



## Summary JAVA

### Sessions No 1 And 2

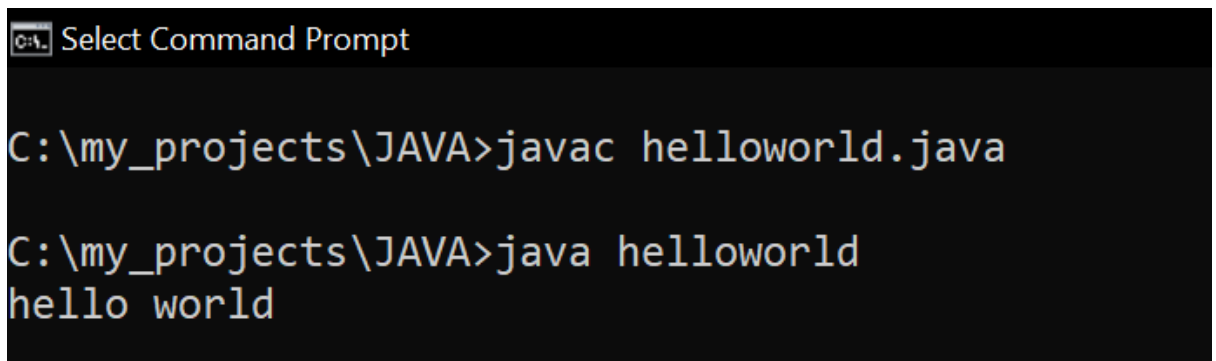
- **Java** is highly used in banks and Enterprise applications, Java's capability to compile one and run anywhere makes it very useful.
- **LTS** Version - Java came up with various versions, but only a few of them are "*long-term supported*" for example java 17,11, and 8 versions are LTS versions means they are stable versions and have long-term support Available.
- Java is open source and various companies have created their own Java by doing some modifications, Some of the known companies which provide Java are Oracle, IBM, and Amazon. Now as a developer, we need the most stable LTS version of JAVA which is highly tested by the companies, and Amazon **corretto** is one of the stable and tested java versions, we will be using in this Training.
- Program file - A file that understood the instructions
- Memory manager - A manager which manages our data in RAM(memory)
- Algorithm - A sequence of statements that collectively achieves a goal
- Developer - A person who writes the instructions in a programming language.
- The semicolon (;) denotes the end of a statement.
- **JDK** - JDK(Java Development Kit) is like SDK(Software development kit), It is a cross-platformed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications.
- Download Amazon Java from here -  
<https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/downloads-list.html>
- For windows directly download from here -  
<https://corretto.aws/downloads/latest/amazon-corretto-17-x64-windows-jdk.msi>
- **High-level languages**-A high-level language is any programming language that enables the development of a program in a much more user-friendly programming, java CPP, python are all high-level languages
- **Low-level languages** - Low-level languages can convert to machine code without a compiler or interpreter, they are written directly in a way which machine understands, it requires memorizing or looking up numerical codes for every instruction, and is extremely difficult to modify
- **JVM** - JVM(Java Virtual Machine) acts as a run-time engine to run Java applications, It understands machine code, A Java virtual machine is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode. JVM is the one, which provides the java facility to compile one and run it anywhere.

- JVM is built on CPP, As CPP has the capability to directly interact with real hardware. Because JVM needs to interact with real hardware to execute our byte code.
- **JAVAC** is the compiler for JAVA which we get from JDK,
- **JRE** - JRE (Java Runtime engine) is the one that runs the byte code in JVM
- Java Executable file(byte code) has the extension of **.class**
- In Java writing the “class” name is compulsory and at the time of compiling, Java will create a “.class” file with the same name of “class” as defined in the code. It means if you have created two classes in a Java programming file then while compiling the code it will create two “.class files”(byte code) with their respective names.

- “Helloworld” java program

```
class HelloWorld          -> Class
{
    public static void main(String [] args)  -> main function (entrypoint)
    {
        System.out.println("hello world"); -> It will print the string in the Console
    }
}
```

- To run a java program, First of all, we will compile the code with **javac** Compiler and then we will execute the compiled code with “**java**” command.



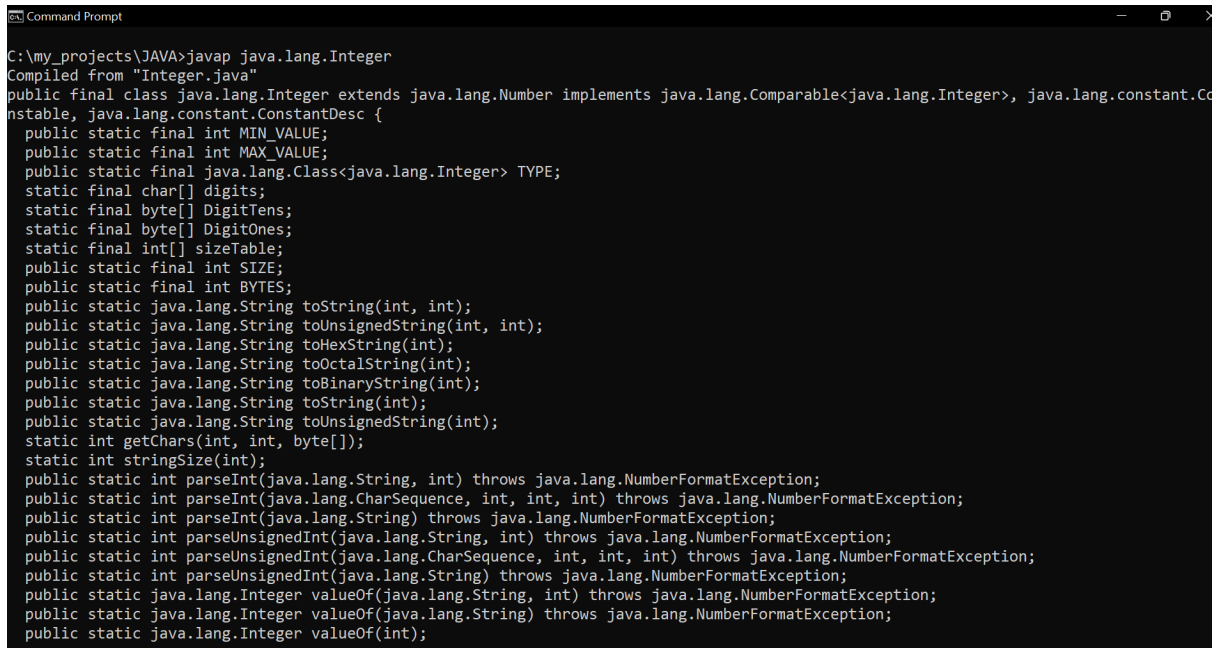
The screenshot shows a Windows Command Prompt window with the title "Select Command Prompt". The prompt is at the directory "C:\my\_projects\JAVA". The user enters the command "javac helloworld.java" to compile the code. Then, the user enters "java helloworld" to run the program, and the output "hello world" is displayed on the next line.

```
C:\my_projects\JAVA>javac helloworld.java

C:\my_projects\JAVA>java helloworld
hello world
```

- The **byte code** of java can be run in any CPU architecture, that’s why it is called as compile one and run anywhere.
- The binary notation of the number 65 and the letter “A” is **01000001**, Now as both of them have the same binary notation, So at the time of storing this “A” or 65 in RAM we can store it, But while reading this binary data we should know in which form we need to read it (in character or in integer form) and that’s why **data type** plays a vital role, data type tell our program to read this binary notation in char or in integer form.
- Compilers are the ones that convert our code into machine language, so different compilers reserve different bytes of space for the same data types

- In java, **char** datatype takes 2 bytes of space in RAM, which means a total of  $2^{16}$  combinations of characters we can store with this data type.
- The Size reserved by datatype in RAM depends on which JVM we are using.
- **System.out.println(Integer.SIZE);** - it will print the size integer datatype is reserving in the RAM
- **#javap java.lang.Integer** - It will tell us all about the Integer Class, like what function it supports for example it supports MIN\_VALUE, MAX\_VALUE, and many more.



```

C:\my_projects\JAVA>javap java.lang.Integer
Compiled from "Integer.java"
public final class java.lang.Integer extends java.lang.Number implements java.lang.Comparable<java.lang.Integer>, java.lang.constant.Combinable {
    public static final int MIN_VALUE;
    public static final int MAX_VALUE;
    public static final java.lang.Class<java.lang.Integer> TYPE;
    static final char[] digits;
    static final byte[] DigitTens;
    static final byte[] DigitOnes;
    static final int[] sizeTable;
    public static final int SIZE;
    public static final int BYTES;
    public static java.lang.String toString(int, int);
    public static java.lang.String toUnsignedString(int, int);
    public static java.lang.String toHexString(int);
    public static java.lang.String toOctalString(int);
    public static java.lang.String toBinaryString(int);
    public static java.lang.String toString(int);
    public static java.lang.String toUnsignedString(int);
    static int getChars(int, int, byte[]);
    static int stringSize(int);
    public static int parseInt(java.lang.String, int) throws java.lang.NumberFormatException;
    public static int parseInt(java.lang.CharSequence, int, int, int) throws java.lang.NumberFormatException;
    public static int parseInt(java.lang.String) throws java.lang.NumberFormatException;
    public static int parseUnsignedInt(java.lang.String, int) throws java.lang.NumberFormatException;
    public static int parseUnsignedInt(java.lang.CharSequence, int, int, int) throws java.lang.NumberFormatException;
    public static int parseUnsignedInt(java.lang.String) throws java.lang.NumberFormatException;
    public static java.lang.Integer valueOf(java.lang.String, int) throws java.lang.NumberFormatException;
    public static java.lang.Integer valueOf(java.lang.String) throws java.lang.NumberFormatException;
    public static java.lang.Integer valueOf(int);
}

```

- Download IntelliJ ID from here, ( After Download, Install it with all the default options)

<https://www.jetbrains.com/idea/download/#section=windows>