Week-1: Introduction to Java

- Java is a general purpose, simple, high level, platform independent, purely object-oriented programming language that combines a well-designed language with dominant features which makes Java Robust and secure language.
- In addition to the core language components, Java software distributions include many software libraries for graphical user interface (GUI), database, network and programming. In this session, we will emphasis on basic features of Java language and got familiar with JVM, JDK, JRE .
- Java is a purely object-oriented programming language. The main objective of this is that in order to write a Java programs we have to use object-oriented concept like **class and object etc.**

Brief History of Java

- Java programming Language was written by **James Gosling** along with two other team members (team named as Green Team) at Sun Microsystems (US) in 1991.
- This language was initially designed for small electronics devices and named it **"Green talk"** (.gt was the file extension), after that it was called "Oak" but later it was renamed to **"Java" in 1995** as Oak was a registered trademark of another company.
- Initially, it was an advance model. But it was fit for internet programming. Later, Java technology merged by Netscape.
- Currently, Java is used in web programming, mobile devices, games, embedded device etc
- **In 1990**, Sun Microsystems got a project to develop application software for electronics devices that could be managed by remote control. They named this project as Stealth Project but later it renamed to Green Project.
- **In 1991**, Green Team divide project module and James Gosling took the responsibility to identify the proper programming language for the project and C and C++ was the best choice for the project. But the problem was that these are system dependent language and hence they can't use it in different processors of embedded devices. So, he started developing a new programming language, which was simple and system independent.
- **In 1994**, the team started developing a Java-based Web Browser named HotJava. HotJava was the first Web browser having the potential to execute ***applets***, which are programs that runs dynamically in Internet.
- **In 1995**, Sun Microsystems Inc. formally announced Java and HotJava.
- **On January 23rd 1996**, Sun Microsystems released the first version of Java i.e. JDK 1.0.

Features of Java:

Compiled and Interpreted

- Java combines both these approaches, thus making a two-stage system. Java compiler translates source code into byte code instructions. Byte code, are not machine instructions and therefore Java interpreter generates machine code that can be directly executed by the machine that is running the Java program.

Object-Oriented: It is a true object-oriented language; everything in Java is an object. All programs and data reside within objects and classes. Java comes with an extensive set of classes, arranged in packages. Object model in Java is simple and easy to extend.

Distributed

- It is designed as a distributed language for creating applications on networks. It has the ability to share both data and programs. Java applications can open and access remote objects on Internet. This enables multiple programmers at remote locations to collaborate and work together on a single project.

Platform-independent and portable

- Java programs can be easily moved from one system to another. Changes and upgrades in operating systems, processors will not force any changes in Java programs.
- We can download a Java applet from a remote system onto local via Internet and execute in locally. This makes the Internet an extension of user's basic systems providing unlimited number of accessible applets and applications.
- Java ensures portability in two ways. Java compiler generates byte code instructions that can be implemented on any machine. The size of primitive data types is machine-independent.

Robust and Secure

- Java is a robust language. Robust means strong. It provides many safeguards to ensure reliable code. It has strict compile-time and run-time checking for data types.
- It is designed as garbage collected language relieving programmers virtually all memory management problems. It also incorporates the concept of exception handling.
- Java systems not only verify all the memory access but also ensure that no threat of viruses communicated through an applet. The absence of pointers in Java ensures that programs cannot give access to memory locations without proper authorization.

Simple, Small and Familiar

- Java is a small and simple language. Many features of C & C++ that are redundant are not part of Java.
- Java does not use pointers, header files, goto statement. It eliminates operator overloading and multiple inheritance.
- To make the language look familiar to the existing programmers, it was modeled on C and C++ language.

Multithreaded and Interactive

- A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.
- The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.

.High Performance

- Java performance is impressive due to the use of intermediate byte code. Java architecture is also designed to reduce overheads during runtime. Incorporation of multithreading enhances the overall execution speed of Java programs.

Dynamic and Extensible

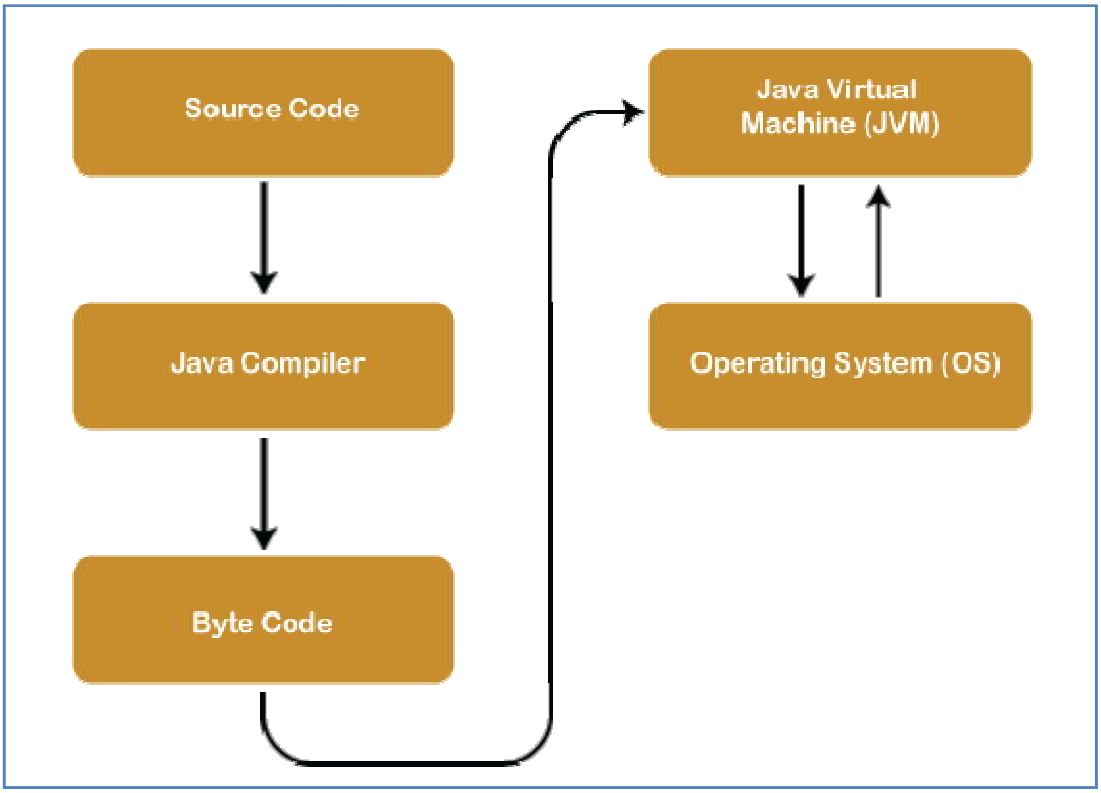
- Java is a dynamic language. It is capable of dynamically linking in new class libraries, methods and objects. Java can determine the type of class through a query.
- Java programs support functions written in other languages such as C and C++. These functions are known as Native methods.

Major Difference C++ and JAVA:

C++	Java
C++ is platform dependent	Java is platform independent
C++supports operator overloading	Java doesn't support operator overloading.
C++ uses compiler only.	Java uses compiler and interpreter both.
C++ supports both call by value and call by reference	Java supports call by value only. There is no Call by reference in Java
C++ supports structures and unions	Java doesn't support structures and unions
C++ supports multiple Inheritances	Java doesn't support multiple inheritances. Itcan be achieved by interfaces in Java.
C++ supports goto statement	Java doesn't support goto statement.
C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support	Java has built in thread support
C++ supports templates	Java does not have template
C++ support destructors, which is automatically invoked when the object is destroyed	Java support automatic garbage collection.It does not support destructors as C++ does
C++ supports virtual keyword so that we can decide whether or not override a function	Java has no virtual keyword. We canoverride all non-static methods by default. In other words, non-static methods are virtual by default.
C++ supports pointers. You can write pointer programin C++	Java supports pointer internally. But you can't write the pointer program in Java. Itmeans Java has restricted pointer support in Java.
C++ write once compile anywhere	Java write once run anywhere (WORA)

Java Architecture:

- **Java Architecture** is a collection of components, i.e., **JVM, JRE,** and **JDK**.
- It integrates the process of interpretation and compilation. It defines all the processes involved in creating a Java program.
- **Java Architecture** explains each and every step of how a program is compiled and executed.
- **Java Architecture** can be explained by using the following steps:
  - There is a process of compilation and interpretation in Java.
  - Java compiler converts the Java code into byte code.
  - After that, the JVM converts the byte code into machine code.
  - The machine code is then executed by the machine.
- The following figure represents the **Java Architecture** in which each step is elaborate graphically.



Components of Java Architecture:

- The Java architecture includes the three main components:
  - Java Virtual Machine (JVM)
  - Java Runtime Environment (JRE)
  - Java Development Kit (JDK)

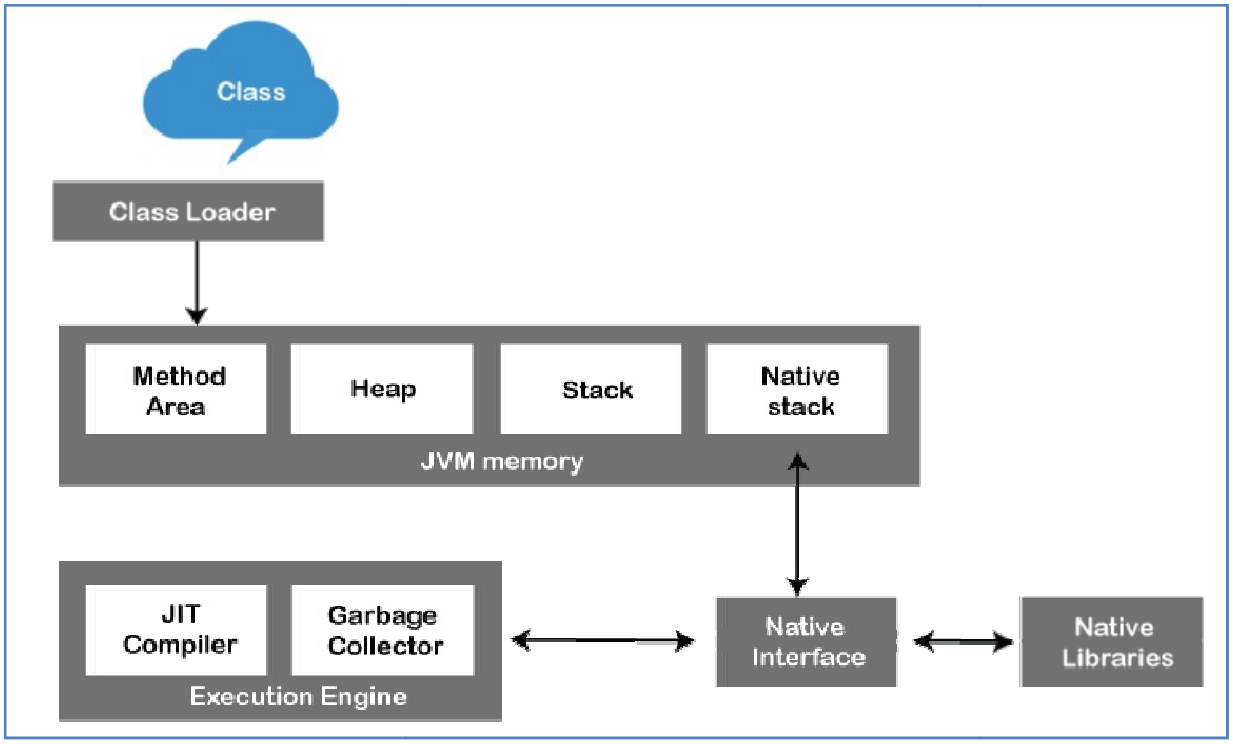
Java Virtual Machine (JVM)

- The main feature of Java is **WORA**. WORA stands for **Write Once Run Anywhere**.
- The feature states that we can write our code once and use it anywhere or on any operating system.
- Our Java program can run any of the platforms only because of the Java Virtual Machine.
- It is a Java platform component that gives us an environment to execute Java programs.
- JVM's main task is to convert byte code into machine code.

- JVM, first of all, loads the code into memory and verifies it. After that, it executes the code and provides a runtime environment.
- JVM has its own architecture, which is given below:

JVM Architecture

- JVM is an abstract machine that provides the environment in which Java bytecode is executed. The fallowing figure represents the architecture of the JVM.



- **ClassLoader:** ClassLoader is a subsystem used to load class files. ClassLoader first loads the Java code whenever we run it.
- **Class Method Area:** In the memory, there is an area where the class data is stored during the code's execution. Class method area holds the information of static variables, static methods, static blocks, and instance methods.
- **Heap:** The heap area is a part of the JVM memory and is created when the JVM starts up. Its size cannot be static because it increases or decreases during the application run.
- **Stack:** It is also referred to as thread stack. It is created for a single execution thread. The thread uses this area to store the elements like the partial result, local variable, data used for calling method and returns etc.
- **Native Stack:** It contains the information of all the native methods used in our application.
- **Execution Engine:** It is the central part of the JVM. Its main task is to execute the byte code and execute the Java classes. The execution engine has three main components used for executing Java classes.
  - **Interpreter:** It converts the byte code into native code and executes. It sequentially executes the code. The interpreter interprets continuously and even the same method multiple times. This reduces the performance of the system, and to solve this, the JIT compiler is introduced.

- **JIT Compiler:** JIT compiler is introduced to remove the drawback of the interpreter. It increases the speed of execution and improves performance.
- **Garbage Collector:** The garbage collector is used to manage the memory, and it is a program written in Java. It works in two phases, i.e., **Mark** and **Sweep**. Mark is an area where the garbage collector identifies the used and unused chunks of memory. The Sweep removes the identified object from the **Mark**

- **Java Native Interface:** Java Native Interface works as a mediator between Java method calls and native libraries.

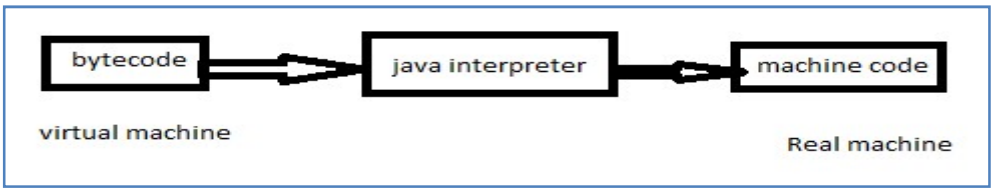
Java Runtime Environment (JRE):

- JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists.
- It contains set of libraries and other files that JVM uses at runtime. The Java runtime environment facilitates the execution of programs developed in Java.

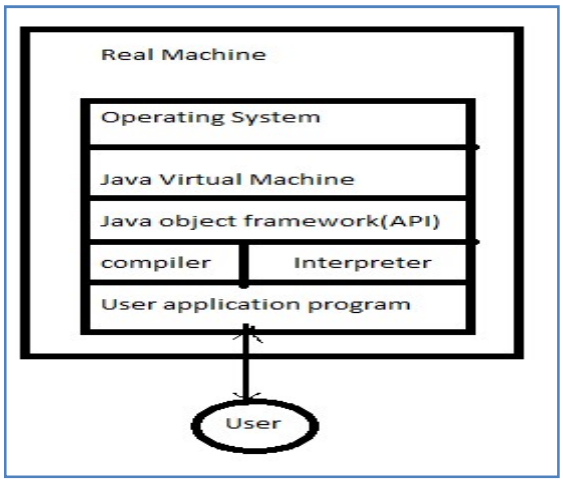
i) JVM(Java Virtual Machine):

- Java compiler produces an intermediate code known as byte code for a machine that doesn't exist. This machine is called Java virtual machine (It exists only inside computer memory).
- It is a computer within a computer and does all major functions of a real computer. That is why Java is portable and platform independent.
  - Process of compilation

  - Process of converting bytecode into the machine code



- The virtual machine code is not machine specific .The machine specific code generated by the Java interpreter by acting as an intermediate between virtual machine and real machine. The below figure illustrates how Java works on a typical computer.



ii) Runtime class libraries: These are set of core class library that are required for the execution of Java program.

iii) User interface toolkits: AWT and swing are example of toolkits that support varied input methods for the users to interact with the application program.

iv) Deployment technologies: JRE consists of

Java plugin: enables the execution of a Java applet on the browser.

Java web start: enables remote deployment of an application.

Java Development Toolkit (JDK).

- JDK comes with a collection of tools that are used for developing and running Java programs. They include:

1. Javac—Java compiler, which translates Java source code to byte code.

2. Java—Java interpreter, which translates byte code to machine code & then executes.

3.Appletviewer—It is used to run Java applets. The applet viewer creates a window in which the applet can be viewed. It provides complete support for all applet functions, including networking and multimedia capabilities.

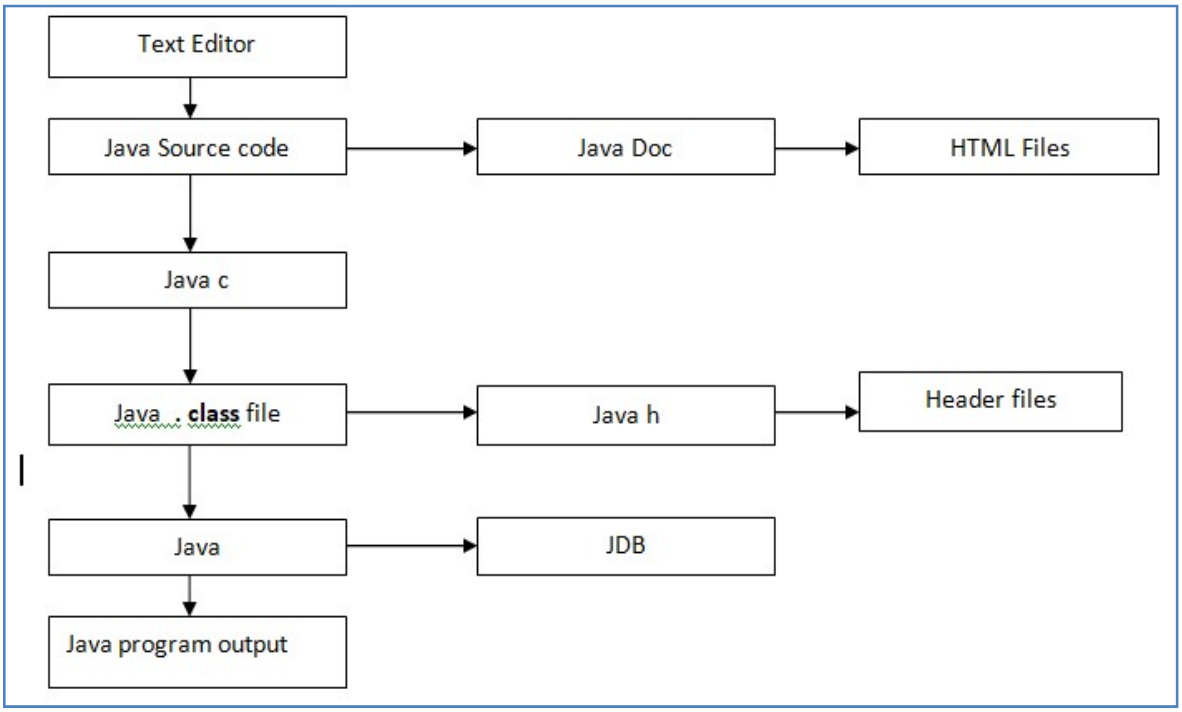
4. JDB—Java debugger, which helps us to find errors in our programs.

5. Javah—Java header file generation tool, used while executing **"Native Methods".** It produces both header and source files. The header file contains C-language definitions that map to Java class definitions. The source files contain the C functions that map to class methods.

6. Javap—Java disassembler, converts byte code to source code. Java debugger is a command-line debugger used to monitor the execution of Java programs in support of debugging and test activities.

7. Javadoc—Java documentation. It creates HTML format documentation from Java source code that document packages and classes, including methods and data fields. A doc-comment block begins with the characters /** and ends like a normal C comment.

- The following figure shows the Process of developing & running Java application Programs.



The following are the steps that illustrate execution process of the application program

- The user creates the Java source code in the text editor say notepad.
- The source code is compiled with **Javac** command.
- **Javadoc tool** can be used to create HTML format documentation from Java source code.
- On compiling the source code a **.class** file gets generated which consists of the byte code.
- The developer may use **Javah tool** for generating the required header files.
- The class file produced by **Javac** can be interpreted using **Java** in order to produce executable file.

Applications of Java Programming Language:

- The expansion of the Java programming language is very wide and it is proved by the statement **3 Billion Devices Runs Java** which is shown during the installation of Java. Java provides a rich and wide range of API that helps programmers to develop applications. Using Java, we can develop different applications for different purposes. We can use Java technology to develop the following applications:

  - Mobile App Development
  - Desktop GUI Applications
  - Web-based Applications
  - Gaming Applications
  - Big Data Technologies
  - Distributed Applications
  - Cloud-based Applications
  - IoT Applications

Mobile App Development

- The Java programming language can be considered as the official language for mobile application development. Most of the android applications build using Java. The most popular android app development IDE **Android Studio** also uses Java for developing android applications. So, if you are already familiar with Java, it will become much easier to develop android applications. The most popular android applications **Spotify** and **Twitter** are developed using Java.

Desktop GUI Applications

- We can also develop a GUI application using Java. Java provides AWT, JavaFX, and Swing for developing the GUI based desktop application. The tools contain the pre-assembled components like list, menu, and button.

Web-based Applications

- It is also used for developing the web-based application because it provides vast support for web development through Servlet, JSP, and Struts. It is the reason that Java is also known as a server-side programming language. Using these technologies, we can develop a variety of applications. The most popular frameworks Spring, Hibernate, Spring Boot, used for developing web-based applications. **LinkedIn, AliExpress, web.archive.org, IRCTC,** etc. are the popular websites that are written using Java programming language.

Game Development

- Java is widely used by game development companies because it has the support of the open-source most powerful 3D engine. The engine provides unparalleled capacity when it comes to the context of the designing of 3D games. The most popular games developed in Java are Minecraft, Mission Impossible III, etc. There are some popular Frameworks and Libraries available for Game Development, like - LibGDX and OpenGL.

Big Data Technology

- As many programming languages are available for Big Data Technology but still Java is the first choice for the same. The tool Hadoop HDFS platform for processing and storing big data applications is written in Java. In big data, Java is widely used in ETL applications such as Apache Camel and Apache Kafka. It is used to extract and transform data, and load in big data environments.

Distributed Applications

- The JINI (Java Intelligent Networking Infrastructure) provides the infrastructure to register and find distributed services based on its specification. It implements a mechanism that is known as JavaSpaces. It supports the distribution, persistence, and migration of objects in a network.

Cloud-Based Applications

- A cloud application is the on-demand availability of IT resources via the internet. The cloud-based application provides the service at a low cost. Java provides the environment to develop cloud-based applications. We can use Java to develop SaaS (Software as a Service), LaaS (Logging as a Service), and PaaS (Platform as a Service). The cloud application widely used to share data between companies or to develop applications remotely.

IoT Application

- IoT is a technology that connects the devices in its network and communicates with them. IoT has found almost in all the small devices such as health gears, smartphones, wearables, smart lighting, TVs, etc. For developing the IoT application there is a lot of programming languages that can be used but Java offers an edge to developers that is unparalleled. IoT programmers gravitate towards Java because of its security, flexibility, and versatility

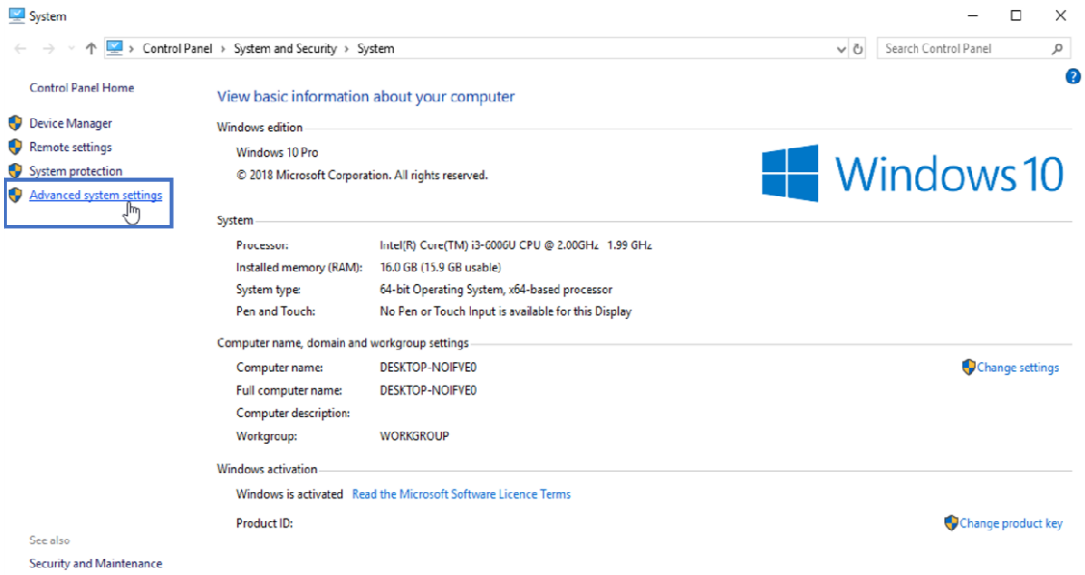
Java environment setup:

In today's world, Java is one of the most popular programming language which is being used by most skilled professionals. However, using it on the command line is not feasible sometimes. Therefore, to overcome this, we can use it on **Eclipse IDE.** Let's see how to setup Java environment on Eclipse IDE.

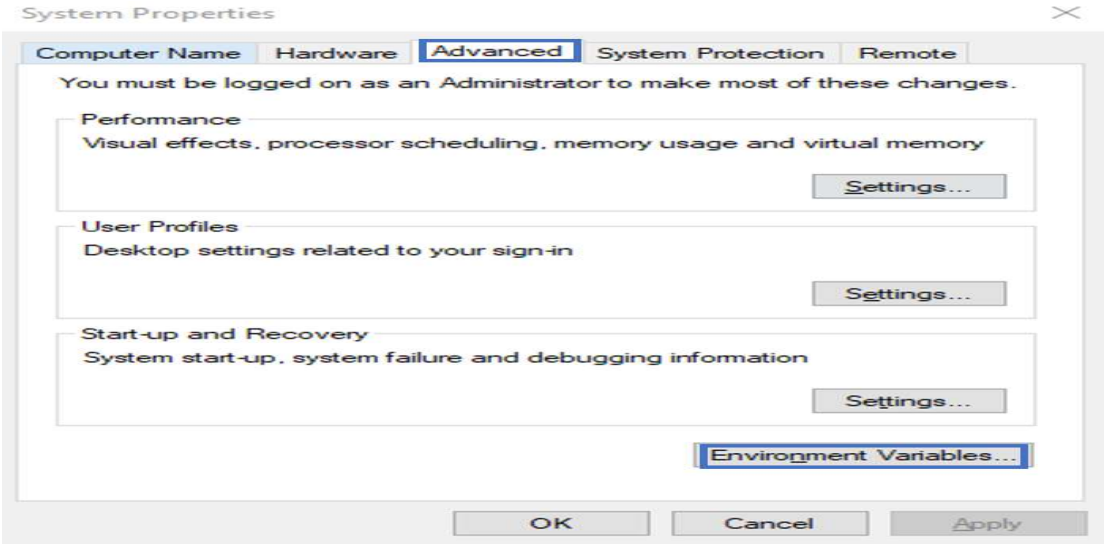
Step 1: Install Java
Step 2: Setup Eclipse IDE on Windows
Step 3: Hello World Program

Install Java:

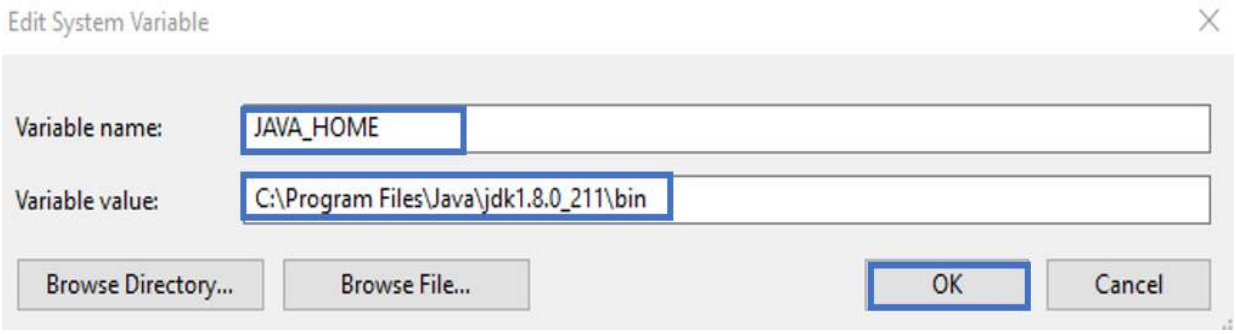
- Follow the below steps to complete your Java installation.
- Go to the Java Downloads Page and click on the option of **Download.**
- Now, once the file is downloaded, run the installer and keep clicking on **Next**, till you finally get a dialog box, which say, you have finished installing.
- Once the installation is over follow the below instructions to set the path of the file.
- Now right click on ThisPC/ My Computer Icon-> Go to its properties and its **Advanced System Settings**. Refer below.



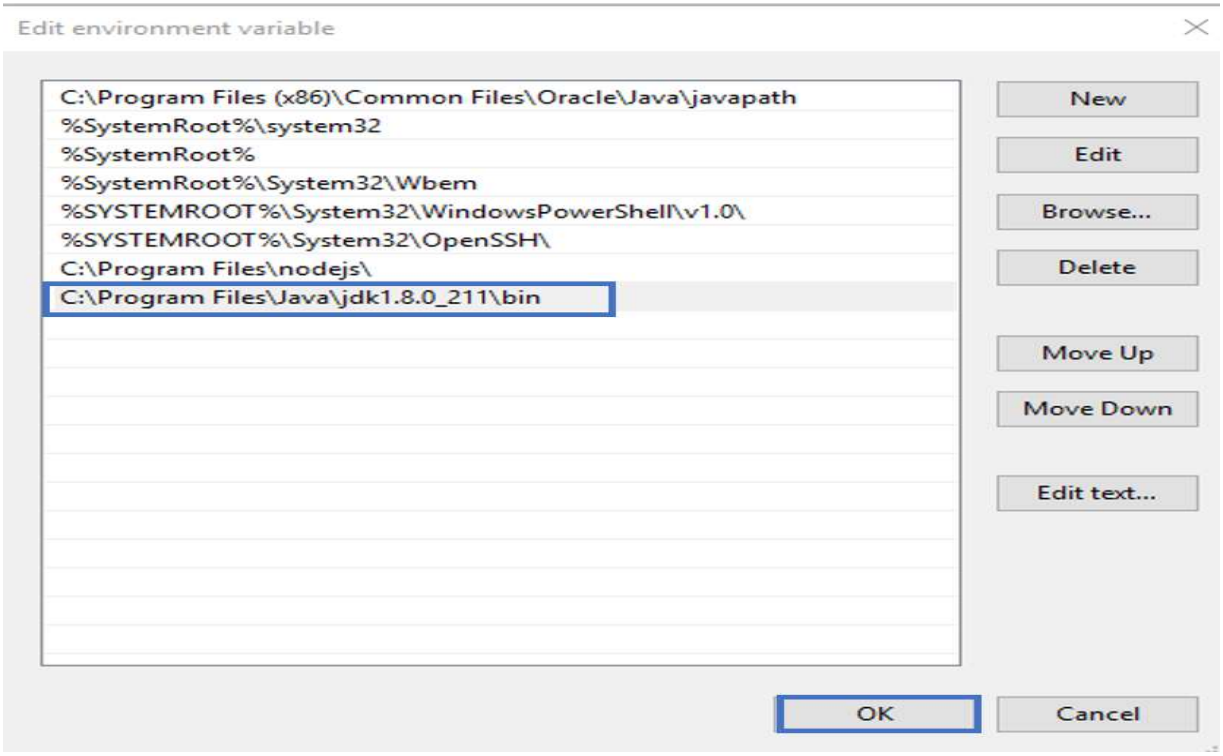
- Now, click on '**Environment Variables**' under '**Advanced**' tab as shown below:



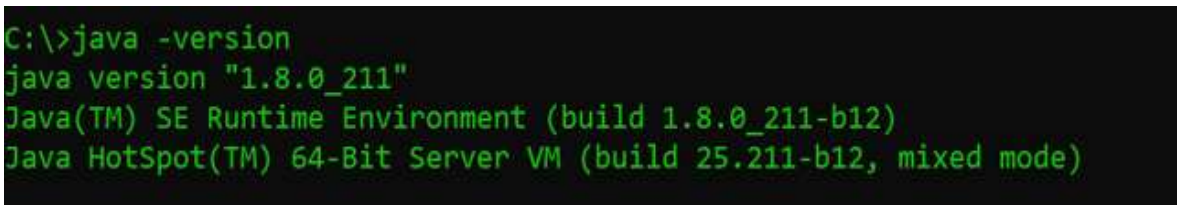
- Next, under **System Variables** choose **New.** Enter the variable name as '**JAVA_HOME**' and the full path to Java installation directory as per your system as shown below:



- Next thing that you have to do is to configure your environment variables. Let's see how to do that. Here, you have to edit the path of the system variable as shown below. Then click OK.



- Now to cross-check the installation, just run following command in cmd – ***java -version***. It should display the installed version of Java in your system.

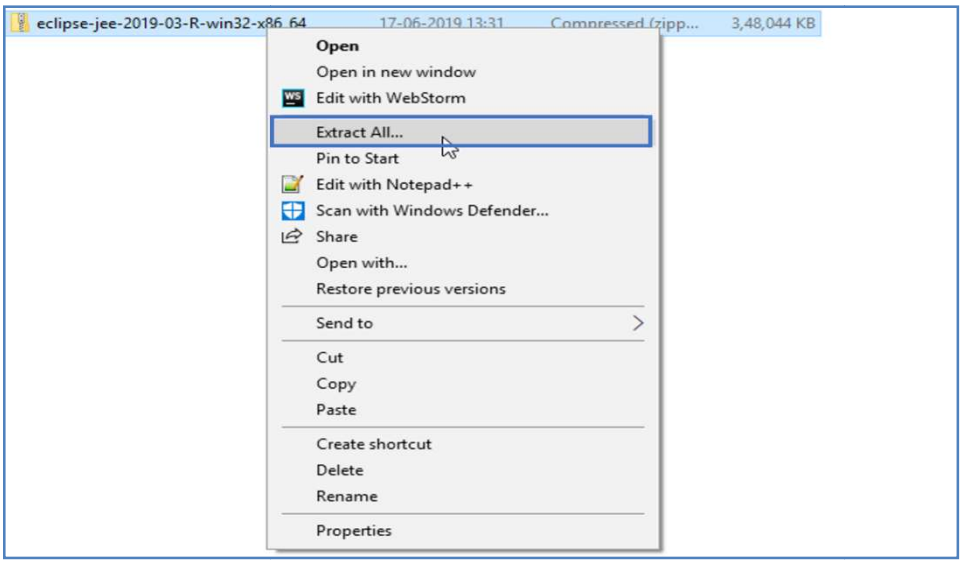


Install Eclipse:

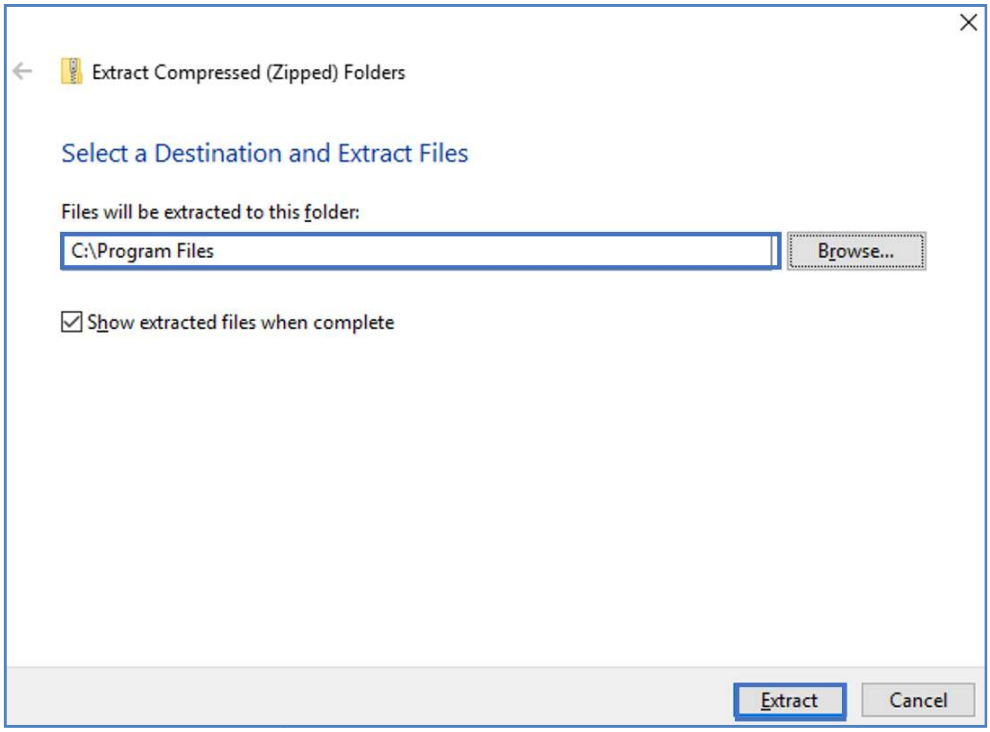
Follow the below steps to configure Eclipse on your system:

Step 1: Navigate to the following URL – https://www.eclipse.org/downloads/packages/ and select the download link depending on your system architecture – (Windows, Mac OS or Linux) and download it.

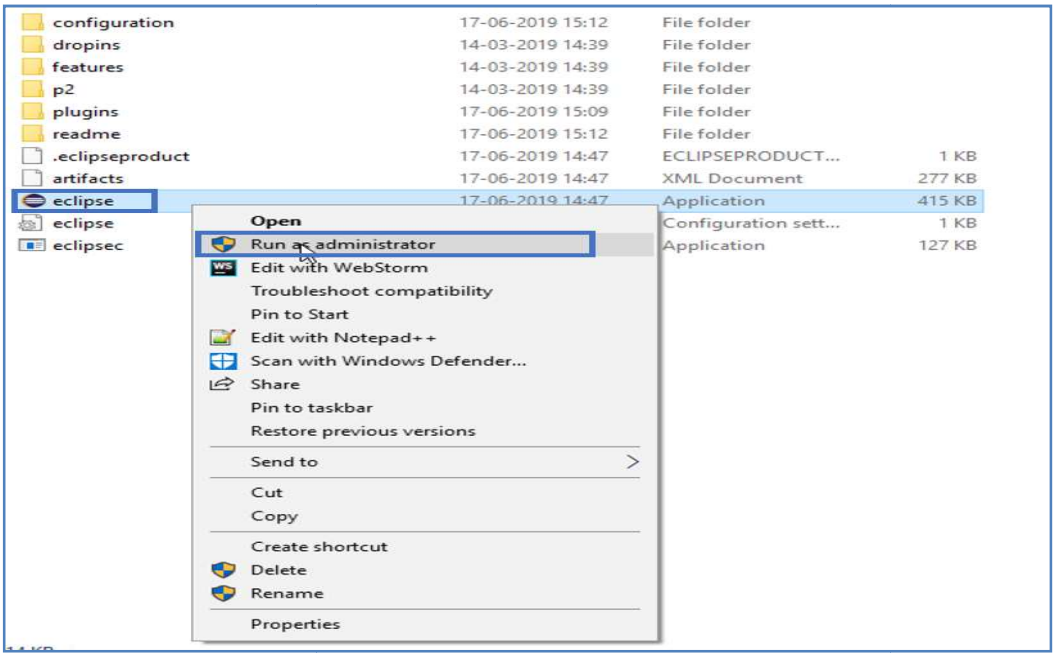
Step 2: Once the download is over, extract the zipped file by right-clicking on the folder and choose **Extract All**. Refer below.



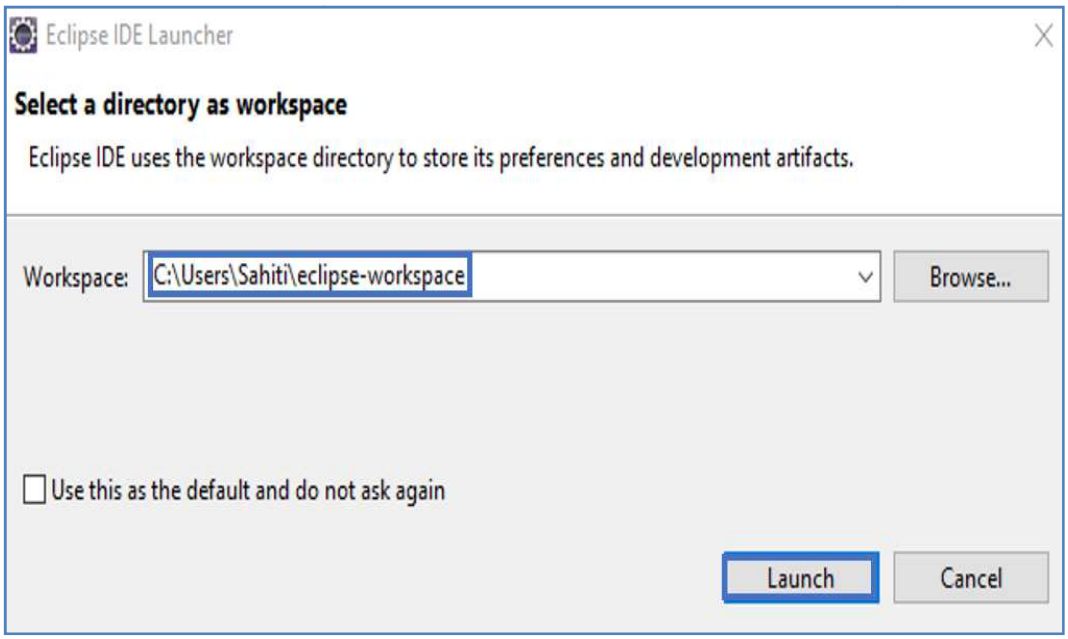
Step 3: You will be then redirected to a dialog box, where you have to choose the directory in which you wish to extract the files. Then click on **Extract**. Refer below.



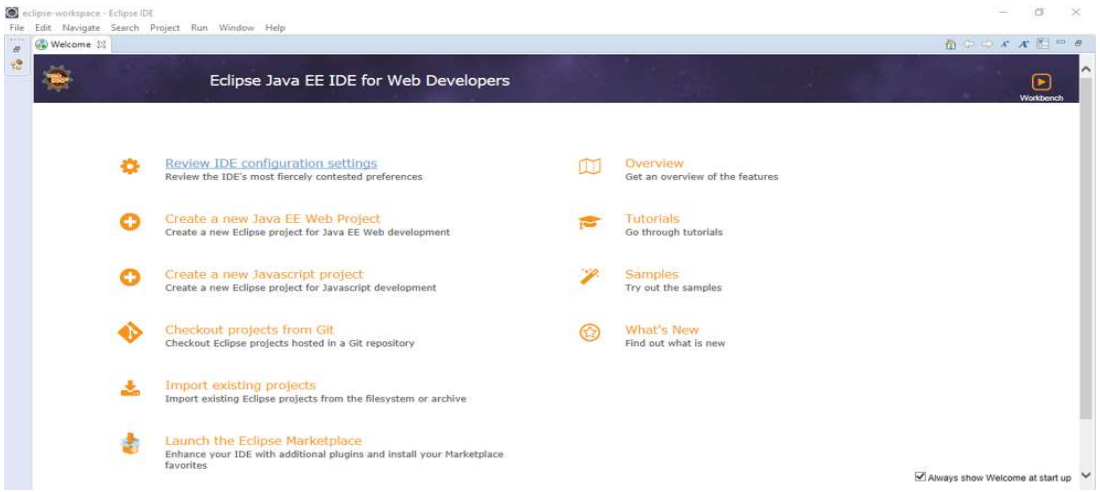
Step 4: After extracting files, open the folder and launch **eclipse.exe.**



Step 5: Then, you have to choose the Launch directory for Eclipse and then click on Launch. Refer below.



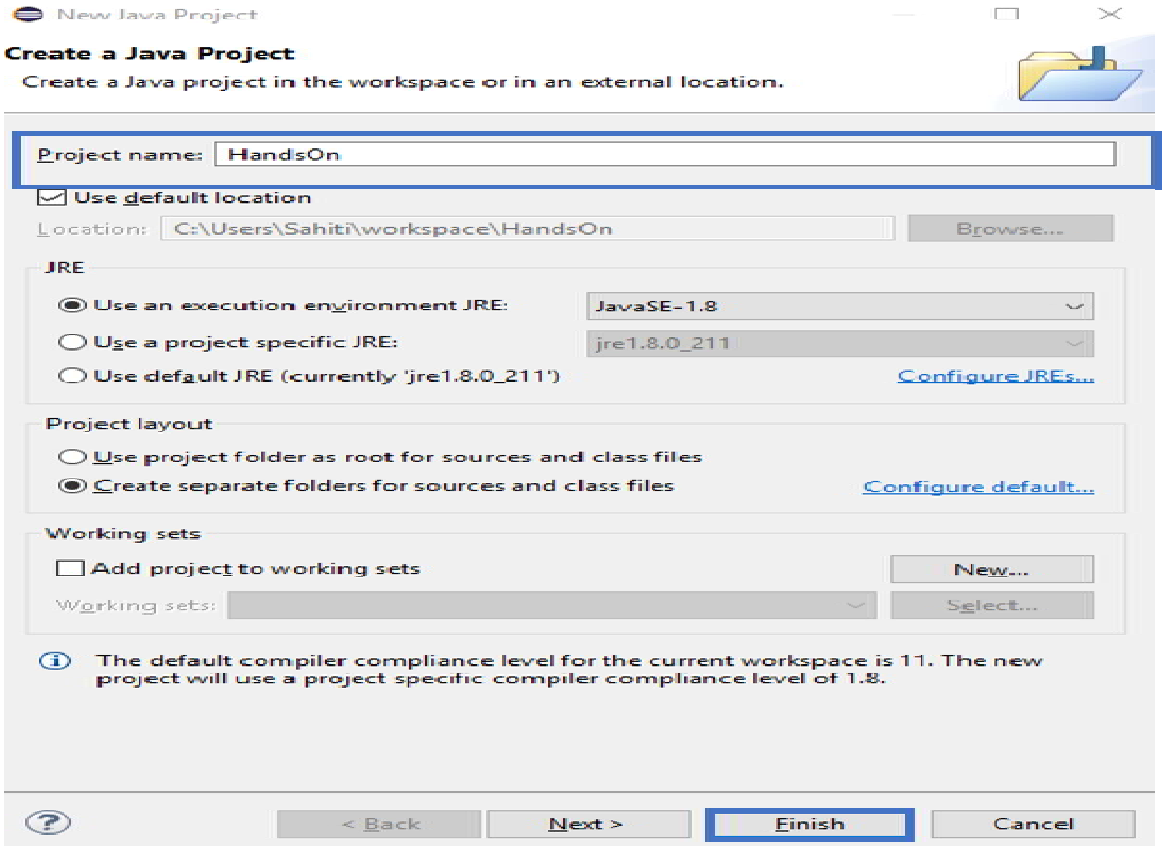
Step 6: Once Eclipse launches, you will see the below window:



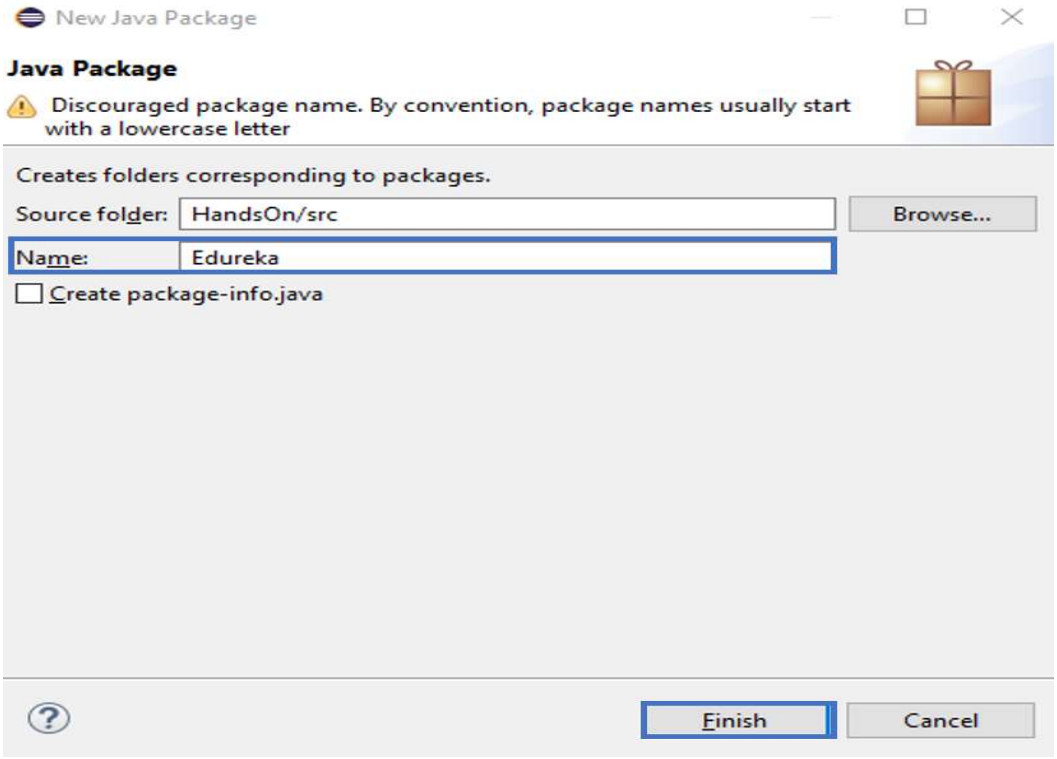
Executing first Java Program : Hello World Program

Step 1: Launch **Eclipse IDE** and go to **File ->New -> Java Project**

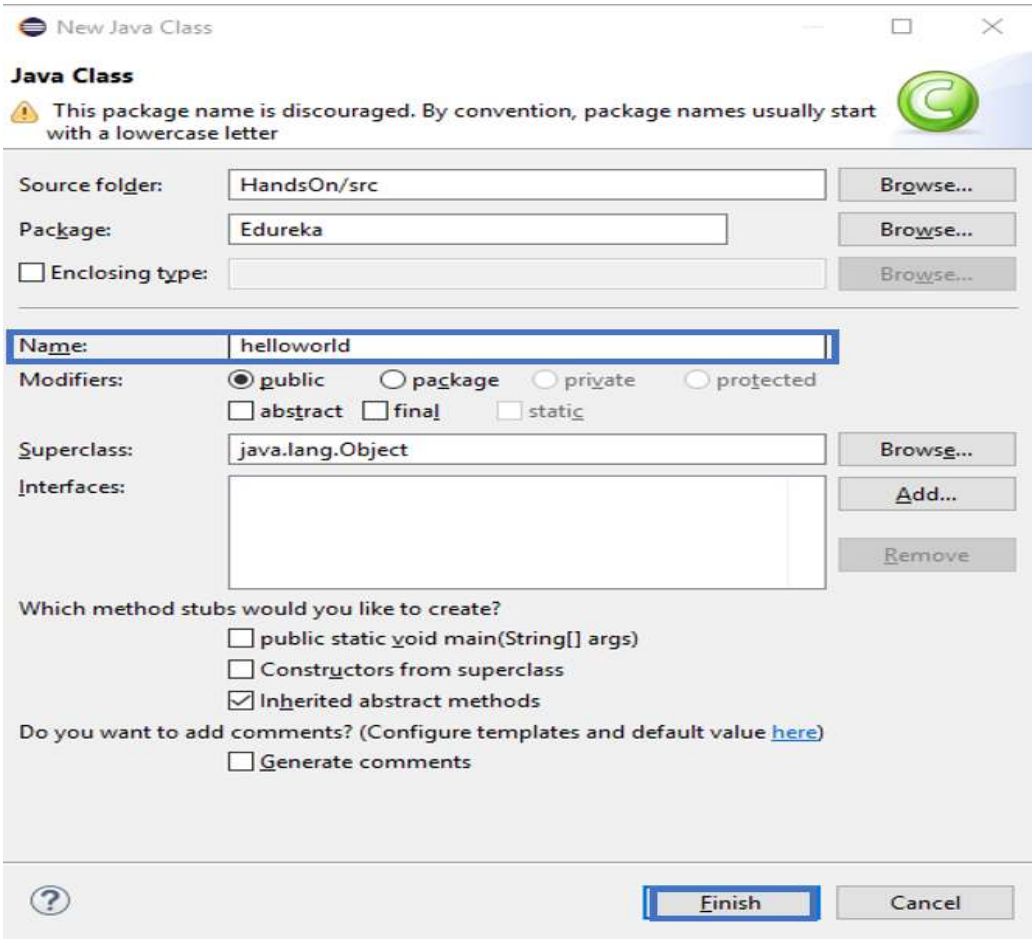
Step 2: Mention the **project name** and click on **Finish.**



Step 3: Now, go to the **Project**, Right-Click on the Project and choose **Package**. In the dialog box, which opens up, mention the **Package name** as below and click on **Finish.**



Step 4: Now, right click on the **Package**, go to **New** and choose **Class**. Mention the **class name** and click on **Finish**. Refer below.



Step 5: Now, mention the following code in the workspace.

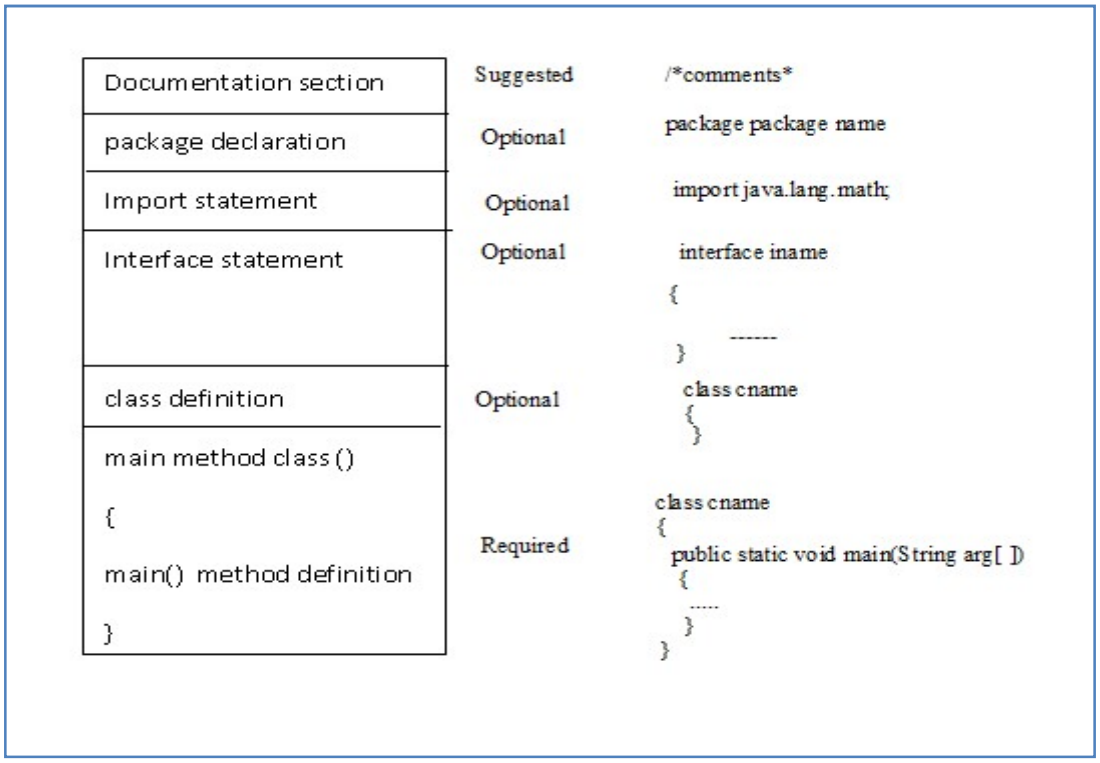
```
package Edureka;
public class helloworld
{
  public static void main(String args[])
        {

                System.out.println("Hello World");
        }
}
```

Step 6: Now, execute your file, by right-clicking on the **helloworld.java file** and choose **Run As -> Java Application.**

Structure of Java Program

- A Java program may contain many classes of which **only one class contain a main() definition**.
- Classes contain data members and methods that operate on the data members. To write a Java program, we first define classes and then put them together . Java program may contain one or more sections as shown below.
- The general structure of Java program is as follows.



1. Documentation section→The documentation section includes a set of comment lines giving the name of the program, the author and other details, which the programmer would like to refer to a later stage. Java supports three types of comments (line, block and documentation).

2. Package Declaration →The first statement allowed in a Java file is a package statement. This statement declares a package name and informs the compiler that the classes defined here belongs to the package. The "package" is a keyword & used to create user defined package.

The general syntax is **package packagename.**

3. Import Statement→ Java environment contains JDK tools & API package. Import statements are used to access built in classes & methods.(similar to #include statements in example: **import Java.lang.*;)**

4. Interface Statement→ An interface is like a class but includes a group of method declarations. This also an optional section and is used only when we wish to implement the multiple inheritance features in the program.

5. Class Definition→ A Java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program. These classes are used to map the objects of real world problems. This includes class name, visibility modes, variables name & methods.

6. main() method class→ The main method creates objects of various classes and establishes communication between them. Java programs requires main () function. The program execution start with main () function.

Implementing a Java Program:

- Implementation of a Java application program involves a series of steps .They are
  1. Creating the program
  2. Compiling the program
  3. Running the program

Step 1: Creating the program

We can create a program using an text editors.(Using notepad).Example:

```
class example
{
  public static void main(String args[])
   {
     System.out.println("C");
     System.out.println("C++");
     System.out.println("JAVA");
   }
}
```

- We must save this program in a file called **example.java** ensuring that the filename contains the class name property. This file is called **the source file.**
- All Java source files will have the extension Java.
- If a program contains multiple classes ,the file name must be the class name of the class containing the main method.

Step 2: Compiling the program

- To compile the program ,we must run the Java compiler(Javac) with the name of the source file
  Ex: **javac example.java**
- On successful compilation ,byte code with an extension .class **example.class** will be created

Step 3: Running the program

- We need to use the Java interpreter to run a standalone program
- Using Java and filename : **java example**
- Now interpreter looks for the main method in the program and begins execution from there.

Output of program will be printed:

C
C++
JAVA

Clean coding in Java:

- Code is clean when it is easily readable by developers other than the original author as well.

Why should we write clean code?

➢ We as developers write code for the computer to understand. At the same time, our code is not going to remain the same as we update the business logic or add new features. Our code might get updated by other developers as well. Hence, we must write code in such a way that other developers can also read and understand it.

How to write clean code?

➢ We can write clean code by following a set of guidelines known as Software Design Principles. Software Design Principles is a set of guidelines proven to work over the years.

A few of the broad guidelines to write clean code are:

- Give meaningful names to variables, functions, classes, and other entities in the code.
- Create functions that are small and do a single thing.
- Encapsulate related data and functions into small independent classes.
- Structure the code for better readability. Keep related code together and keep the lines smaller.
- Enhance readability with proper comments.
- Write readable, fast, independent, and repeatable tests.

Simple Program of Java:

- Java programs are built from classes. A class contains no. of variables (instance variables) & methods(instance methods).

```
class First
  {
      public static void main(String args[])
          {
             System.out.println("Welcome to Java");
          }
  }
```

Output:→ Welcome to Java.

- Every java program starts with a **"class declaration".** A java program can contain any number of classes within it. Out of which one method should contain main () method.
- main () function should be made available to all the classes within the java program. Hence it should be declared as **"public".**
- The keyword **"static"** declares that, the main () function belongs to the entire class & not a part of any object of the class.
- The keyword **"void",** does not return any value. The main () function always accepts string arguments.
- The name of java program is same as name of the class which contains the main() i.e first.java