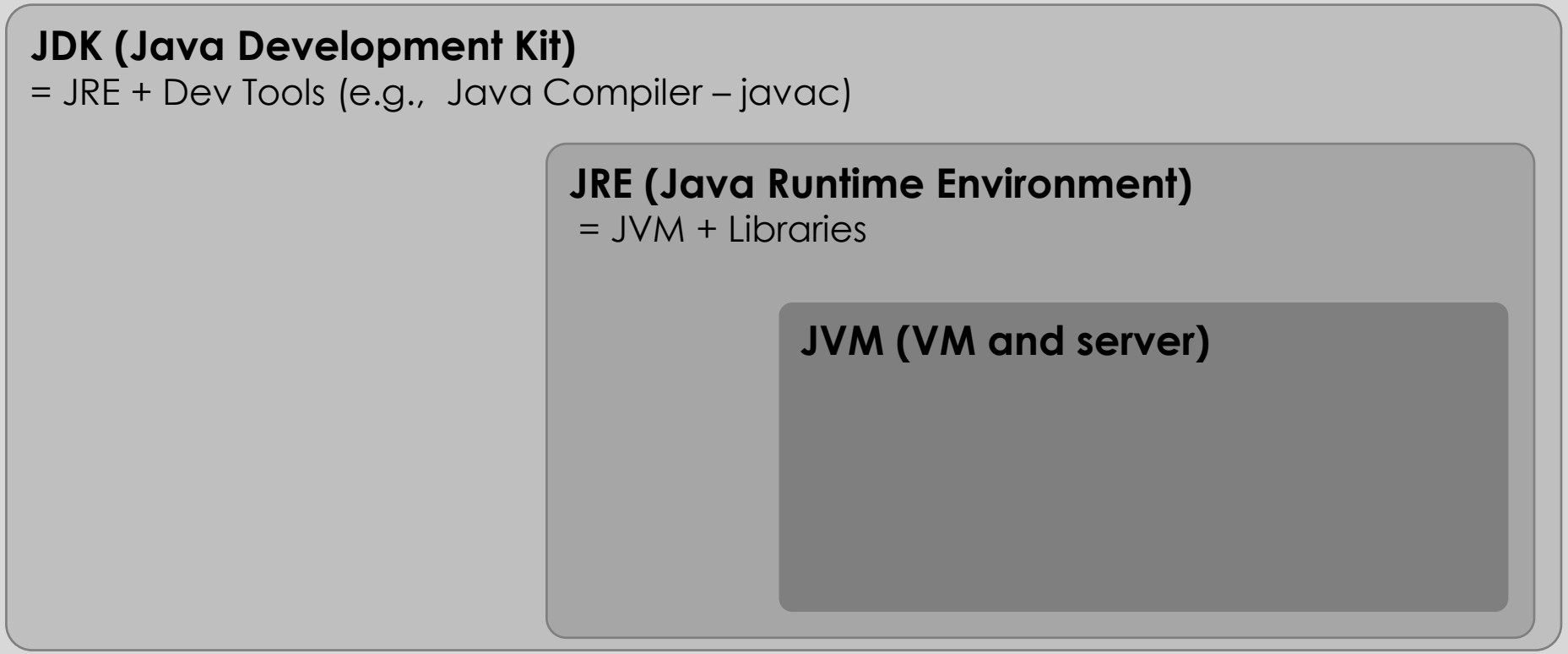
- Java – A programming language and a platform (JRE)
- Language with maximum jobs in the market (Search for "most popular programming languages 2020")
- **JDK and JRE** – what is the difference. What is JVM

JDK = JRE + Development Tools (Javadoc, Javac)
JRE = JVM + Libraries

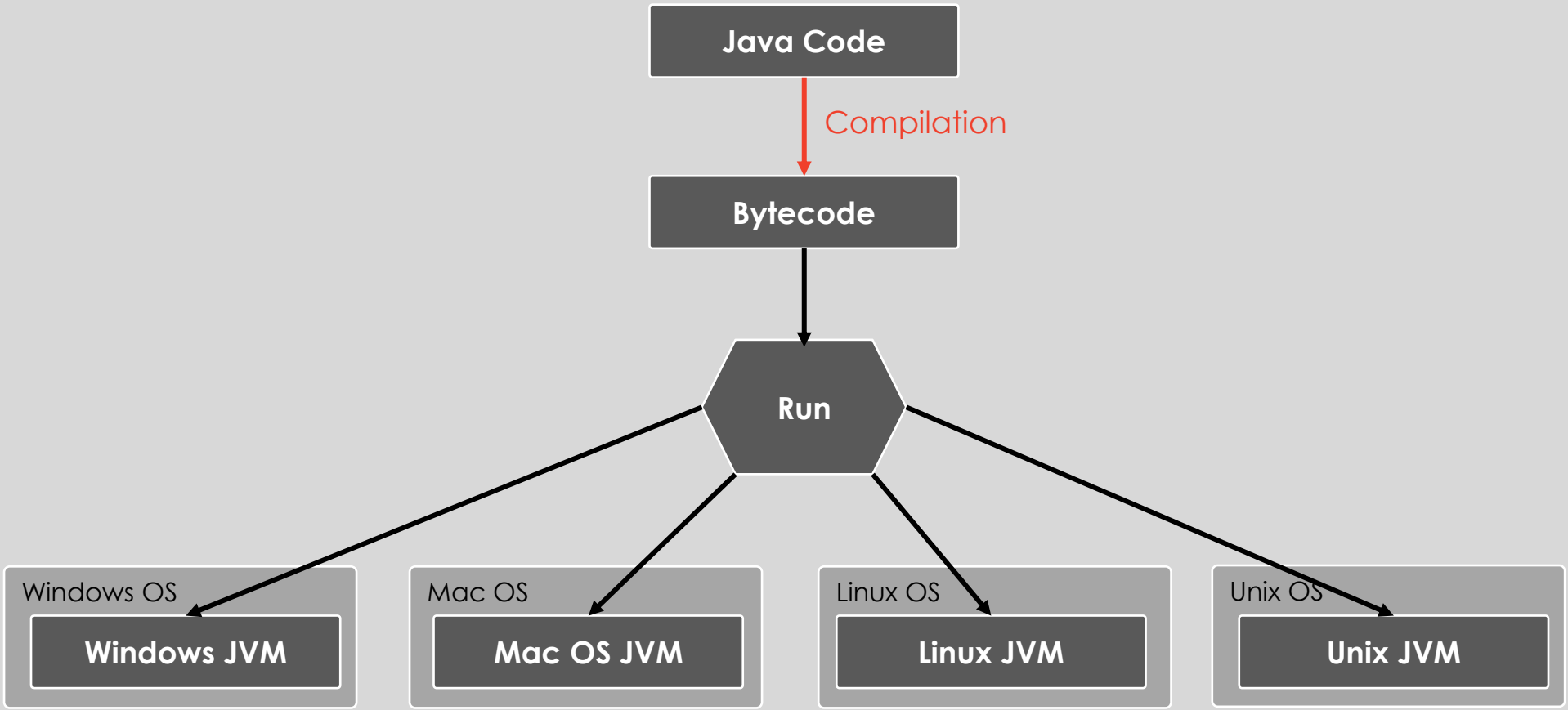
- **James** Bond + Ryan **Gosling** – invented Java in 1995
- Java 1, 1.1, 1.2…..1.4, Java 5, 6, 7, **8**, 9, 10, **11**, 12 … 16 (Java 8 and 11 are **LTS** versions)
- Started under Sun Microsystems, and now under Oracle (since 2009).
- Popularity of Java

  - **Platform Independent**, Object Oriented, Fast, Secure (OS >> JVM >> Programs)

JDK, JRE and JVM



Platform Independence in Java



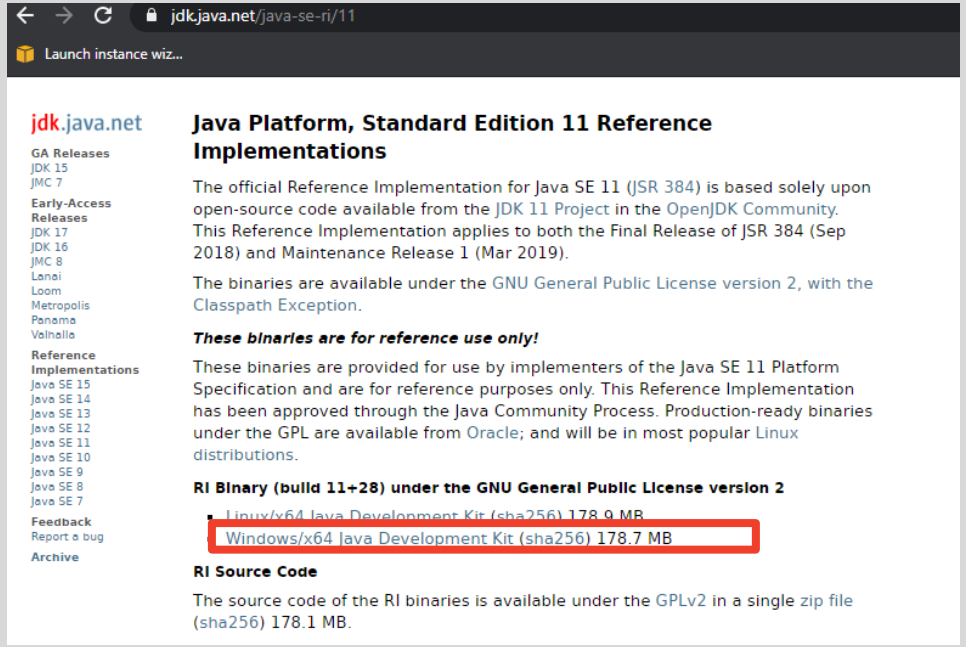
How is Java different from C++

Both are object oriented but there are some major differences between the two.

Aspect	**C++**	**Java**
Security	Lower (usage of pointers)	Higher (no pointers, JVM is used)
Type of Programming	System	Application
Platform	Dependent (only compiler)	Independent (compiler + interpreter)
Supports goto, operator overloading	Yes (program becomes difficult to follow)	No (keeps things simple, yet effective)
Parameter passing	Call by Value & Call by Reference	Only Call by Value
Thread Support	No built in support	Built in thread support

Enough theory; Lets **GET** JAVA…

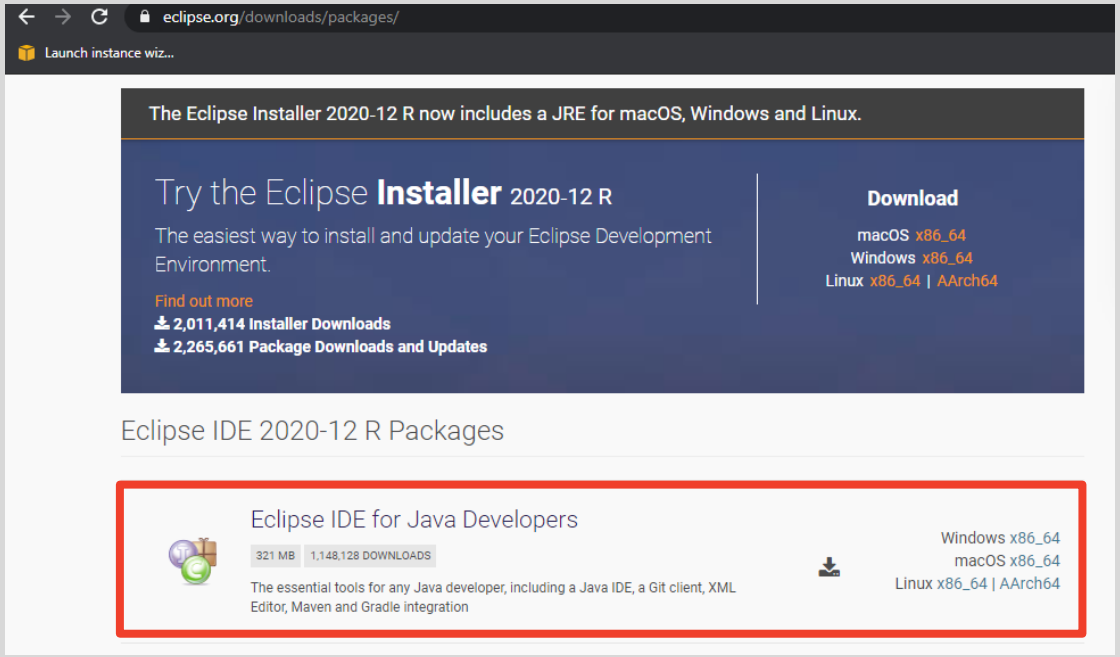
- **Step 1** – Install Java; Setup JAVA_HOME, Validate installation



- Go to page - https://jdk.java.net/java-se-ri/11
- Download the zip file – (highlighted in left image)
- Unzip the downloaded file. It will result in a subfolder folder called jdk-11
- Cut this folder and paste it under "C:\Program Files\Java". If Java folder does not exist, please create first before pasting.
- Go to environment variables and add a new **system environment variable**
  - Name: JAVA_HOME
  - Value: C:\Program Files\Java\jdk-11
- Then look for the **PATH variable** and **edit** it.
- Add an entry there. Value - %JAVA_HOME%\bin
- That's it, you have successfully setup and configured java

And the IDE - ECLIPSE

- **Step 2** - Setting up the IDE (**what is a IDE**)



- Go to page - https://www.eclipse.org/downloads/packages/
- Choose **Eclipse for Java Developers**, and select the **Windows option**. Be careful about these.
- Go ahead with next steps and download the file.
- Once downloaded, run the installer.
- During installation, system may prompt you to select the JDK 11+ version. Please look into the dropdown and select JDK 11 (you have setup this already in prior step – previous slide)
- Once installed open eclipse.
- That's it, you have successfully setup eclipse as well.

Let's **SET** the workspace

- ❑ Creating a Java Project
- ❑ Validating the JRE/JDK used in the project
- ❑ Check for any errors
- ❑ Setup the GIT project to get latest updates from what we learn in sessions
  - ✓ Do not modify anything in this Git project. It is only for your reference

Lets **GO** – Our first program

```java
public class MyFirstJavaProgram {

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

How to compile
How to run >> Console

What do the following terms mean in the example

```
public
class
static
void
main
String
[]
args
System.out.println
Hello World!
```

Why name of file is same as name of class

Using jshell (since Java 9)

```
jshell> String name = "Manish";
name ==> "Manish"

jshell> int age = 35;
age ==> 35

jshell> System.out.println("Hello World");
Hello World

jshell> System.out.println(name);
Manish

jshell> System.out.println("Name is " + name);
Name is Manish

jshell> System.out.println("Name is " + name + " and age is " + age);
Name is Manish and age is 35
```

Ways of printing information to console

```java
public class ConsoleDataPrinter {
    public static void main(String[] args) {

        // Printing text
        System.out.println("This line prints this text and includes a new line at the end");
        System.out.print("Only this text printed, but no new line at end.**");
        System.out.print("##This line continues from last line. No line break.");
        System.out.println("\n"); // This line just prints a new line nothing else

        // Printing variables - formatting string + printing in one shot
        System.out.printf("10 * 3 = %d", 10 * 3).println();
        System.out.printf("%d / %d = %d", 10, 2, 10 / 2).println();

        // Separating concerns - printing and formatting
        System.out.println(String.format("20 * 3 = %d", 20 * 3));
        System.out.println(String.format("%d / %d = %d", 20, 2, 20 / 2));
    }
}
```

Can you print 7 table using such print statements

`System.out` is common across all as it points to the console.

`System.out.print` and `System.out.println` are most common.

Formatting – `String.format()`

Formatting and Printing in one shot – `System.out.printf()`

Printing Variables – what are “variables”

```
public class VariableDemo {
    public static void main(String[] args) {

        int age = 30;
        char gender = 'M';
        boolean isMarried = false;

        System.out.println(String.format("Peter's is %d years old. \nHis gender: %c.", age, gender));
        System.out.println(String.format("'Peter is married' - This statement is %b", isMarried));
    }
}
```

Which are the variables here
What is happening behind the scenes

What are int, char, boolean

Variables – naming conventions

```
public class VariableDemo {
    public static void main(String[] args) {

        int age = 30;
        char gender = 'M';
        boolean ISMARRIED = false; // unnecessarily prominent.
        long nooffriends = 5; // less readable.

        boolean isMarried = false; // more readable. Follows camel casing
        long noOfFriends = 5; // more readable. Follows camel casing

        System.out.println(String.format("Peter's is %d years old. \nHis gender: %c.", age, gender));
        System.out.println(String.format("'Peter is married' - This statement is %b", isMarried));
        System.out.println(String.format("He has %d friends", noOfFriends));
    }
}
```

- Characters, Numbers, $ and _
- Cannot start with a number
- No length restriction
- Naming should follow camel casing
- Cannot be a **keyword**

Best Practice – Overall Naming Conventions

```java
public class P {
    int i;
    String n;
    int a;
    boolean m;

    int calcYOB() {
        return 2021 - a;
    }

    void printYOB() {
        int y = calcYOB();
        System.out.println(String.format("%s - %d", n, y));
    }

    public static void main(String[] args) {
        P p = new P();
        p.n = "Samuel";
        p.a = 25;
        p.printYOB();
    }
}
```

Are you able to follow this code

Do you think any developer will ever be able to guess what a class "P" is used for

What can we do to make this more readable.

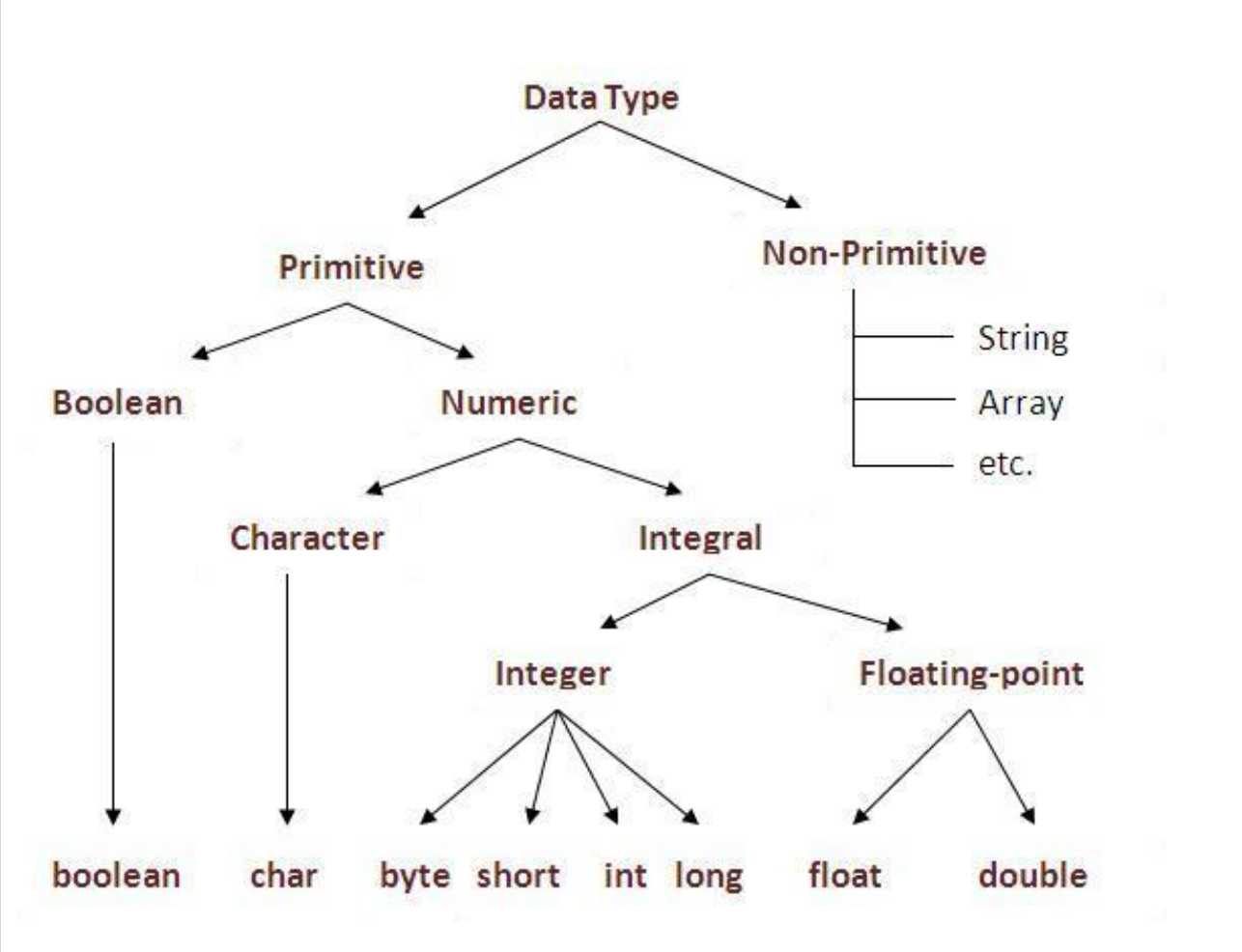
Naming conventions –

- Variables - Camel Case – nouns
- Functions – Camel Case - verbs
- Classes – Pascal Case – nouns

Can you refactor this back to the expected form

Eclipse Trick: Use `Alt + Shift + R` as shortcut to refactor

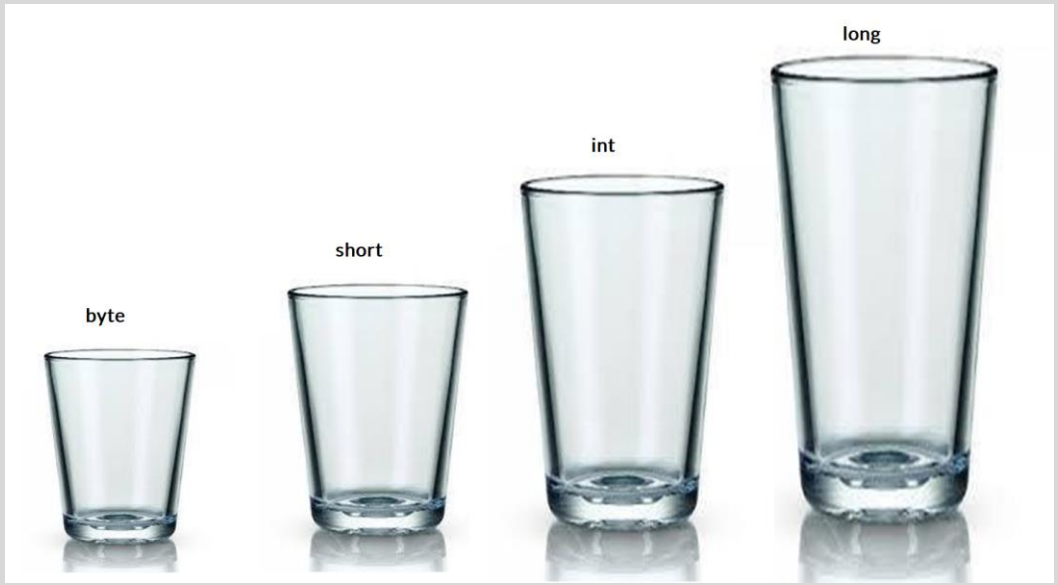
Data Types



Type	Size	Range/Examples
boolean	-	true/false
char	2 bytes (16 bits)	'A', 'X', '0'
byte	1 byte (8 bits)	-128 to 127
short	2 bytes (16 bits)	-32768 to 32767
int	4 bytes (32 bits)	- 2.14 Billion to 2.14 Billion
long	8 bytes (64 bits)	Huge number
float	4 bytes (32 bits)	Not precise
double	8 bytes (64 bits)	Not precise

be **c**areful; **b**aboons **s**houldn't **i**nfuriate **l**arge **f**lying **d**ragons

Data Types



Larger glass/cup can easily hold what smaller glass/spoon can. Hence, implicit conversion is allowed.

```
int i = 1000;
long l = i; // allowed
float f = i; // allowed
double d = l; // allowed
```

Smaller glass may not be able to hold what larger glass is holding. Hence, if you are moving from larger to smaller glass, then you need to explicitly say that it can actually hold it.

```
long l = 1000;
int i = (int) l; // not allowed without the explicit type casting
```

What are methods

```java
public void printHelloWorld() {
    System.out.println("Hello, World");
}

public void printMyName(String name) {
    System.out.println(name);
}

public void printSumOfTwoNumbers(int firstNumber, int secondNumber) {
    System.out.println(
            String.format("Sum of %d and %d is %d", firstNumber, secondNumber, firstNumber + secondNumber));
}

public int calculateSumOfTwoNumbers(int firstNumber, int secondNumber) {
    int sum = firstNumber + secondNumber;
    return sum;
}
```

Methods “perform” some work.

What are parameters and what are arguments.

What is a return type

Let's connect the dots

```
public double calculateAverageOfTwoNumbers(int firstNumber, int secondNumber) {
    double average = (firstNumber + secondNumber)/2;
    return average;
}
```

- `Methods` perform work – hence their names will look like verbs
- `Variables` store values – hence their names will look like nouns
- Every variable has a `data type` – which defines which type of data it will store
- Functions/Methods may or may not return anything. If they do not return anything, the return type is `void`.
- If functions/methods `return a value` (and they can only return one value), then that has to be designated by a data type as well.
- When we call a method, we pass `parameters`.
- Parameters passed into method call are received by method as `arguments`

Our first program with dynamic entry

```java
import java.util.Scanner;

public class MyFirstJavaProgram {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name : ");
        String name = scanner.nextLine();
        scanner.close();
        System.out.println("\nHello, " + name);
    }
}
```

What do the following terms mean in the example

`System.in`

`scanner.nextLine() >> This is a method`

What does the + in below line do

```java
System.out.println("Hello, " + name);
```

That's all for now. Please practice below

1. Write your first program, and print your own favorite line.
2. WAP (Write a Program) to do the following:
    a. Accept your name and print your name with a welcome message
    b. Accept a number and print the square and cube of the number
    c. Take 2 numbers and write methods to do the following:
        i. Return sum of the two numbers
        ii. Return the difference between the two numbers (use `Math.abs()`)
        iii. Return first number raised to the power of second number (use `Math.pow()`)
        iv. Print the average of the two numbers.

For each of the cases, use `System.out.println()` with and without `String.format()` to achieve the same result in printing.