WEEK1
Introduction to Java, Brief history; features; java architecture; components: JVM, JRE,JDK; Applications; Java environment setup; Structure of java program; Compilation and execution of java program; Clean coding in java.

Introduction:

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs.

Why Use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming language in the world
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has a huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa.

Brief history

- James Gosling, Mike Sheridan, and Patrick Naughton, a team of Sun engineers known as the **Green team** initiated the Java language in 1991.
- In 1995 Java was developed by **James Gosling**, who is known as the Father of Java. Currently, Java is used in mobile devices, internet programming, games, e-business, etc.
- **Sun Microsystems** released its first public implementation in 1996 as **Java 1.0**. It provides no-cost-run-times on popular platforms.
- On November 13, 2006, Sun released much of its Java virtual machine as free, open-source software. On May 8, 2007, Sun finished the process, making all of its JVM's core code available under open-source distribution terms.

Main Features of Java

1. Platform Independent: Compiler converts source code to byte-code and then the JVM executes the byte-code generated by the compiler. This byte-code can run on any platform be it Windows, Linux, macOS which means if we compile a program on Windows, then we can run it on Linux and vice versa. Each operating system has a different JVM, but the output produced by all the OS is the same after the execution of byte-code. That is why we call java a platform-independent language.

2. Object-Oriented Programming Language: Organizing the program in the terms of collection of objects is a way of object-oriented programming, each of which represents an instance of the class.

The four main concepts of Object-Oriented programming are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

3. Simple: Java is one of the simple languages as it does not have complex features like pointers, operator overloading, multiple inheritances, explicit memory allocation.

4. Robust: Java language is robust which means reliable. It is developed in such a way that it puts a lot of effort into checking errors as early as possible that is why the java compiler is able to detect even those errors that are not easy to detect by another programming language. The main features of java that make it robust are garbage collection, Exception Handling, and memory allocation.

5. Secure: In java, we don't have pointers, so we cannot access out-of-bound arrays i.e. it shows **ArrayIndexOutOfBound Exception** if we try to do so. That's why several security flaws like stack corruption or buffer overflow are impossible to exploit in Java.

6. Distributed: We can create distributed applications using the java programming language. Remote Method Invocation and Enterprise Java Beans are used for creating distributed applications in java. The java programs can be easily distributed on one or more systems that are connected to each other through an internet connection.

7. Multithreading: Java supports multithreading. It is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

8. Portable: As we know, java code written on one machine can be run on another machine. The platform-independent feature of java in which its platform-independent byte-code can be taken to any platform for execution makes java portable.

9. High Performance: Java architecture is defined in such a way that it reduces overhead during the runtime and at some time java uses Just In Time (JIT) compiler where the compiler compiles code on-demand basics where it only compiles those methods that are called making applications to execute faster.

10. Dynamic flexibility: Java being completely object-oriented gives us the flexibility to add classes, new methods to existing classes. Java even supports functions written in other languages such as C, C++ which are referred to as native methods.

11. Architecture-neutral: Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

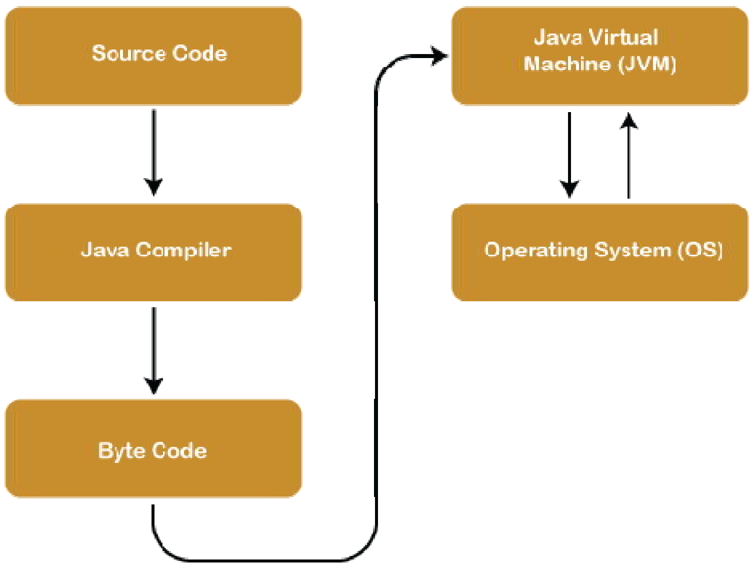
Java Architecture

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

- 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.

Java Runtime Environment

- It provides an environment in which Java programs are executed.
- JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it.

JDK (Java Development Kit)

JDK is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

Applications of Java

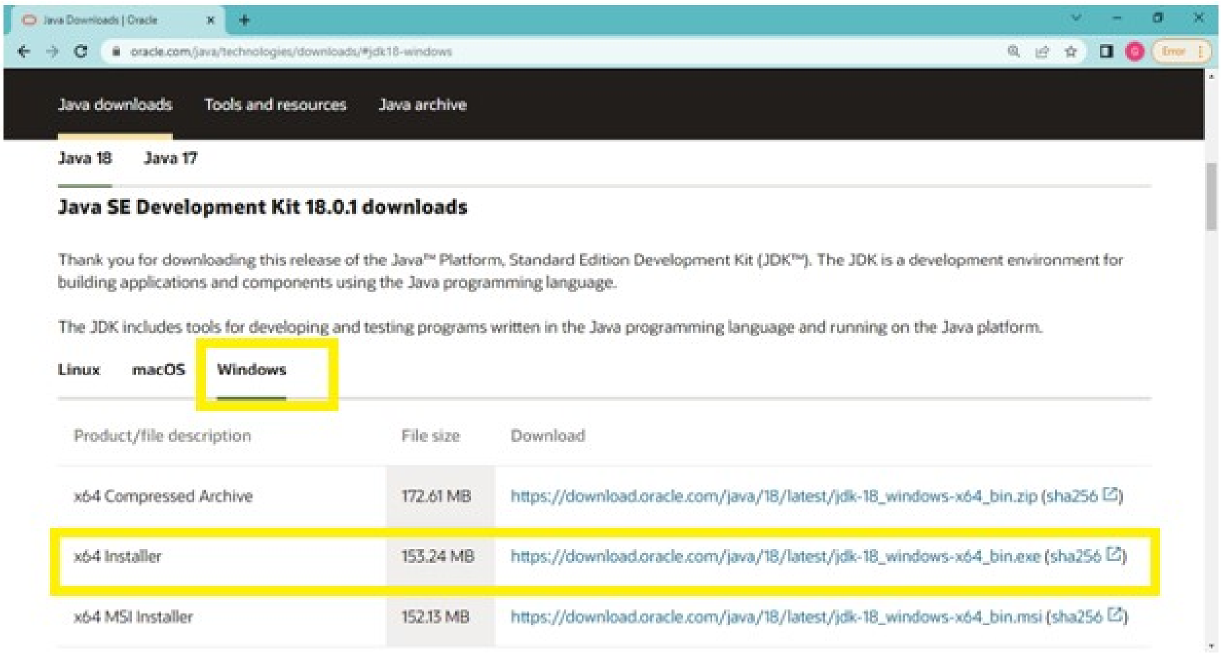
- Desktop GUI Applications
  E.g.: Acrobat Reader, ThinkFree
- Web Applications
  E.g.: Amazon, Broadleaf, Wayfair
- Mobile Applications
  E.g.: Netflix, Tinder, Google Earth, Uber
- Enterprise Applications
  E.g.: Enterprise Resource Planning (ERP) systems, Customer Resource Management (CRM) systems
- Scientific Applications
  E.g.: Mat lab
- Web Servers & Applications Servers
  E.g.: Apache Tomcat, Project Jigsaw, Rimfaxe Web Server (RWS)
- Embedded Systems
  E.g.: SIM cards use Java technology, Blue-ray disc player
- Server Apps in Financial Industry.
  E.g.: Most of the leading financial institutions like Barclays, Citi group, Goldman Sach, etc. use Java-based software tools for their business.

- Software Tools.
  E.g.: IDEs like Eclipse, IntelliJ IDEA, and Net beans are all written and developed in Java.
- Trading Applications.
  E.g.: The popular trading application Murex, which is used in many banks.
- J2ME Apps.
  E.g.: The popular application WhatsApp available on Nokia is available in J2ME.
- Big Data Technologies.
  E.g.: Hadoop, Apache HBase, ElasticSearch, Accumulo

Java environment setup

Steps for setting the environment in Windows operation system are as follows:

Step 1: Java18 JDK is available at "https://www.oracle.com/java/technologies/downloads/#jdk18-windows". Go to the Windows tab and Click the second last link for download as highlighted below.



Java SE Development Kit 18.0.1 downloads

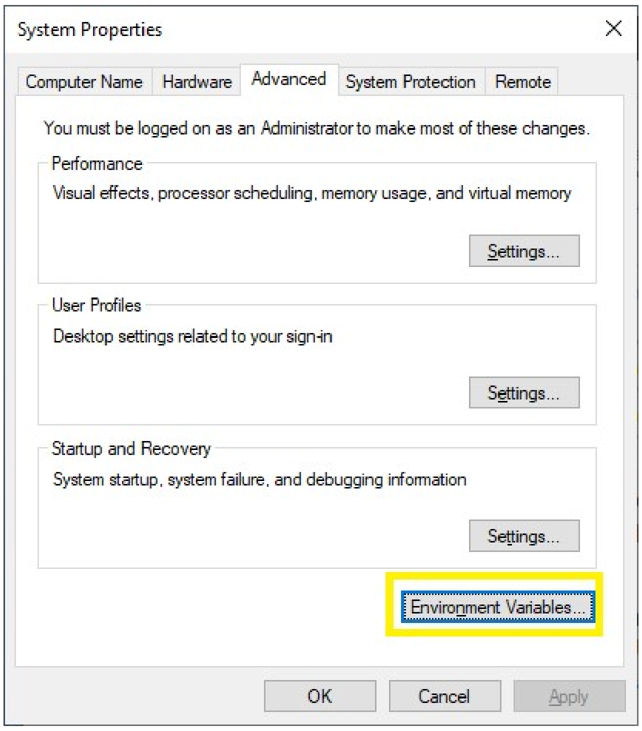
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

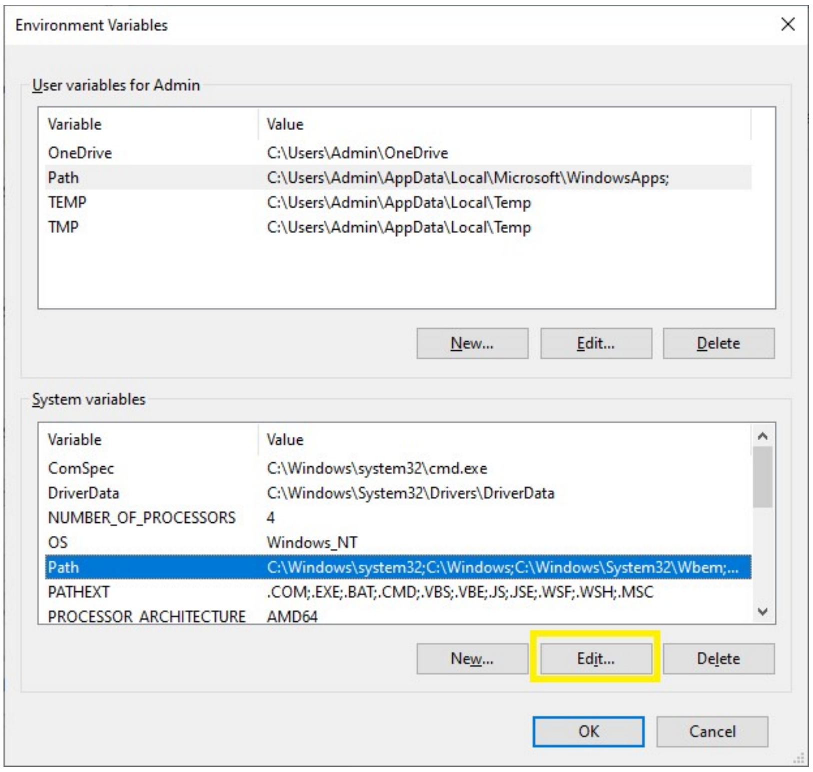
Product/file description	File size	Download
x64 Compressed Archive	172.61 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.zip (sha256)
x64 Installer	153.24 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.exe (sha256)
x64 MSI Installer	152.13 MB	https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.msi (sha256)

Step 2: After download, run the .exe file and follow the instructions to install Java on your machine. Once you installed Java on your machine, you have to set up the environment variable.

Step 3: Go to **Start** and search for **"Environment Variables".** Click on **"Edit the system Environment Variable**s". Under the Advanced System Setting option click on **Environment Variables** as highlighted below.


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

Step 4: Now, you have to alter the “Path” variable under System variables so that it also contains the path to the Java environment. Select the “Path” variable and click on the Edit button as highlighted below.



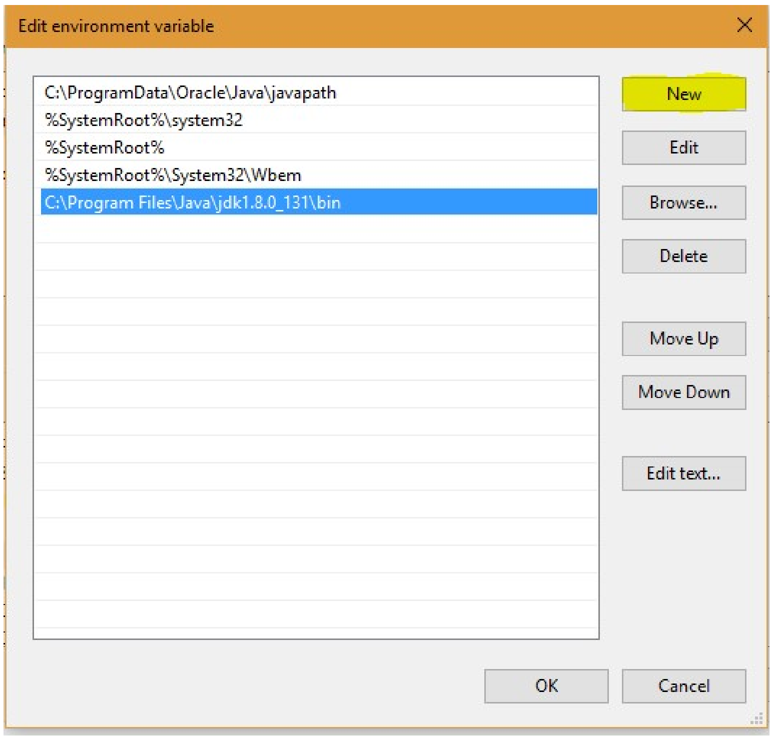
User variables for Admin

Variable	Value
OneDrive	C:\Users\Admin\OneDrive
Path	C:\Users\Admin\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\Admin\AppData\Local\Temp
TMP	C:\Users\Admin\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

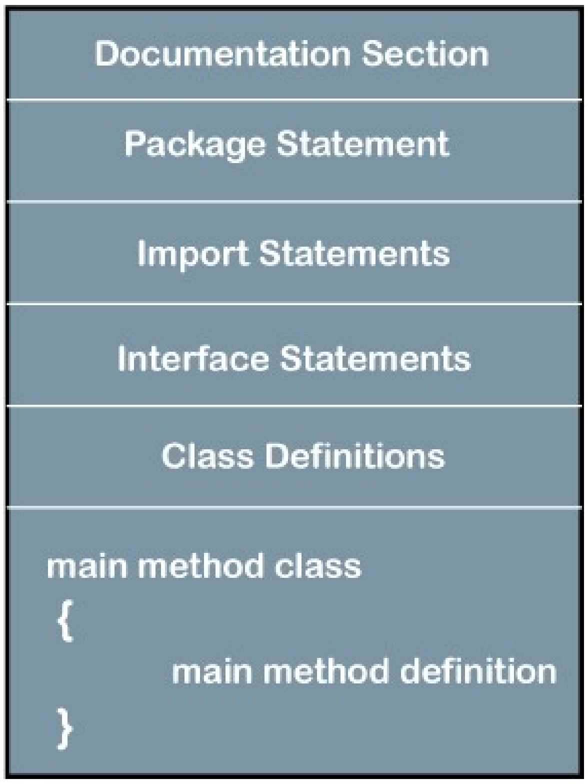
Step 5: You will see a list of different paths, click on the New button, and then add the path where java is installed. By default, java is installed in “C:\Program Files\Java\jdk\bin” folder OR “C:\Program Files(x86)\Java\jdk\bin”. In case, you have installed java at any other location, then add that path.



```
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
C:\Program Files\Java\jdk1.8.0_131\bin
```

Step 6: Click on OK, Save the settings. Now to check whether the installation is done correctly, open the command prompt and type javac -version. You will see that java is running on your machine.

Structure of Java Program



Structure of Java Program

A typical structure of a Java program contains the following elements:

- Documentation Section
- Package Declaration
- Import Statements
- Interface Section
- Class Definition
- Main Method Class

Documentation Section

The documentation section is an important section but optional for a Java program. It includes **basic information** about a Java program. The information includes the **author's name, date of creation, version, program name, company name,** and **description** of the program. It improves the readability of the program. Whatever we write in the documentation section, the Java compiler ignores the statements during the execution of the program. To write the statements in the documentation section, we use **comments**. The comments may be **single-line, multi-line,** and **documentation** comments.

- **Single-line Comment:** It starts with a pair of forwarding slash **(//)**.

  For example: //First Java Program

- **Multi-line Comment:** It starts with a **/*** and ends with ***/.** We write between these two symbols.

  For example: /*It is an example of multiline comment*/

- **Documentation Comment:** It starts with the delimiter **(/**)** and ends with ***/**.

  For example: /**It is an example of documentation comment*/

Package Declaration

There is a provision in Java that allows you to declare your classes in a collection called package. There can be only one package statement in a Java program and it has to be at the beginning of the code before any class or interface declaration. This statement is optional.

For example:
package Student; //This statement declares that all the classes and interfaces defined in this source file are a part of the student package.

Import Statements

Many predefined classes are stored in packages in Java, an import statement is used to refer to the classes stored in other packages. An import statement is always written after the package statement but it has to be before any class declaration.

We can import a specific class or classes in an import statement. For example:

```
import java.util.Date; //imports the date class
import java.applet.*;  //imports all the classes from the java applet package
```

Interface Section

It is an optional section. We can create an **interface** in this section if required. We use the **interface** keyword to create an interface.

For example:

```
interface car
{
        void start();
        void stop();
}
```

Class Definition

In this section, we define the class. Without the class, we cannot create any Java program. A Java program may contain more than one class definition. We use the **class** keyword to define the class. It contains information about user-defined methods, variables, and constants. Every Java program has at least one class that contains the main() method.

For example:

```
class Student //class definition
{
}
```

Main Method Class

In this section, we define the **main() method.** It is essential for all Java programs. Because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class. It must be inside the class. Inside the main method, we create objects and call the methods. We use the following statement to define the main() method:

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

For example:

```
public class Student //class definition
{
        public static void main(String args[])
        {
        //statements
        }
}
```

How to Compile and Run Java Program

Step 1:

Write a program on the notepad and save it with **.java** (for example, **GreetApp.java**) extension. The file name and main() method class name should be same.

```
public class GreetApp
{
        public static void main(String[] args)
        {
            System.out.println("Hi Santhosh, Nice to meet you");
        }
}
```

Step 2:

Open Command Prompt using 'cmd' command.

Step 3:

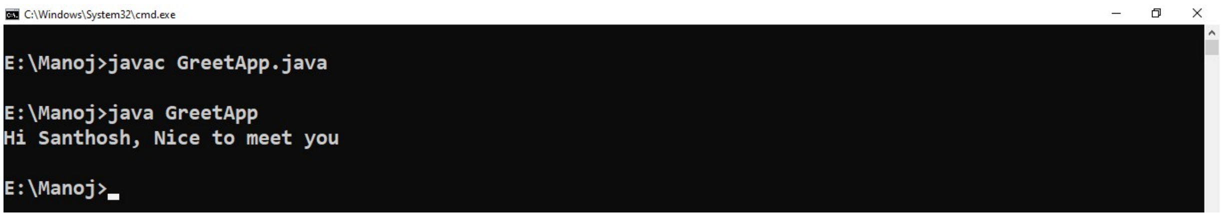
Use the following command to compile the Java program. It generates a .class file in the same folder. It also shows an error if any.

javac GreetApp.java

Step 4:

Use the following command to run the Java program:

java GreetApp



```
E:\Manoj>javac GreetApp.java

E:\Manoj>java GreetApp
Hi Santhosh, Nice to meet you

E:\Manoj>
```