JDK, JVM, JRE, JShell

What is a programming language

A program is a set of instructions given to the computer to perform a process. The process can be as simple as adding two numbers to something very complex. Compilers and interpreters take the code which is written in human-writable-readable form and convert it to computer-readable form. In the case of programs that are written in languages which need compilation, there is a compiler which translates the code into bytecode. The target machine directly translates the bytecode into machine language. In this context, target machine is the machine in which the code is executed. In an interpreted language, the source code is directly translated by the target machine. Instead, the interpreter, reads and executes the code sequentially, in most cases. The disadvantage in this case is that it is time consuming; many programming errors are not apparent till the time of execution.

Java, JDK, JRE

Java has been around for more than 2 decades and JDK has constantly evolved all throughout to now be the robust and stable OOP Language used in many huge applications. Java has two components - Java Development Kit (JDK) and Java Runtime Environment(JRE).

- JDK includes all the packages that you need to program in Java. The bin folder in JDK also includes the Java compiler. The compiler is required only if you want to develop the code and compile it into bytecodes.
- JRE includes all the executables you need to run java programs. Most modern computers come with JRE.

Open a terminal and type `java -version`. Without any additional installations, you will be able to run a runnable java program (Not all Java programs are runnable. Some are also meant to be just reusable components; more on this later). Among the many executables in the runtime environment there is `java` which creates a virtual machine that loads the class, creates objects as and when required and also manages memory allocation issues. This virtual machine is called the JVM. You can think of it like the iTunes executable file which creates a virtual music player in your phones or computer. We may not be be mindful of the fact that it is an executable which needs to be run to start the application as it happens implicitly and that actually a virtual machine that is loaded which simulates most features of a real music player. JVM behaves like a virtual computer specifically meant for Java.

For this course we will use JDK 11. Please visit the link and download the compressed version that is meant for your operating system. For Mac it is gz file and for windows it is a zip file.

1. Open the compressed file into a folder.

2. The bin folder inside the jdk folder contains all the executable files you need for Java.

3. To set the java home on your system, open a terminal.

   At the command prompt, type `vscode ~/.bash_profile` open and paste this in the appropriate place.

   `export JAVA_HOME=/Users/lavanyas/Downloads/jdk-11.0.3.jdk/Contents/Home`.

   This command will ensure that the java home is set and you can compile and run the java programs from your terminal.

4. Open a new terminal and type JShell. JShell is an extremely useful utility available from version Java 9 onwards, which allows you learn basic Java syntax, before you actually start writing programs.

5. We will write some code to print our traditional 'Hello World'.

6. Type `System.out.println("Hello World!!")

7. `System.out` represents the standard output of the computer which is most cases is the console. This is the same as `console` in javascript. `System.in` represents the standard input which is the keyboard. We will look at an example using this later.

8. Just like there are modules/packages in javascript, there are packages in Java which provide reusable components and code. The main packages which java has are :

* java.lang - Provides classes that are fundamental to the design of the Java programming language.

* java.util - Contains the collections framework, some internationalization support classes (when you have to provide for multiple language support), a service loader, properties, random number generation, string parsing and scanning classes, base64 encoding and decoding, a bit array, and several miscellaneous utility classes.

* java.io - Provides classes that handle I/O from standard I/O, Files, network, etc.,

* java.math - For all the math related functionalities

* java.net - For all network related functionalities

* java.sql - For all database and sql related functionalities

* java.time - For time and data related functionalities

* and many more such packages with specific functionalities and there are many more open source packages as well.

All of these packages are available in a central repository called **Maven**. We will look at this at a later stage. For now, we will use the default packages that come with JDK. While working on the JShell, you don't have to do anything special to use the packages that come with JDK.

9. In Java, all the statements have to end with a semicolon. ';' marks the end of the statement. JShell will execute single line commands like the one in step 6, without a semicolon. But that will not compile inside a Java program. However, in Java you can give multiple statements on the same line, as long as they

are separated with a semi-colon. Try executing this on JShell.

```
System.out.println("Hello World!");System.out.println("next line");
```

10. In Java, variables have to be declared before they are used. Java variables are type specific. Meaning, when you declare a variable, you have to mention the type of variable, the reference name and the initial value. The format for declaring variable is <type> <refname> = <initial value>. Run

```
int i=0;
```

This means we are creating a variable of type int which will be referred by the name i whose initial value will be 0.

11. Let's now output the value of i.

```
System.out.println(i)
```

This will print out 0.

12. In a Java program, the variables are statically-typed. You cannot change the type of a variable once you create it as long as the variable exists. However, JShell let's you do that and the type of the variable is what you defined last. Try the following in JShell.

```
String i="New i String"

System.out.println(i)
```

13. A block of code in Java is written within parenthesis as in {//block of code} just like in Javascript. In JShell, if you want to execute a set of statements together, you will enclose them in {}. Try executing the following.

```
 {

 int i=0;

System.out.println(i);

 i=1;

System.out.println(i);

}
```

Note: that in the third line we didn't have to mention the type again, as the variable is already declared.

To do: Change the third line to `String i="hello"` and see what happens.

Few things about JShell which you might find useful

* /vars gets the list of all variable in the shell environment at the given point of time.

* /history lists all the commands including the JShell and Java commands that have been run in the session.

* /list lists the Java commands which are currently valid.

* You can quote the line number and run the command again preceding the line number with a '/'. For example to run a statement in line number 5 of the list, type `/5` and press enter.

14. There are many primitive data types in Java. These are:

    * byte (-128 to 127)

    * short (-32,768 to 32,767)

    * int (-2,147,483,648 to 2,147,483,647)

    * long (-9223372036854775808 to 9223372036854775807)

    * float (int with decimal portion)

    * double (long with decimal portion)

    * char (16 bits)

    * boolean (1 bit)

They are called primitive because they are primitive and all systems know how to treat these data types. Anything other than these, will be built based on these. For instance, String is an array of char type. Don't get hassled as Java provides ways to use these. They provide wrapper classes for all the primitive data types. Before we know what the difference is, let's explore the primitive data types on JShell.

Try the following assignments.

```
byte b =127

System.out.println(b)

short sh = 32767

System.out.println(sh)

int j = 2147483647

System.out.println(j)

long l = 9223372036854775807L

System.out.println(l)

float f = 9999999

System.out.println(f)
```

```
double d = 9999999.999999999

System.out.println(f)

byte b =128

System.out.println(b)

short sh = 32768

System.out.println(sh)

int j = 2147483648

System.out.println(j)

long l = 9223372036854775808

System.out.println(l)

float f = 10000000

System.out.println(f)
```

Casting

Casting is the name given to changing the type of the variable. 'Casting' in movies or TV Series means someone plays a role of a person, different from who they really are. In programming language, a variable of one type is made to play the role of another type.

```
short sh = 5;

int i = (int)sh;

System.out.println(i);
```

15. We learnt the range of values each data type can store. 'short' cannot store the entire range of 'int'. What happens when you do the following?

```
int j = 2147483647;

short sh = (short)j;

System.out.println(sh);
```

Exercise: For the next 15 minutes, try converting from one type to another in all permutation and combination and note down your observations. Discuss why it is important to use the appropriate data type.

Creating a Runnable class

We have many types of components in Java. There are Classes, Interfaces, Enums, Packages, Modules, Lambdas. The most basic of these is a Class. Everything in Java is a class. Literally everything. When you write the simplest application to print Hello World, even that is a class. It is called a Runnable Class. What we did in JShell is just executing statement. We will now write a runnable class. We will use **https://www.tutorialspoint.com/compile_java_online.php (https://www.tutorialspoint.com/compile_java_online.php)**

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

As mentioned earlier, Java language grammar thrives on { } and ;. Every block of code in Java has to be within {} and every statement which is to be executed should end with ';'. We didn't have to do this in the JShell because JShell does it for you implicitly.

Try a few things:

- Try removing the word public in the class.
- Try switching public and static in the main method.
- Try removing the word public in the main method.
- Try removing the word static in the main method.

Things you should know about Java

1. To run a class it has to be a public class
2. You run a runnable class with its classname `java `
3. The class should have a public static void main method which takes a String [] argument.
4. The String [] argument is nothing but the command line arguments you give the class when you run it. You can refer to the string array with any variable name. Commonly used ones as **s** and **args**.
5. A class has to be written in a file with the extension Java.
6. A class is first compiled and then run. Compilation is the process of converting the source Java file into bytecodes, which the JVM understands. A class can be compiled on one system and run on another system - This is called **Platform independence**
7. When you compile a .java file, it produces a .class file with the same name if the compilation is successful. The .class file contains bytecodes (non-human readable form) which can be interpreted by an Operating System which has Java Runtime Environment set up.
8. Most computers in the modern days have Java Runtime Environment.

We will use IntelliJ IDE for our learning and development purpose. Download the same from this **link (https://www.jetbrains.com/idea/download/#section=mac)** . We will use the community edition.