

## Week - 01 Introduction to Java

### What is Java?

Java is a cross-platform object-oriented programming language that was released by Sun Microsystems in the year 1995. Today, Java is needed to run various applications such as games, social media applications, audio and video applications, etc.

### THE COMPLETE HISTORY OF JAVA PROGRAMMING LANGUAGE

- The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time.
- The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc.
- Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.
- Java Programming Language was conceived by the effort of 5 great people, James Gosling, Patrick Naughton, Chris Warth, Mike Sheridan and Ed Frank. They all worked for Sun Microsystems, Inc and came up with Java in 1991.
- The idea was to develop a language which was platform-independent and which could create embedded software's for consumer electronic devices. C and C++ were quite inefficient for the purpose because they were not platform-independent as their programs have to be compiled for particular hardware before execution.
- **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called Green Team.
- Firstly, it was called "Greentalk" by James Gosling, and the file extension was .gt.
- After that, it was called Oak and was developed as a part of the Green project. Why Oak? Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.
- In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.
- Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office.
- Notice that Java is just a name, not an acronym, initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.
- JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language.
- Many Java Versions have released till now
  - JDK Alpha and Beta (1995)
  - JDK 1.0 (23rd Jan 1996),
  - JDK 1.1 (19th Feb 1997)
  - J2SE 1.2 (8th Dec 1998)
  - J2SE 1.3 (8th May 2000)

- J2SE 1.4 (6th Feb 2002)
- J2SE 5.0 (30th Sep 2004)
- Java SE 6 (11th Dec 2006)
- Java SE 7 (28th July 2011)
- Java SE 8 (18th Mar 2014)
- Java SE 9 (21st Sep 2017)
- Java SE 10 (20th Mar 2018)
- Java SE 11 (September 2018)
- Java SE 12 (March 2019)
- Java SE 13 (September 2019)
- Java SE 14 (Mar 2020)
- Java SE 15 (September 2020)
- Java SE 16 (Mar 2021)
- Java SE 17 (September 2021)
- Java SE 18 (to be released by March 2022)

Since Java SE 8 release, the Oracle corporation follows a pattern in which every even version is release in March month and an odd version released in September month.

## **JAVA FEATURES (CHARACTERISTICS OR BUZZWORD)**

### **Java is simple**

- Fixes some clumsy features of C++
- No pointers
- Automatic garbage collection
- Rich pre-defined class library

### **Java is object-oriented (Pure OOP)**

- Focus on the data (objects) and methods manipulating the data
- All functions are associated with objects
- Almost all data types are objects (files, strings, etc.)
- Potentially better code organization and reuse

### **Java is Compiled and Interpreted**

- Java compiler generate byte-codes, not native machine code
- The compiled byte-codes are platform-independent
- Java bytecodes are translated on the fly to machine readable instructions in runtime (Java Virtual Machine)

### **Java is Architectural Neutral (Platform-Independent)**

- “Write Once Run Anywhere, Anytime, Forever” (WORA)
- Bytecodes runs in all platforms like Macintosh, PC, Unix and whatever the future platforms can offer.

### **Java is portable**

- Same application runs on all platforms
- The sizes of the primitive data types are always the same
- The libraries define portable interfaces

### **Java is Reliable**

- Extensive compile-time and runtime error checking

- No pointers but real arrays. Memory corruptions or unauthorized memory accesses are impossible
- Automatic garbage collection tracks objects usage over time

**Java is Secure**

- Usage in networked environments requires more security
- Memory allocation model is a major defence.
- Access restrictions are forced (private, public)

**Java is Multithreaded**

- Multiple concurrent threads of executions can run simultaneously
- Utilizes a sophisticated set of synchronization primitives (based on monitors and condition variables paradigm) to achieve this

**Java is Dynamic**

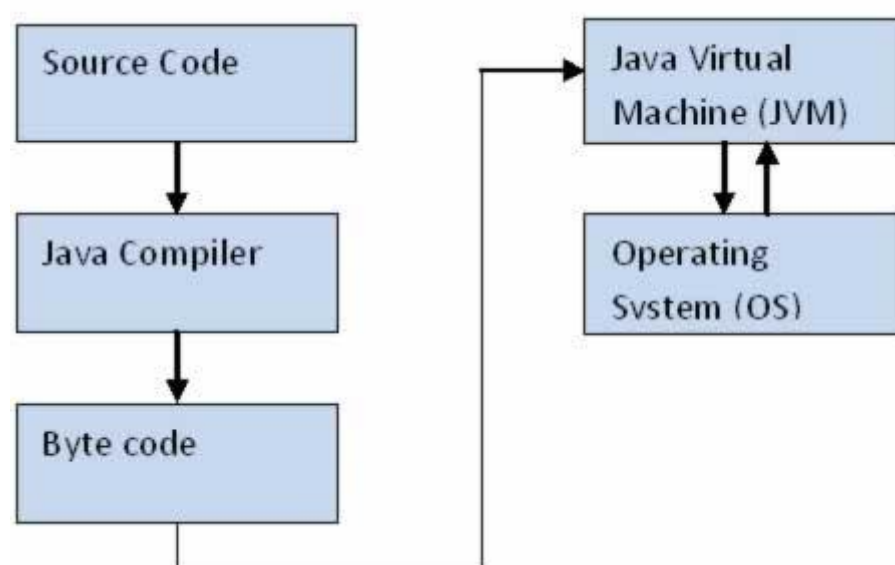
- Java is designed to adapt to evolving environment
- Libraries can freely add new methods and instance variables without any effect on their clients
- Interfaces promote flexibility and reusability in code by specifying a set of methods an object can perform, but leaves open how these methods should be implemented
- Can check the class type in runtime

## JAVA ARCHITECTURE

Java Architecture in simple steps.

- In Java, there is a process of compilation and interpretation.
- The code written in [Java](#), is converted into byte codes which is done by the Java Compiler.
- The byte codes, then are converted into machine code by the JVM.
- The Machine code is executed directly by the machine.

This diagram illustrates the internal working of a Java code, or precisely, Java Architecture!



## COMPONENTS OF JAVA

There are three main components of Java language:

1. JVM (Java Virtual Machine),
2. JRE (Java Runtime Environment), and
3. JDK (Java Development Kit)

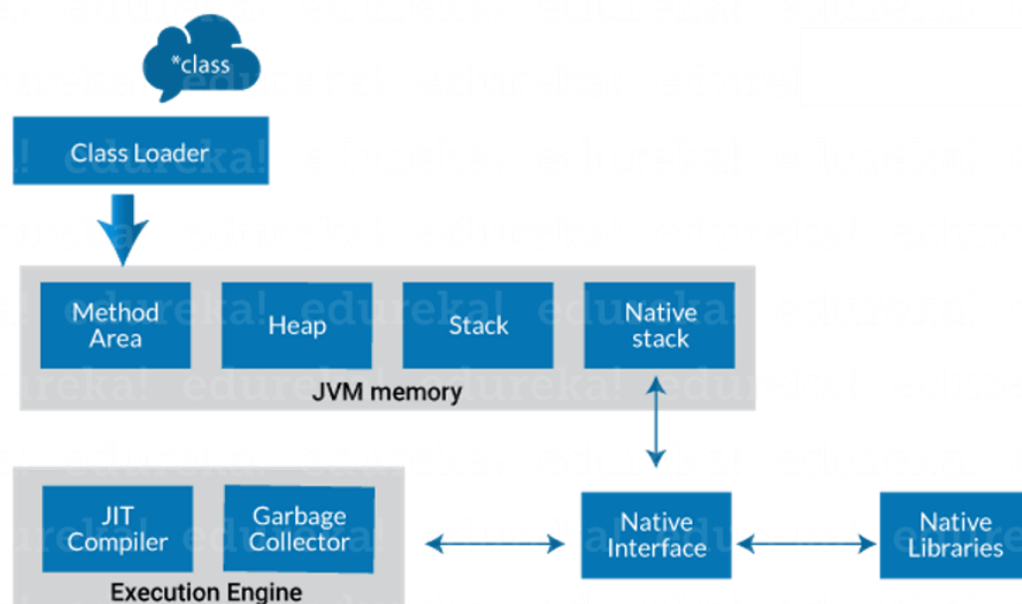
### Java Virtual Machine:

Ever heard about WORA? (Write once Run Anywhere). Well, Java applications are called WORA because of their ability to run a code on any platform. This is done only because of JVM. The JVM is a Java platform component that provides an environment for executing Java programs. JVM interprets the bytecode into machine code which is executed in the machine in which the Java program runs.

JVM performs the following functions:

- Loads the code
- Verifies the code
- Executes the code
- Provides runtime environment

Below figure shows JVM architecture



**Class Loader:** Class loader is a subsystem of JVM. It is used to load class files. Whenever we run the java program, class loader loads it first.

**Class method area:** It is one of the Data Area in JVM, in which Class data will be stored. Static Variables, Static Blocks, Static Methods, Instance Methods are stored in this area.

**Heap:** A heap is created when the JVM starts up. It may increase or decrease in size while the application runs.

**Stack:** JVM stack is known as a thread stack. It is a data area in the JVM memory which is created for a single execution thread. The JVM stack of a thread is used by the thread to store various elements i.e.; local variables, partial results, and data for calling method and returns.

**Native stack:** It subsumes all the native methods used in your application.

### Execution Engine:

- JIT compiler
- Garbage collector

**JIT compiler:** The [Just-In-Time \(JIT\) compiler](#) is a part of the runtime environment. It helps in improving the performance of Java applications by compiling bytecodes to machine code at run time. The JIT compiler is enabled by default. When a method is compiled, the JVM calls the compiled code of that method directly. The JIT compiler compiles the bytecode of that method into machine code, compiling it “just in time” to run.

**Garbage collector:** As the name explains that [Garbage Collector](#) means to collect the unused material. Well, in JVM this work is done by Garbage collection. It tracks each and every object available in the JVM heap space and removes unwanted ones. Garbage collector works in two simple steps known as Mark and Sweep:

- Mark – it is where the garbage collector identifies which piece of memory is in use and which are not
- Sweep – it removes objects identified during the “mark” phase.

### Java Runtime Environment:

The JRE software builds a runtime environment in which Java programs can be executed. The JRE is the on-disk system that takes your Java code, combines it with the needed libraries, and starts the JVM to execute it. The JRE contains libraries and software needed by your Java programs to run. JRE is a part of JDK but can be downloaded separately.

### Java Development Kit (JDK)

The JDK comes with a collection of tools that are used for developing and running java programs.

Tools	Description
java	Java interpreter, which runs applets and applications by reading and interpreting bytecode files
javac	The java compiler, which translates Java source code to bytecode files that the interpreter can understand
javadoc	Creates HTML-format documentation from java source code files

javah	Produces header files for use with native methods
javap	Java disassembler, which enables us to convert bytecode files into a program description
jdb	Java debugger, which helps us to find errors in our programs

**JDK=JRE + DEVELOPMENT TOOLS**

**JRE = JVM + LIBRARY CLASSES**

## JAVA APPLICATIONS

Some top applications of Java are:

- Mobile Applications
- Desktop GUI Applications
- Web-based Applications
- Enterprise Applications
- Scientific Applications
- Gaming Applications
- Cloud-based Applications

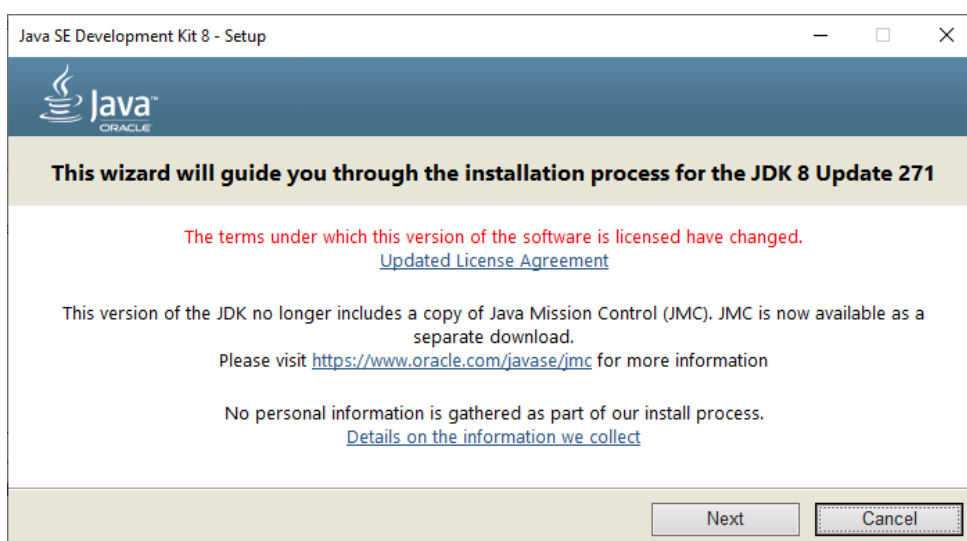
## JAVA ENVIRONMENT SETUP

### Install And Setup Java Environment [Practice Session]

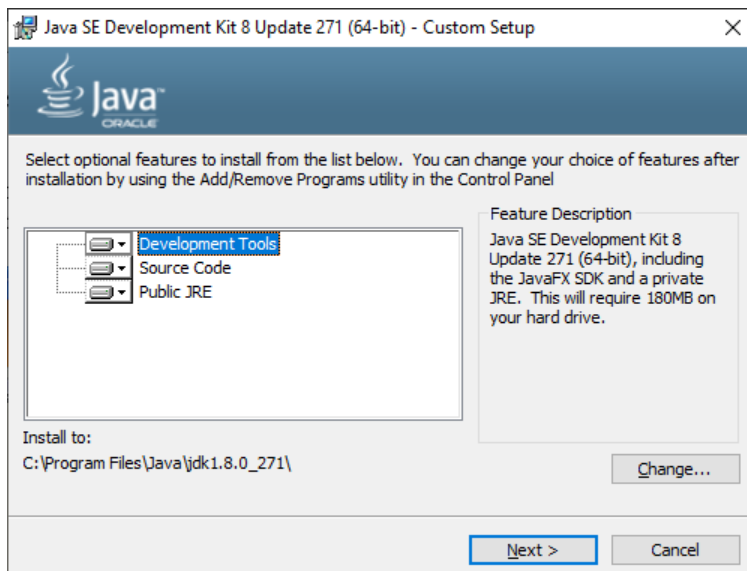
You can download the JDK 8 from this link click (CTRL + CLICK) here → [jdk software](#)  
or

You can download from <https://www.oracle.com/java/technologies/downloads/>

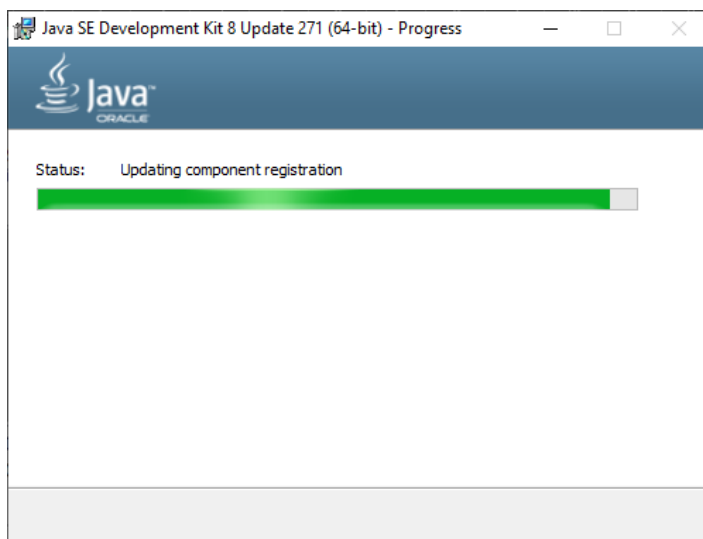
For this you have to create an account on Oracle



**Step 1:** click on the *jdk-8u271-windows-x64* exe file and click on Next button



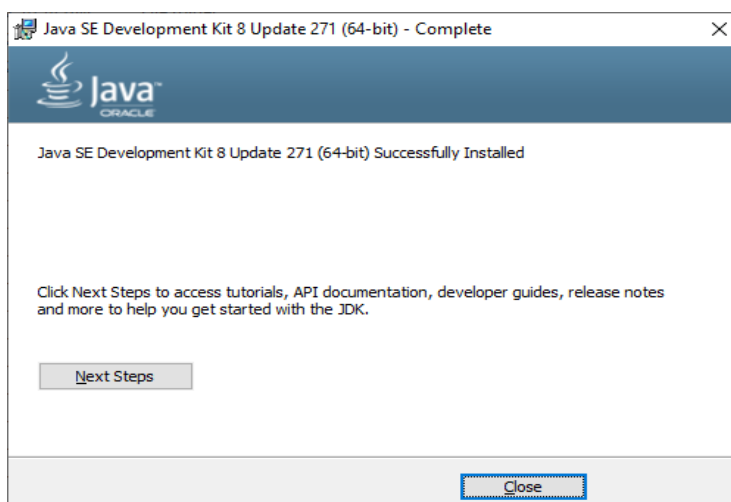
**Step 2:** if you want to change the path change it or else let it be default depends on your convenience. And click on to Next



**Step 3:** This gives the window for installing wait till it goes to another window.

**Step 4:** if you want to change the JRE path you can change it or let it be default depends on your convenience and click on to Next

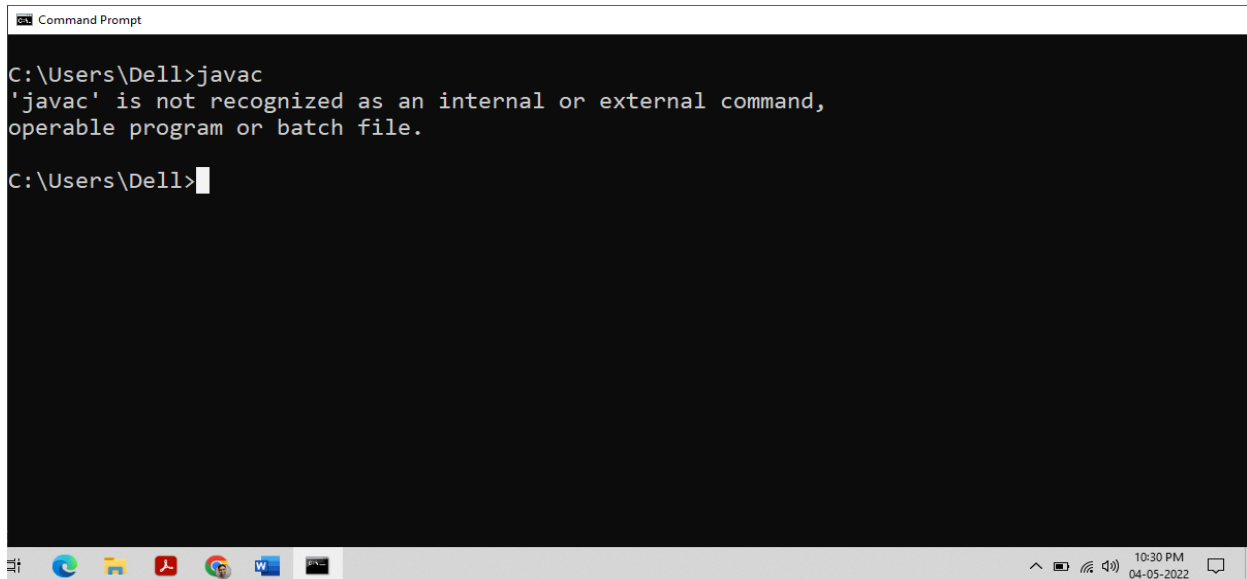
**Step 5:** wait until it completes



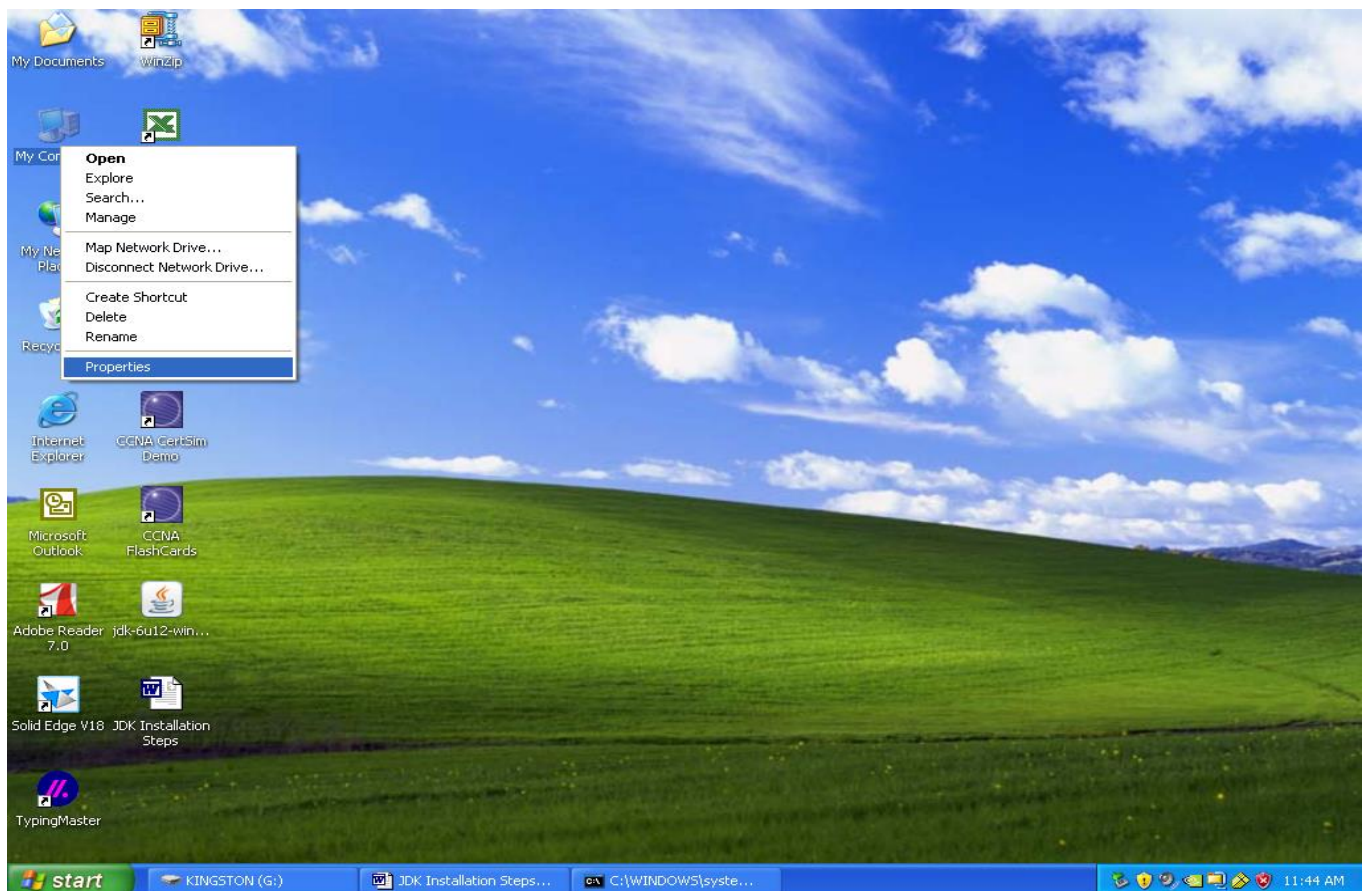


**Step 6:** Now you have successfully installed JDK software only when you click Finish/close button. After installing your JDK your work is not done Still lot more things to do.....

### Setting up your path and classpath for making your programs run anywhere in the drive

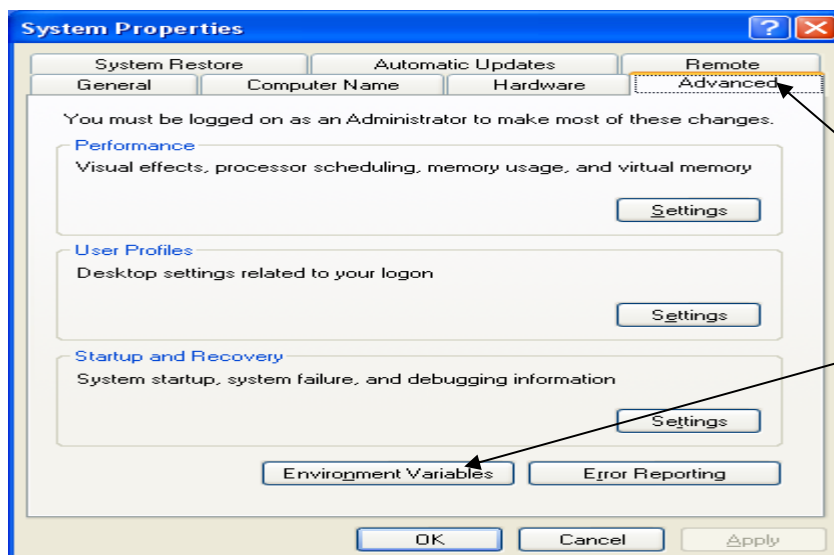


Open the command prompt and give javac command you get the above error as javac is not recognized as an internal or external command. So you need to set the path. Here how it goes.....

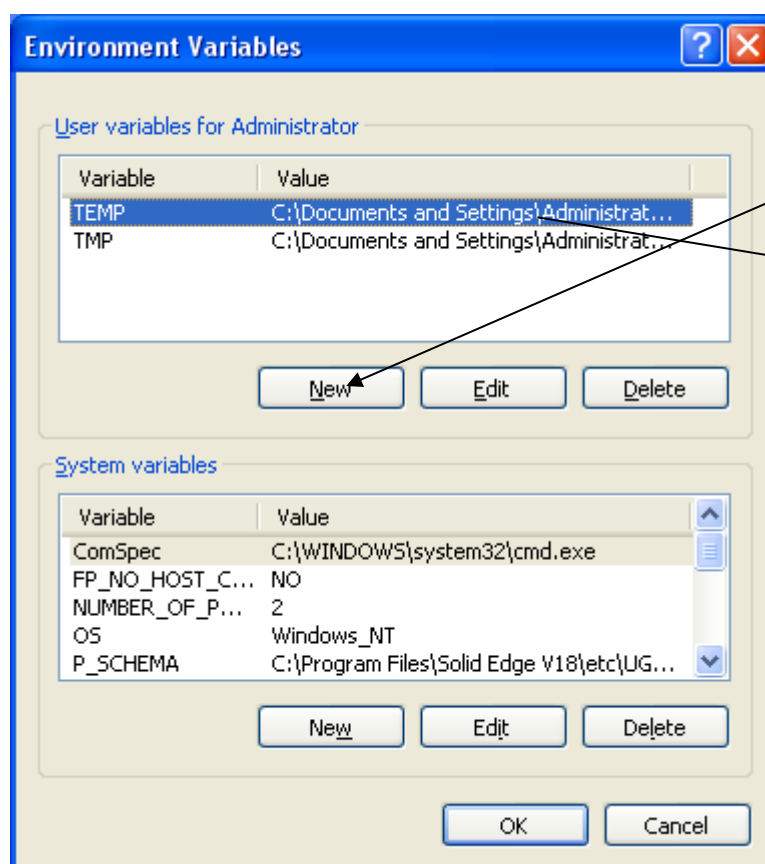


**Right click on My computer and click properties**





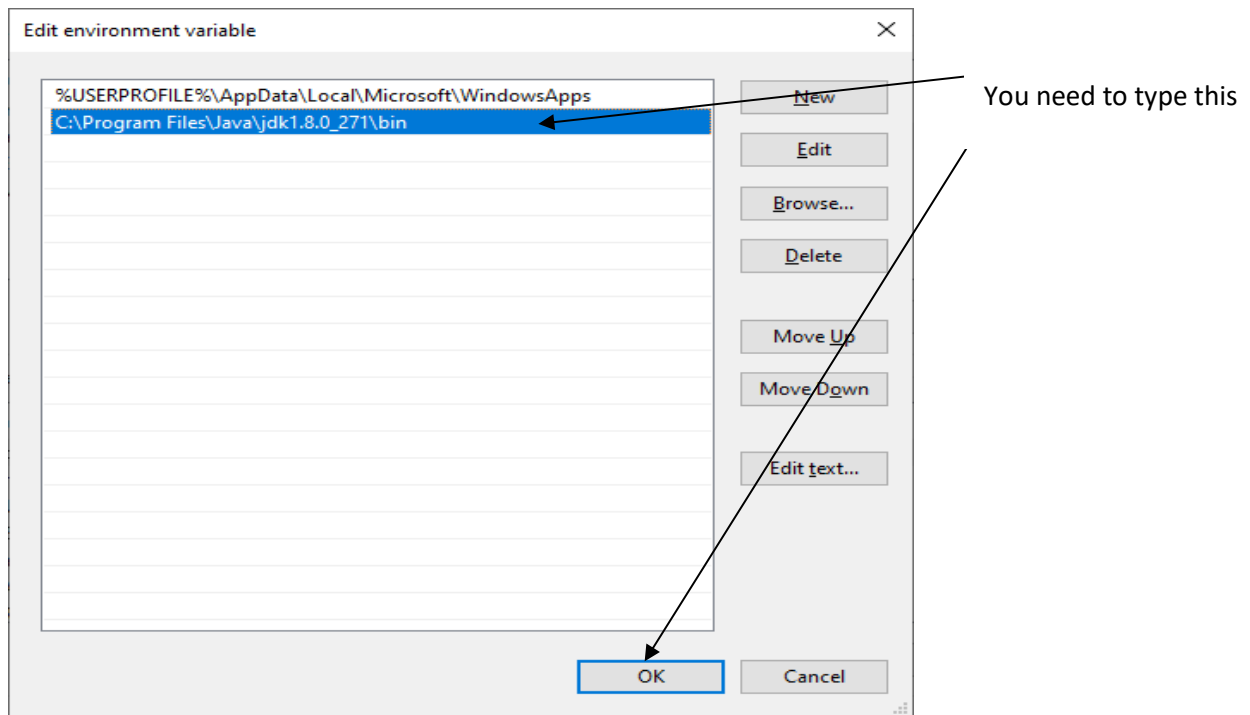
Click on the advanced tab... in the system properties dialog box... and then click for environment variables.. Button



If you don't have **path** in the display then go for new otherwise go for Edit button

In this we don't have path in the display we'll go for new

In this section you have 2 parts one is **user variables** and **system variables**



click the new tab in the *user variable* and type  
**C:\Program Files\Java\jdk1.8.0\_271\bin** next click on **OK**

Because the jdk is installed in this path and you should take the full path till bin because bin contains javac tool

Finally click on to Ok

Now open a Fresh command prompt and give *javac -version* and *java -version*

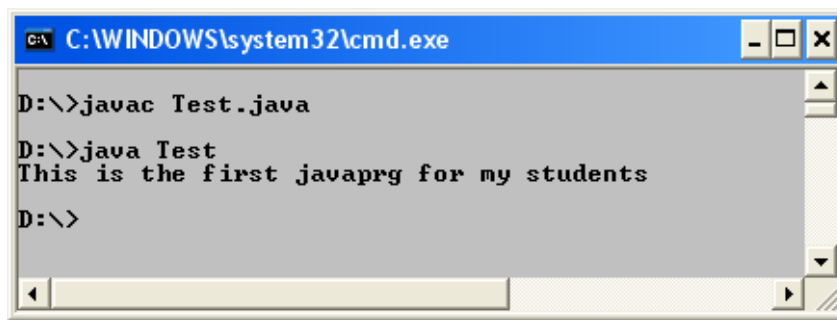
```
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell>javac -version
javac 1.8.0_271

C:\Users\Dell>java -version
java version "1.8.0_271"
Java(TM) SE Runtime Environment (build 1.8.0_271-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.271-b09, mixed mode)

C:\Users\Dell>
```

Now you can see the version displayed now you path and class path is set successfully and you can run anywhere in your hard disk drives (c, d, e etc...)



```
C:\WINDOWS\system32\cmd.exe

D:\>javac Test.java
D:\>java Test
This is the first javaprg for my students
D:\>
```

- This Test.java is in the D: drive since you have already set the path and class path. Any where you can compile and run..
- If you use any of the IDE's like eclipse IDE, Netbeans IDE you no need to worry of the path settings. This IDE's will take care..
- But this path setting should be known by every one because if you sit working in any of the system in any part of the world unfortunately you might not get this IDE software at that moment. only option is to use notepad and set the path.

## STRUCTURE OF JAVA PROGRAM

Document Section
Package Section
Import Statements
Interface Statements
Class Definitions
Main method class { Method definition }

### Documentation Section

It includes the comments that improve the readability of the program. A comment is a non-executable statement that helps to read and understand a program especially when your programs get more complex.

This section is optional and comments may appear anywhere in the program. Java programming language supports three types of comments.

#### Single line (or end-of line) comment:

It starts with a double slash symbol (//) and terminates at the end of the current line. The compiler ignores everything from // to the end of the line. For example:

```
// Calculate sum of two numbers
```

**Multiline Comment:**

Java programmer can use C/C++ comment style that begins with delimiter `/*` and ends with `*/`. All the text written between the delimiter is ignored by the compiler.

```
/*calculate sum of two numbers  
and it is a multiline comment */
```

**Documentation comments:**

This comment style is new in Java. Such comments begin with delimiter `/**` and end with `*/`.

```
/** The text enclosed here will be part of program documentation */
```

**Package Statement**

A package is a collection of classes, interfaces and sub-packages. A sub package contains collection of classes, interfaces and sub-sub packages etc.

```
package institute;
```

**Import statements**

Java contains many predefined classes that are stored into packages. In order to refer these standard predefined classes in your program, you need to use fully qualified name (i.e. `Packagename.className`).

Example If you want to import Date class of java.util package using import statement then write.

```
import java.util.Date;
```

**Interface Section**

It is an optional section and is used when we wish to implement multiple inheritance feature in the program.

**Class Definition:**

Java program may contain multiple class definition. Classes are primary feature of Java program.

**Main Method Class Section:**

The Class section describes the information about user-defined classes present in the program. A class is a collection of fields (data variables) and methods that operate on the fields. Every program in Java consists of at least one class, the one that contains the main method. The `main ()` method which is from where the execution of program actually starts and follow the statements in the order specified.

## Compilation and Execution of Java Program [Practice Session]

### 1) Create the source file:

- Open a text editor, type in the code which defines a class (HelloWorld) and then save it in a file (HelloWorld.java)
- File and class name are case sensitive and must be matched exactly (except the .java part)

*Example Code: HelloWorld.java*

```
/**
 * The HelloWorld class implements an application
 * that displays "Hello World!" to the standard output
 */
public class HelloWorld {

    public static void main(String[] args) {

        // Display "Hello World!"

        System.out.println("Hello World!");

    }

}
```

### 2) Compile the program:

compile HelloWorld.java by using the following command:

**javac HelloWorld.java** it generates a file named HelloWorld.class

- ⊗ **‘javac’ is not recognized as an internal or external command, operable program or hatch file. javac: Command not found**

If you see one of these errors, you have two choices:

- 1) Specify the full path in which the javac program locates every time. For example: C:\j2sdk1.4.2\_09\bin\javac HelloWorld.java
- 2) Set the PATH environment variable

### 3) Run the program:

- Run the code through: **java HelloWorld**
  - Note that the command is java, not javac, and you refer to
    - HelloWorld, not HelloWorld.java or HelloWorld.class
- ⊗ **Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld**

If you see this error, you may need to set the environment variable CLASSPATH.

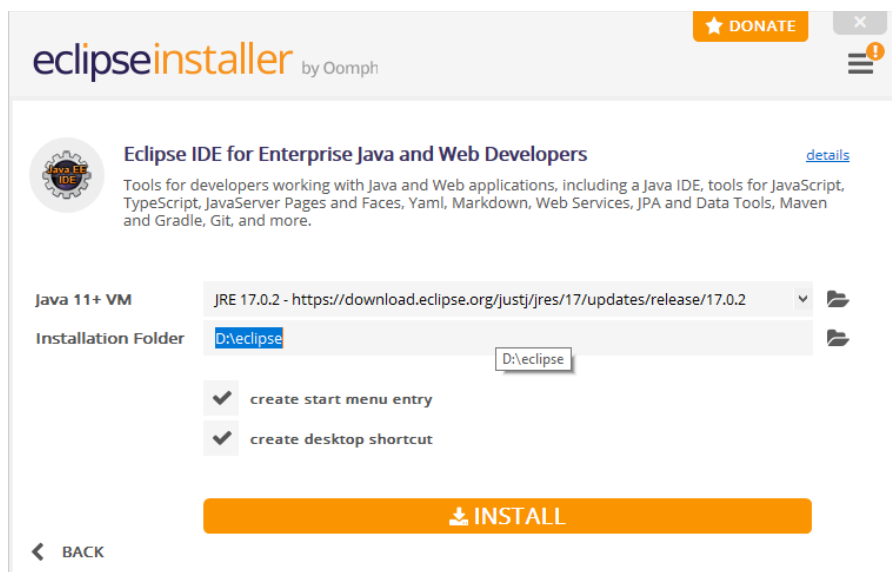
## Install Java Editor – ECLIPSE IDE [Practice Session]

### Eclipse for Enterprise Java

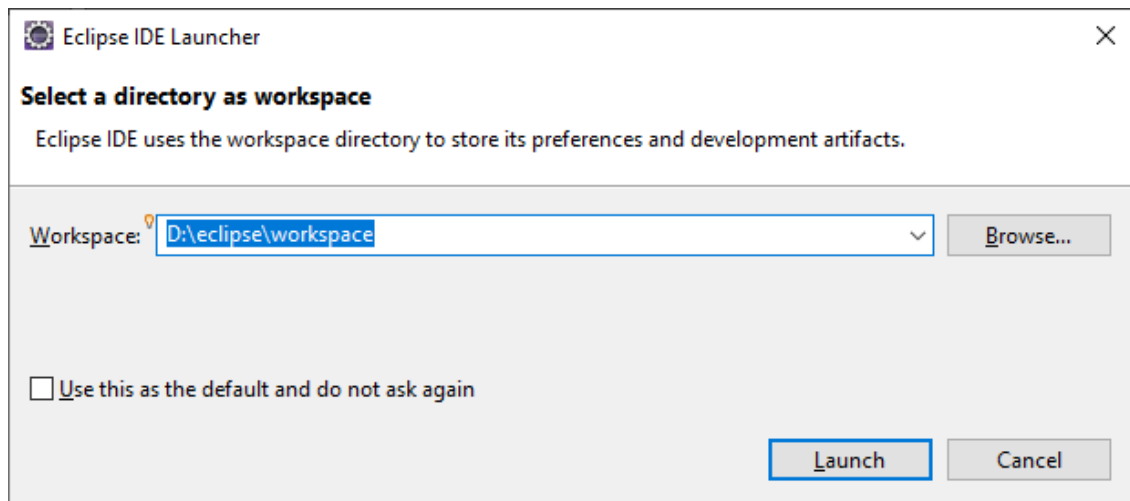
- You can download the Eclipse IDE from the link (Ctrl+Click) click→ [Eclipse Software](#) or
- Directly you can download from the website <https://www.eclipse.org/downloads/>
- Click on eclipse-inst-jre-win64 exe file to install the software



- Click on Eclipse IDE for Enterprise java and web developers



- Choose the installation folder, here I have taken D:\eclipse (created eclipse folder in d: drive)
- Click on install and in next window Accept now
- Wait until installation
- After installation finished click on Launch.
- Select the workspace folder you can choose your own path, here I have chosen D:\eclipse\workspace (created workspace folder inside D:\eclipse).
- Click on launch. Eclipse will starts loading.....
- After it loads click on Create a New Java Web Project
- Give the project name and click finish.



- After entering the project name, right click on project name here in this example it is java practice select new → class → give class name → finish.
- Here class name is Hello, when you click on finish it will generate as Hello.java
- Type the program and click on Run as Application you will get the output as below.

