# Unit 1: Programming in Java (8 Hrs.)
# BSC-CSIT Seventh Semester
# Patan Multiple Campus
**Shashi Shekhar Acharya**

Java Architecture, Java Buzzwords, Path and ClassPath variables, Sample Java Program, Compiling and Running Java Programs.

**History of C++**

C++ is a powerful general-purpose programming language. It can be used to develop operating systems, browsers, games, and so on. C++ supports different ways of programming like procedural, object-oriented, functional, and so on. This makes C++ powerful as well as flexible.

**History of Java**

The Java language was initially called OAK. Originally, it was developed for handling devices and set-top boxes. Oak was a massive failure. Thus, in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web development business.

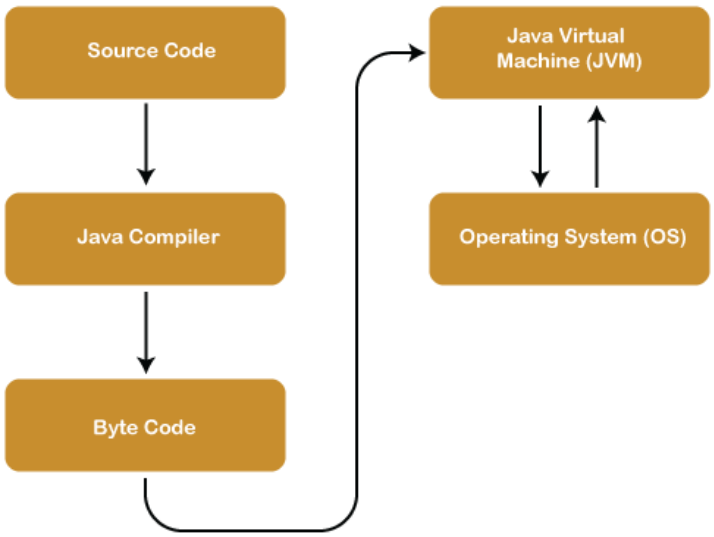
**C++ vs Java**

- C++ uses only compiler, whereas Java uses compiler and interpreter both.
- C++ supports both operator overloading & method overloading whereas Java only supports method overloading.
- C++ supports manual object management with the help of new and delete keywords whereas Java has built-in automatic garbage collection.(i.e Allocation of memory)
- C++ supports structures whereas Java doesn't supports structures.
- C++ supports unions while Java doesn't support unions.

Java is an object-oriented programming language developed by Sun Microsystems. The Java language was designed to be small, simple, and **portable across platforms** and operating systems, both at the source and at the binary level. Java is a high level, robust, object-oriented and **secure programming language**. Writing, compiling and debugging a program is easy in java.

**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**.

**Components of Java Architecture**

The Java architecture includes the three main components:

- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

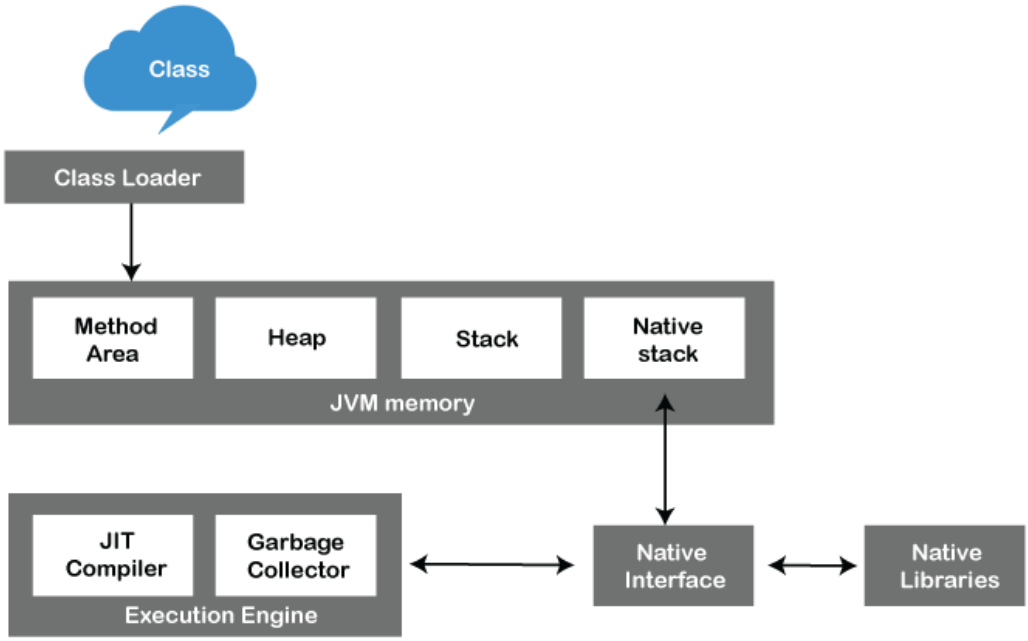
**Java Virtual Machine**

*JVM(Java Virtual Machine) acts as a run-time engine to run Java applications.* JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE (Java Runtime Environment). 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 Virtual Machine (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 falling figure represents the architecture of the JVM.



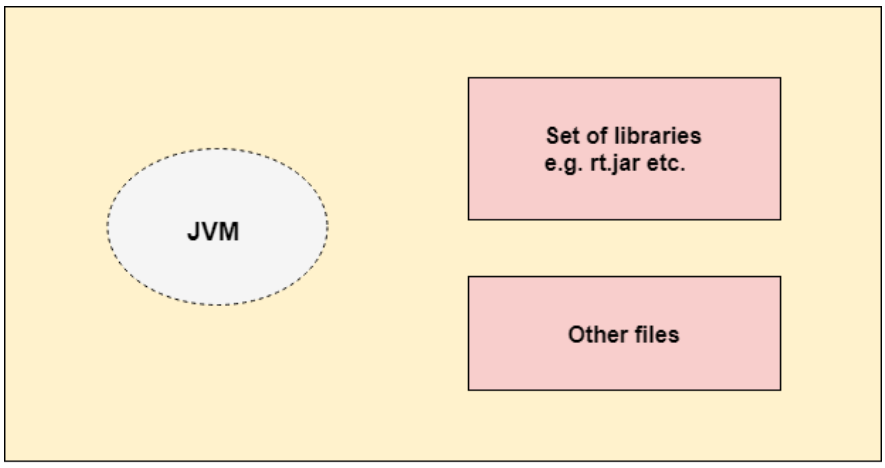

- **Class Loader:** Class Loader is a subsystem used to load class files. Class Loader first loads the Java code whenever we run it.
- **Class Method Area:** In the memory, there is an area where the class data is stored during the code's execution. Class method area holds the information of static variables, 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 increase or decrease during the application runs.
- **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. A native method can likely access the runtime data areas of the virtual machine
- **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 evens 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. Native Method Libraries are the collection of the Native Libraries(C, C++) which are required by the Execution **Engine**.

**Java Runtime Environment**

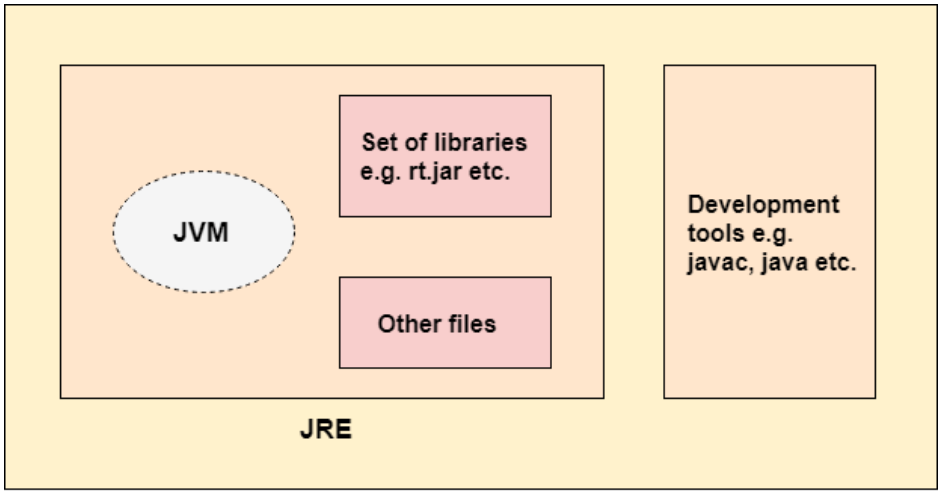
JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It contains a set of libraries + other files that JVM uses at runtime. 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.




JRE

**Java Development Kit**

It is a software development environment used in the development of Java applications and applets. Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it.




JDK

These are three main components of Java Architecture. The execution of a program is done with all these three components.

## Features of Java

The primary objective of Java Programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as java ***buzzwords*.**

## Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

## Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

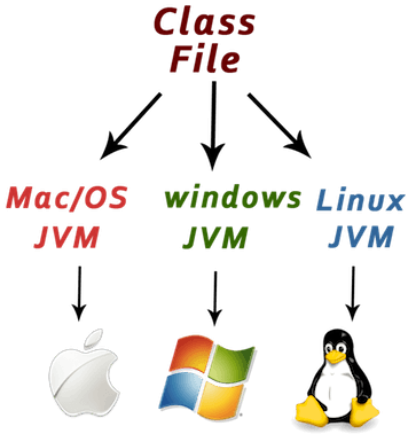
Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

## Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A ***platform*** is the hardware or software environment in which a program runs.




There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment: *The runtime environment is the environment in which a program or application is executed.*
2. API (Application Programming Interface): *An API is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.*

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

## Secured

Java is best known for its security. With Java, we can develop virus-free systems. When Java programs are executed they don't instruct commands to the machine directly. Instead Java Virtual machine reads the program (ByteCode) and convert it into the machine instructions. This

way any program tries to get illegal access to the system will not be allowed by the JVM. Allowing Java programs to be executed by the JVM makes Java program fully secured under the control of the JVM.

### Robust

Robust simply means strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

### Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed. In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

### Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

### High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

### Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

### Multi-threaded

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.

### Dynamic

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

# CLASSPATH:

CLASSPATH is an environment variable which is used by Application ClassLoader to locate and load the .class files. The CLASSPATH defines the path, to find third-party and user-defined classes that are not extensions or part of Java platform. Include all the directories which contain .class files and JAR (Java Archive) files when setting the CLASSPATH.

The default value of CLASSPATH is a dot (.). It means the only current directory searched. The default value of CLASSPATH overrides when you set the CLASSPATH variable or using the -classpath command (for short -cp). Put a dot (.) in the new setting if you want to include the current directory in the search path.

If CLASSPATH finds a class file which is present in the current directory, then it will load the class and use it, irrespective of the same name class presents in another directory which is also included in the CLASSPATH.

If you want to set multiple classpaths, then you need to separate each CLASSPATH by a semicolon (;).

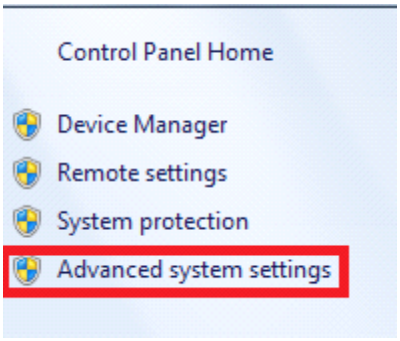
The third-party applications (MySQL and Oracle) that use the JVM can modify the CLASSPATH environment variable to include the libraries they use. The classes can be stored in directories or archives files. The classes of the Java platform are stored in rt.jar.

There are two ways to ways to set CLASSPATH: through Command Prompt or by setting Environment Variable.

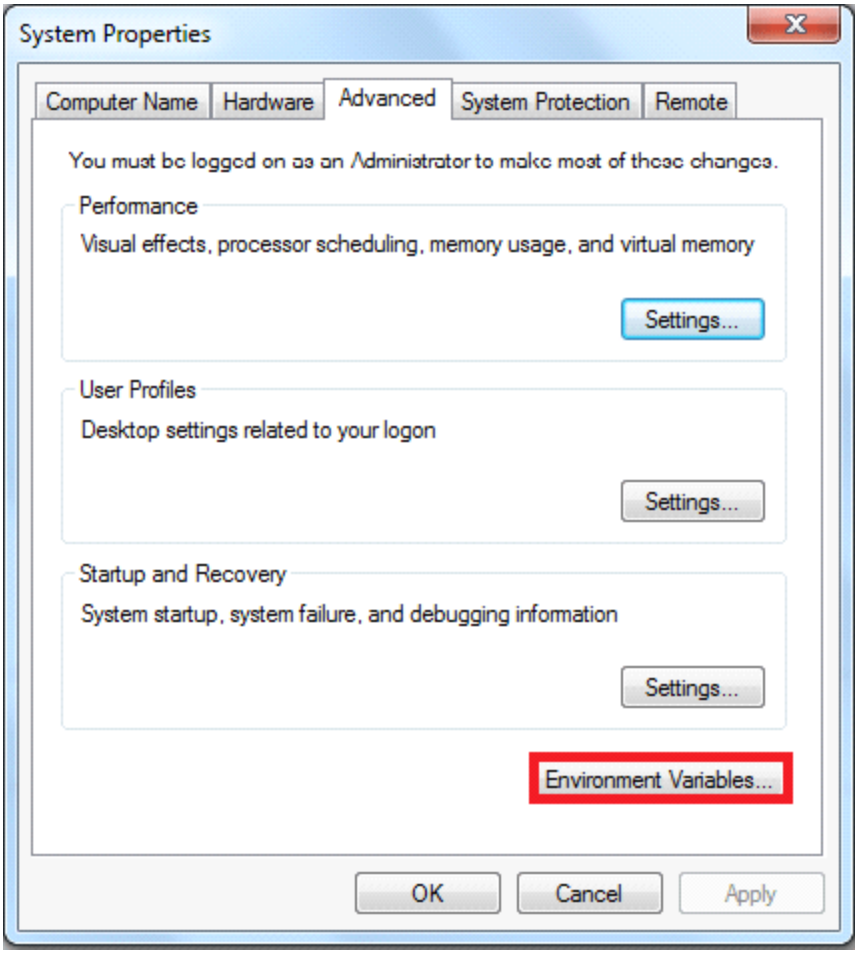
Let's see how to set CLASSPATH of MySQL database:

Step 1: Click on the Windows button and choose Control Panel. Select System.


Step 2: Click on Advanced System Settings.



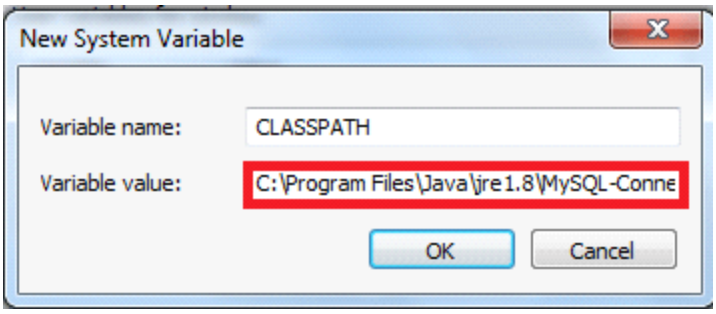

Step 3: A dialog box will open. Click on Environment Variables.




Step 4: If the CLASSPATH already exists in System Variables, click on the Edit button then put a semicolon (;) at the end. Paste the Path of MySQL-Connector Java.jar file.

If the CLASSPATH doesn't exist in System Variables, then click on the New button and type Variable name as CLASSPATH and Variable value as C:\Program Files\Java\jre1.8\MySQL-Connector Java.jar;.;

Remember: Put ;.; at the end of the CLASSPATH.




Path:

Once you installed Java on your machine, it is required to Set the PATH environment variable to conveniently run the executable (javac.exe, java.exe, javadoc.exe, and so on) from any directory without having to type the full path of the command.

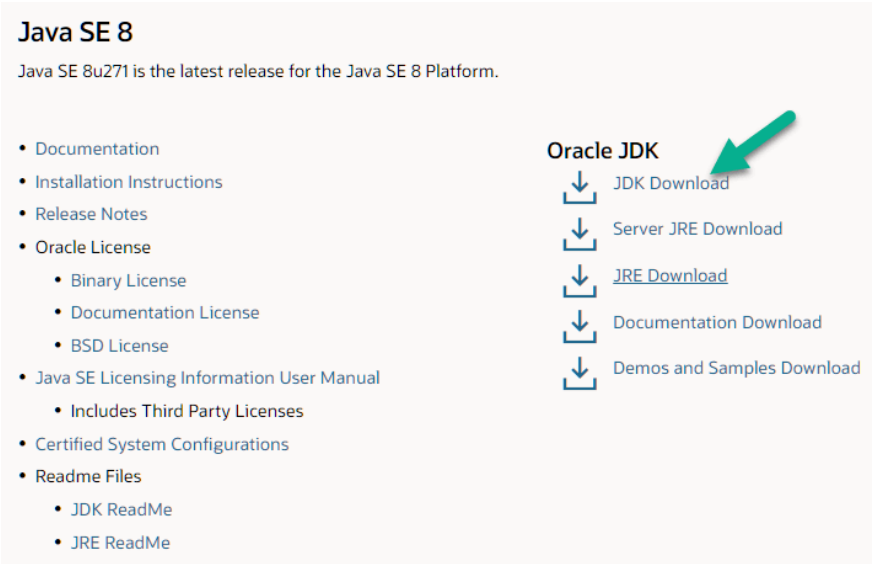
**How to Download & Install Java JDK 8 in Windows**

This Java Development Kit(JDK) allows you to code and run Java programs. It's possible that you install multiple JDK versions on the same PC. But Its recommended that you install only latest version.

**How to install Java for Windows**

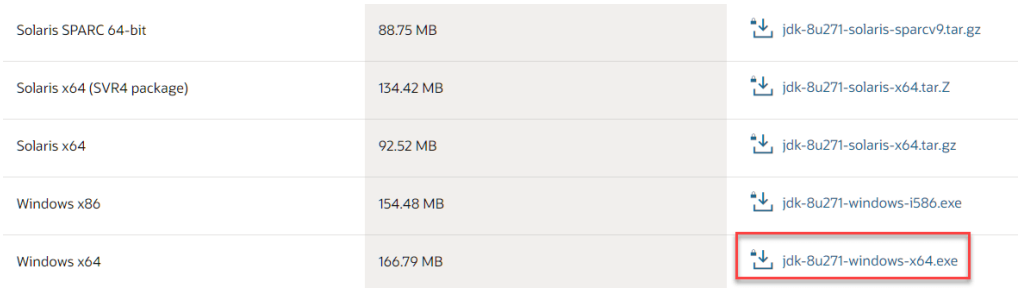
Following are the steps for JDK 8 free download for 32 bit or JDK 8 download 64 bit and installation

**Step 1)** Go to link. Click on JDK Download for Java JDK 8 download.



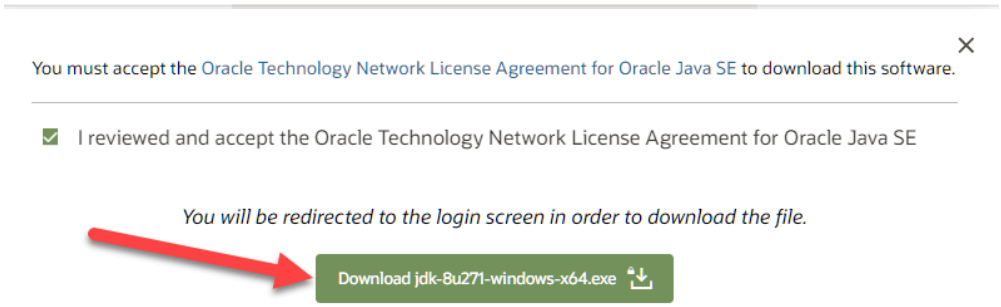

**Step 2)** Next,

1. Accept License Agreement
2. Download Java 8 JDK for your version 32 bit or JDK 8 download for windows 10 64 bit.



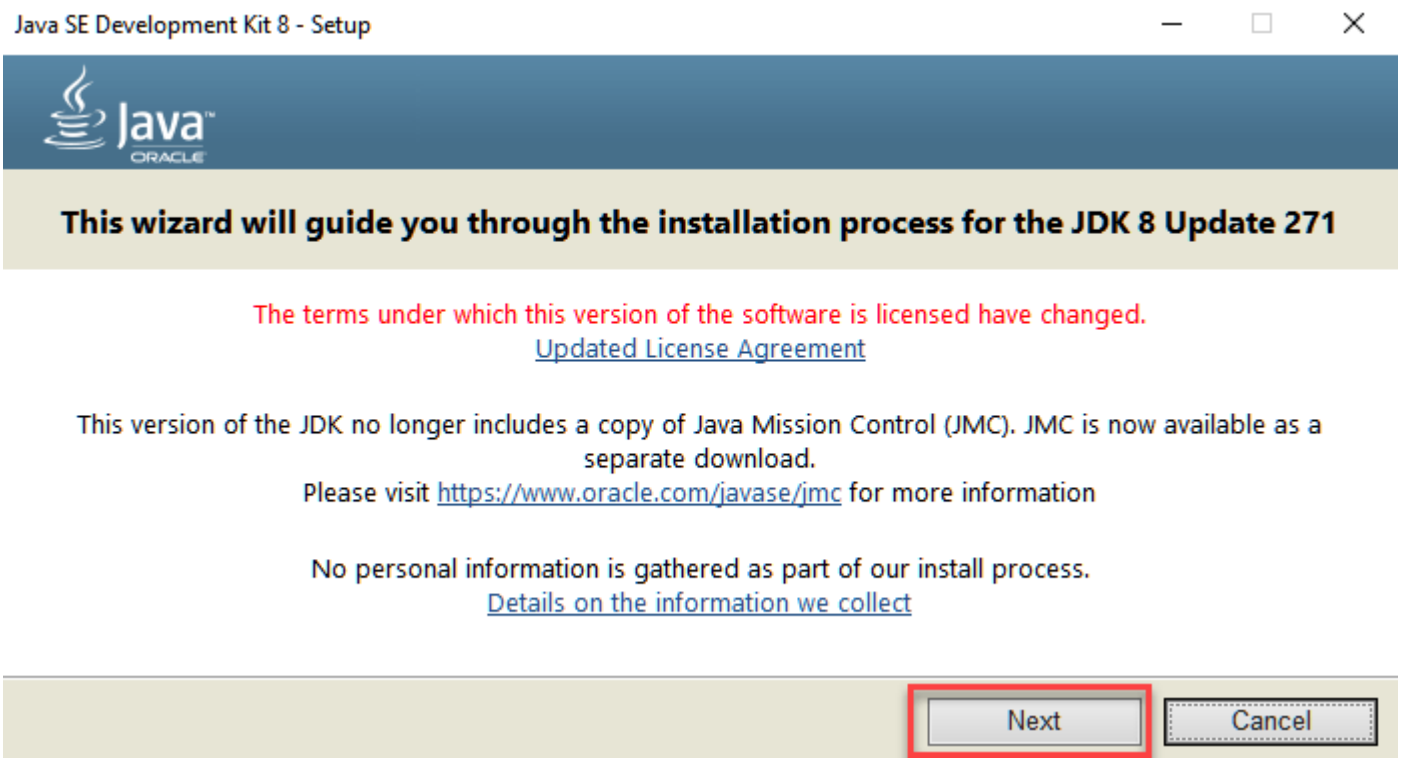
| | | |
|---|---|---|
| Solaris SPARC 64-bit | 88.75 MB | jdk-8u271-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 134.42 MB | jdk-8u271-solaris-x64.tar.Z |
| Solaris x64 | 92.52 MB | jdk-8u271-solaris-x64.tar.gz |
| Windows x86 | 154.48 MB | jdk-8u271-windows-i586.exe |
| Windows x64 | 166.79 MB | jdk-8u271-windows-x64.exe |

**Step 3)** When you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE and you will be redirected to the login page. If you don't have an oracle account you can easily sign up by adding basics details of yours.

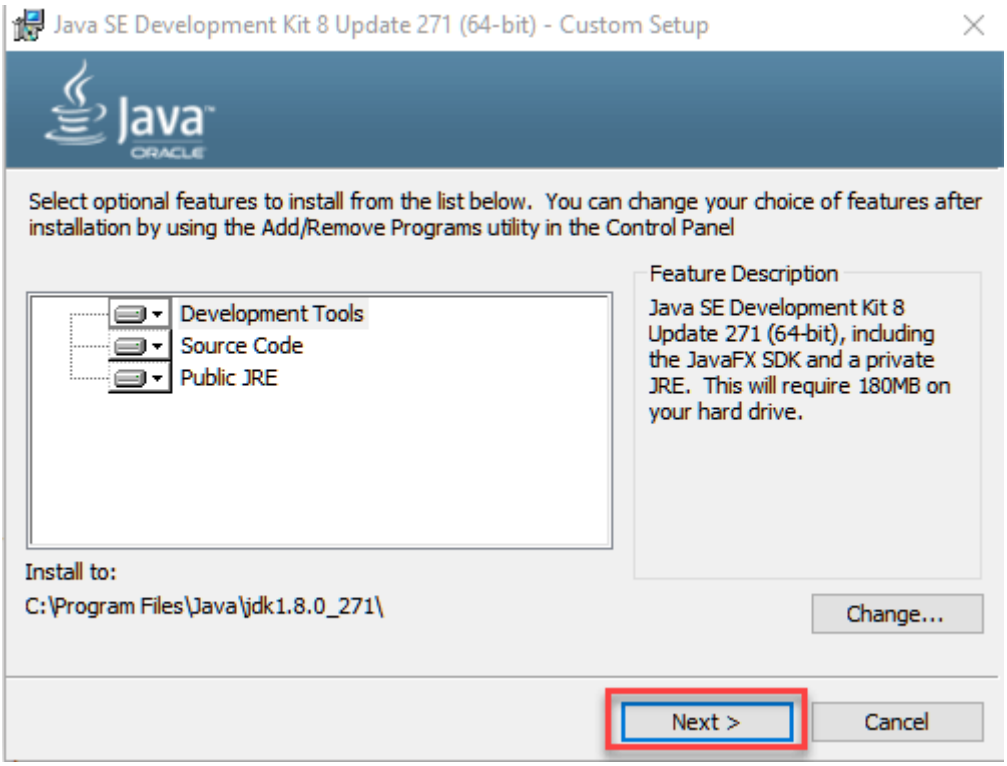



**NOTE:** You will be required to create an Oracle Account to start download of the file.

**Step 4)** Once the Java JDK 8 download is complete, run the exe for install JDK. Click Next

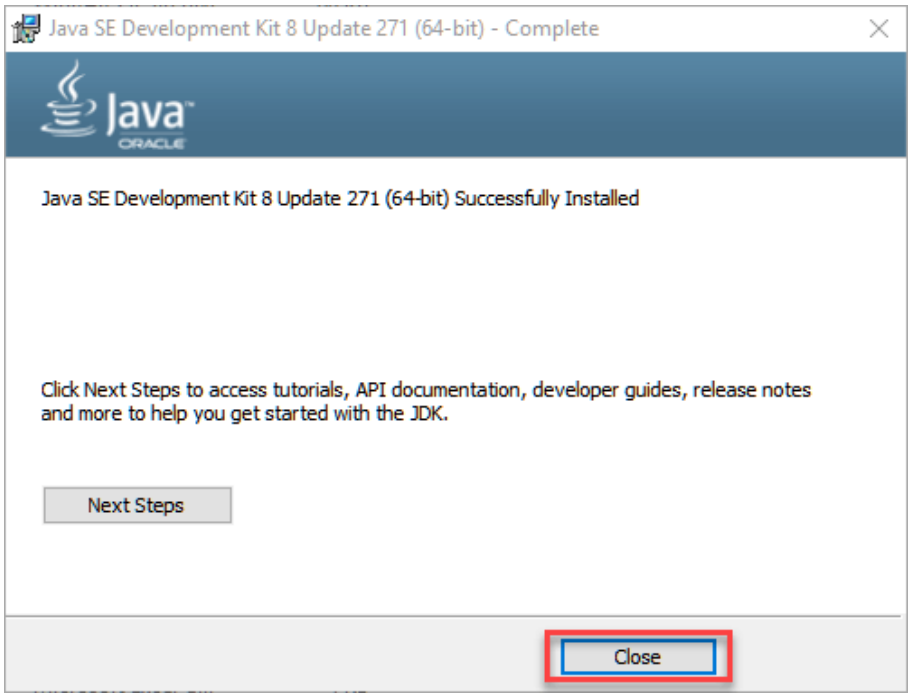



**Step 5)** Select the PATH to install Java in Windows… You can leave it Default. Click next.




**NOTE:** Follow the onscreen instructions in succeeding installation steps.

**Step 6)** Once you install Java in windows, click Close



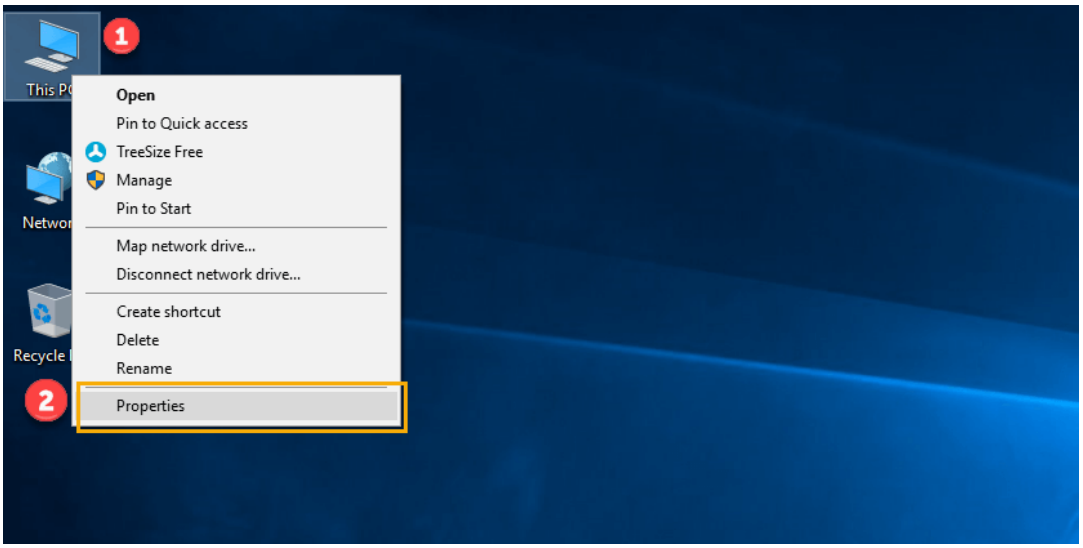

**How to set Environment Variables in Java: Path and Classpath**

The PATH variable gives the location of executables like javac, java etc. It is possible to run a program without specifying the PATH but you will need to give full path of executable like **C:\Program Files\Java\jdk1.8.0_271\bin\javac A.java** instead of simple **javac A.java**

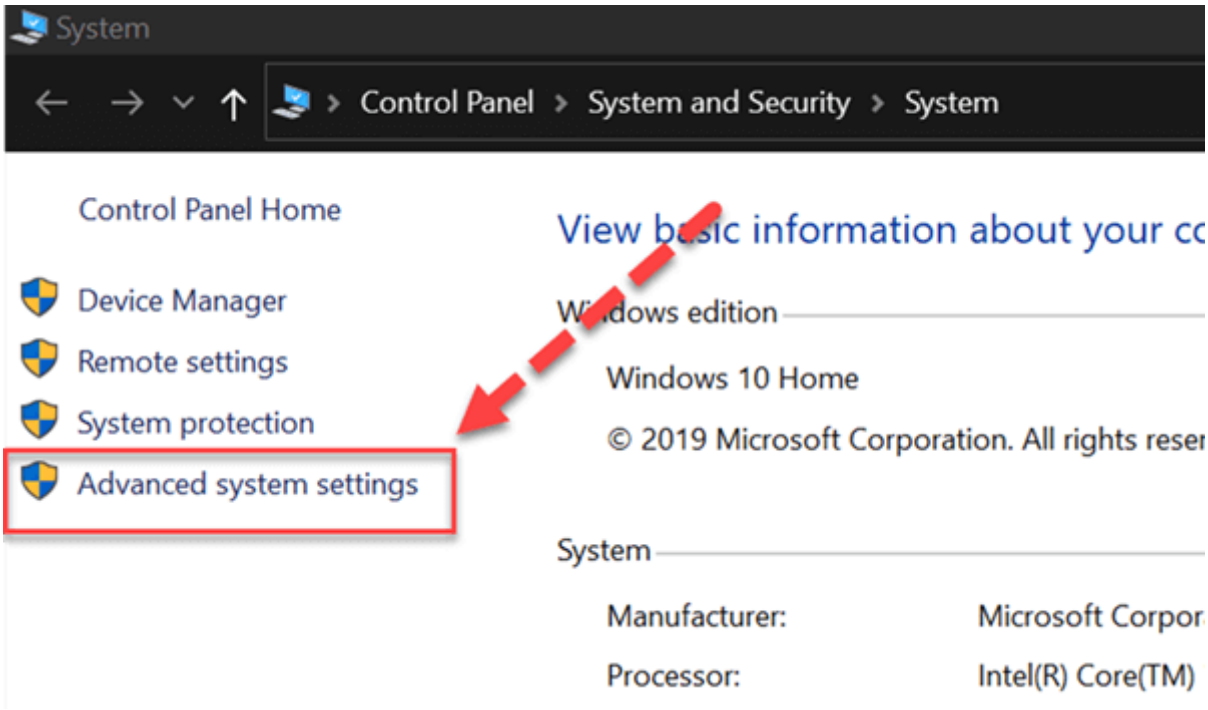
The CLASSPATH variable gives location of the Library Files.

Let's look into the steps to set the PATH and CLASSPATH

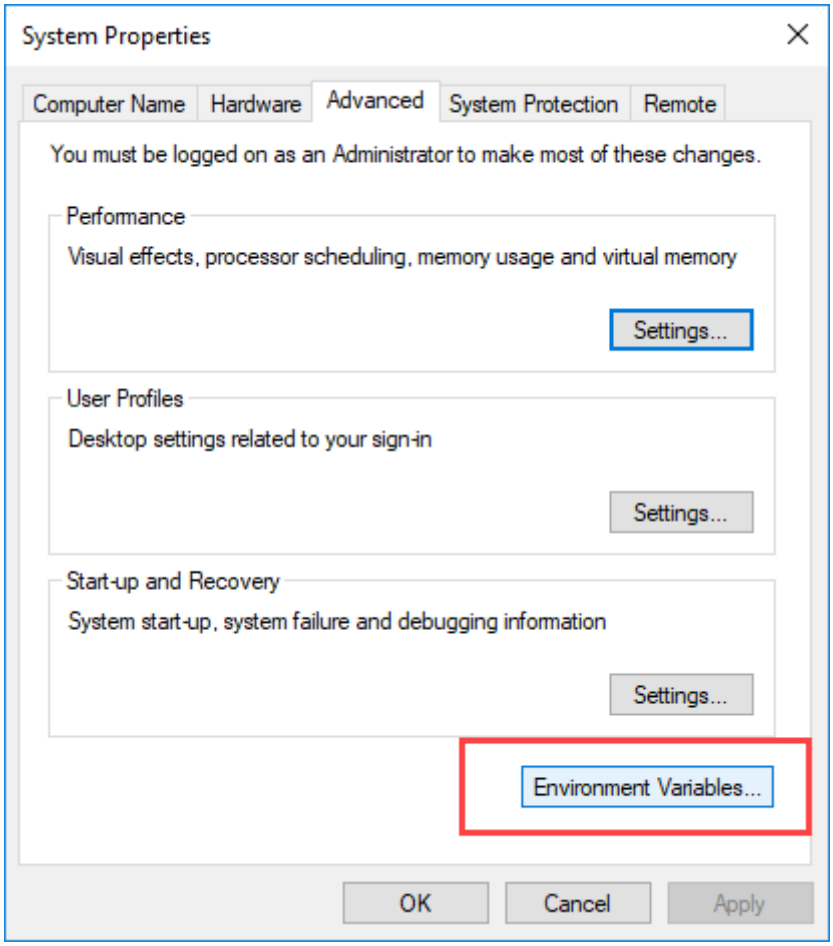
**Step 1)** Right Click on the My Computer and Select the properties




**Step 2)** Click on advanced system settings

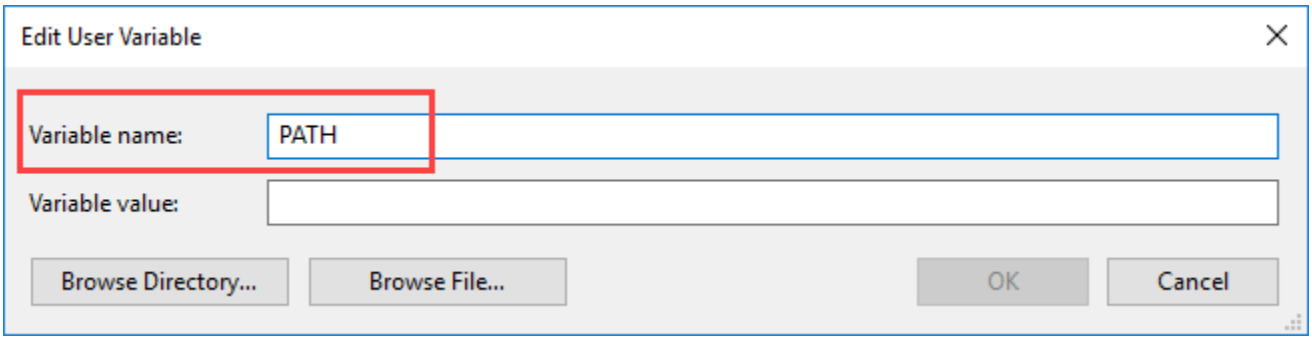



**Step 3)** Click on Environment Variables

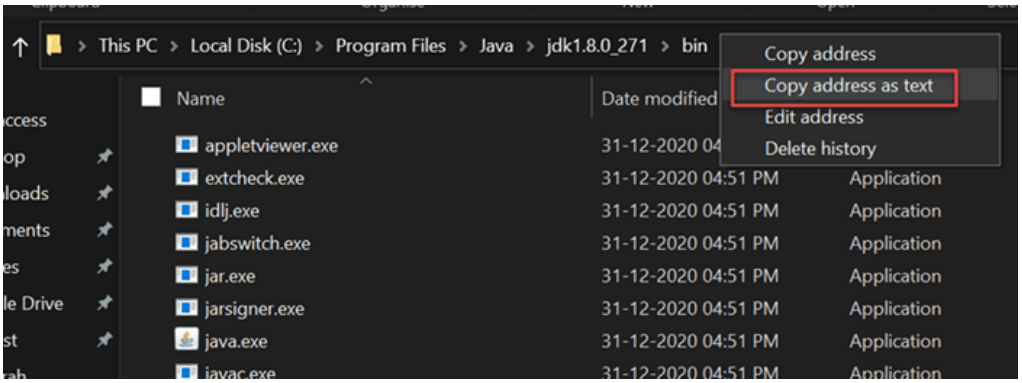



**Step 4)** Click on new Button of User variables

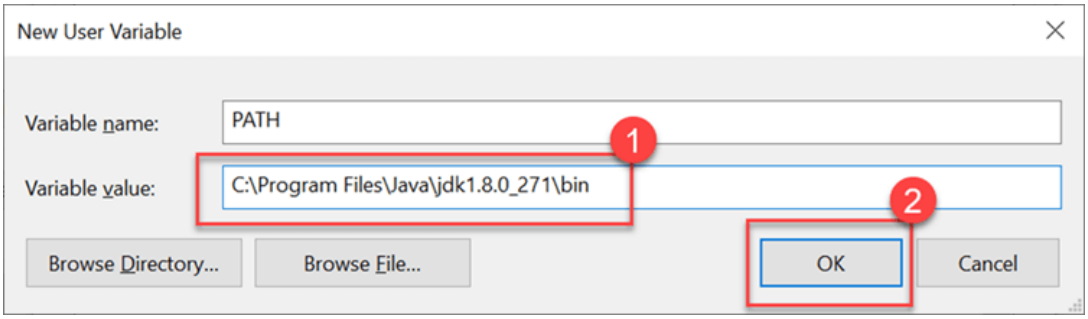
**Step 5)** Type PATH in the Variable name.




**Step 6)** Copy the path of bin folder which is installed in JDK folder.




**Step 7)** Paste Path of bin folder in Variable value. Click on OK Button.

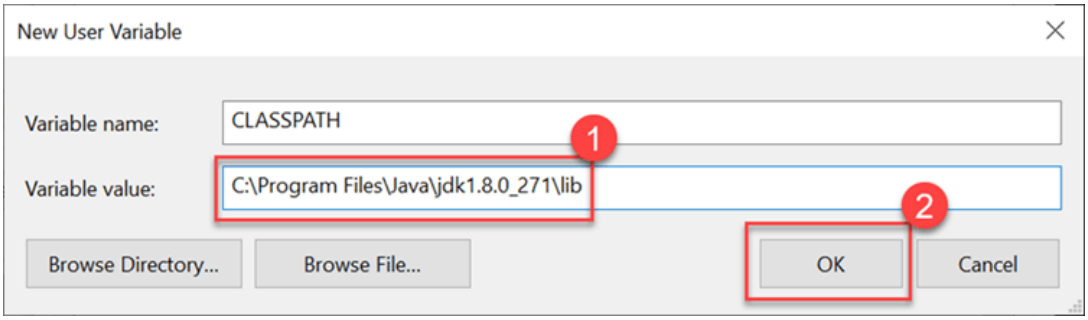



**Note:** In case you already have a PATH variable created in your PC, edit the PATH variable to

```
PATH = <JDK installation directory>\bin;%PATH%;
```

Here, %PATH% appends the existing path variable to our new value

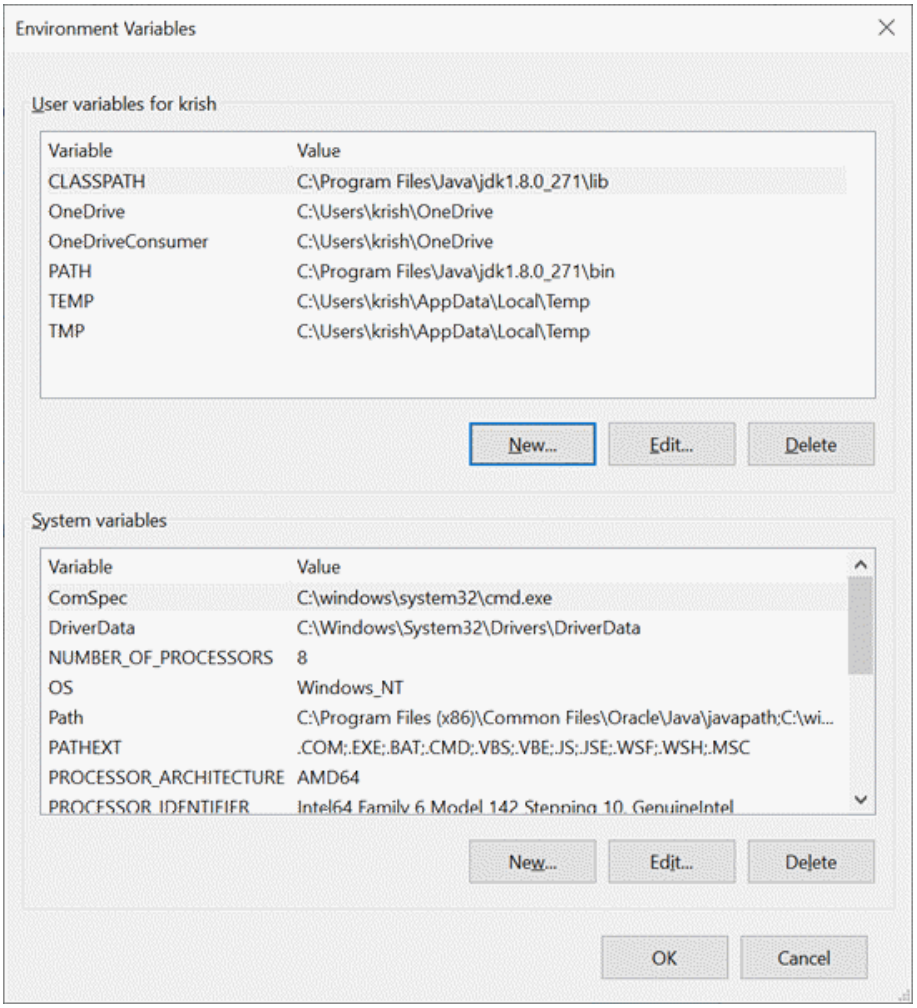
**Step 8)** You can follow a similar process to set CLASSPATH.




**Note:** In case you java installation does not work after installation, change classpath to

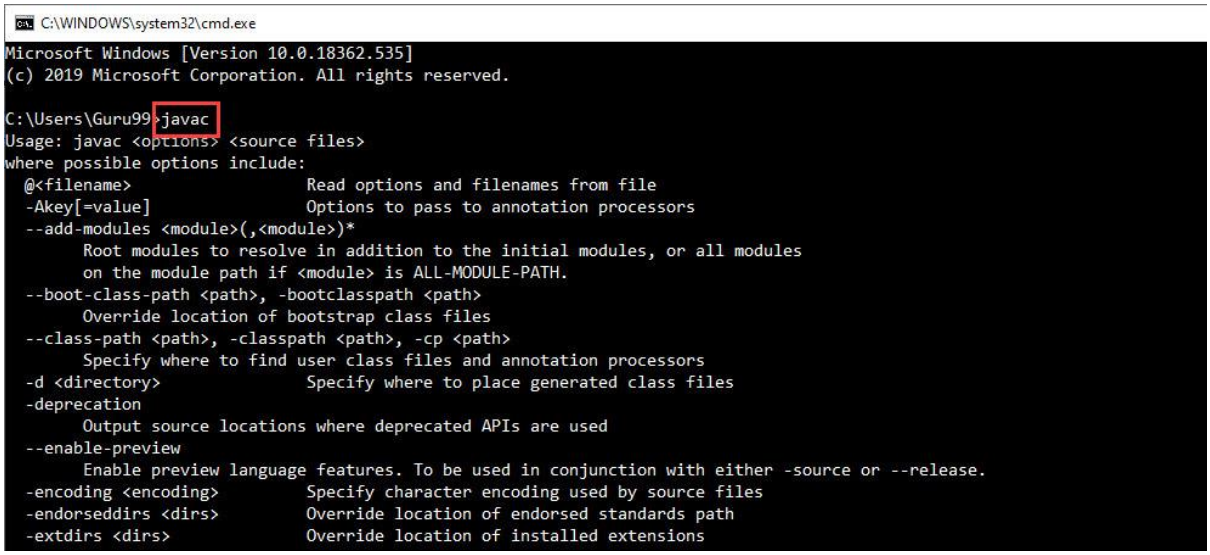
```
CLASSPATH = <JDK installation directory>\lib\tools.jar;
```

**Step 9)** Click on OK button




**Step 10)** Go to command prompt and type javac commands.

If you see a screen like below, Java is installed.




## Java Basic Input and Output

### Java Output

In Java, you can simply use

- System.out.println(); or
- System.out.print(); or
- System.out.printf();

to send output to standard output (screen).

Here,

- `System` is a class
- `out` is a `public` `static` field: it accepts output data.

  Don't worry if you don't understand it. We will discuss `class`, `public`, and `static` in later chapters.

Let's take an example to output a line.

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

        System.out.println("Java programming is interesting.");
    }
}
```

**Output**:

```
Java programming is interesting.
```

Here, we have used the `println()` method to display the string.

**Difference between println(), print() and printf()**

- `print()` - It prints string inside the quotes.
- `println()` - It prints string inside the quotes similar like `print()` method. Then the cursor moves to the beginning of the next line.
- `printf()` - It provides string formatting (similar to printf in C/C++ programming).

**Example: print() and println()**

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

        System.out.println("1. println ");
        System.out.println("2. println ");

        System.out.print("1. print ");
        System.out.print("2. print");
    }
}
```

**Output**:

```
1. println
2. println
```

```
1. print 2. print
```

In the above example, we have shown the working of the `print()` and `println()` methods. To learn about the `printf()` method, visit Java printf().

**Example: Printing Variables and Literals**

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

        Double number = -10.6;

        System.out.println(5);
        System.out.println(number);
    }
}
```

When you run the program, the output will be:

```
5
-10.6
```

Here, you can see that we have not used the quotation marks. It is because to display integers, variables and so on, we don't use quotation marks.

---

**Example: Print Concatenated Strings**

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

        Double number = -10.6;

        System.out.println("I am " + "awesome.");
        System.out.println("Number = " + number);
```

```
    }
}
```

**Output**:

```
I am awesome.
Number = -10.6
```

In the above example, notice the line,

```
System.out.println("I am " + "awesome.");
```

Here, we have used the `+` operator to concatenate (join) the two strings: `"I am "` and `"awesome."`.

And also, the line,

```
System.out.println("Number = " + number);
```

Here, first the value of variable `number` is evaluated. Then, the value is concatenated to the string: `"Number = "`.

---

**Java Input**

Java provides different ways to get input from the user. However, in this tutorial, you will learn to get input from user using the object of `Scanner` class.

In order to use the object of `Scanner`, we need to import `java.util.Scanner` package.

```
import java.util.Scanner;
```

To learn more about importing packages in Java, visit Java Import Packages.

Then, we need to create an object of the `Scanner` class. We can use the object to take input from the user.

```
// create an object of Scanner
Scanner input = new Scanner(System.in);

// take input from the user
int number = input.nextInt();
```

---

**Example: Get Integer Input From the User**

```
import java.util.Scanner;

class Input {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int number = input.nextInt();
        System.out.println("You entered " + number);

        // closing the scanner object
        input.close();
    }
}
```

**Output**:

```
Enter an integer: 23
You entered 23
```

In the above example, we have created an object named `input` of the `Scanner` class. We then call the `nextInt()` method of the `Scanner` class to get an integer input from the user.

Similarly, we can use `nextLong()`, `nextFloat()`, `nextDouble()`, and `next()` methods to get `long`, `float`, `double`, and `string` input respectively from the user.

**Note**: We have used the `close()` method to close the object. It is recommended to close the scanner object once the input is taken.

---

**Example: Get float, double and String Input**

```
import java.util.Scanner;

class Input {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        // Getting float input
        System.out.print("Enter float: ");
        float myFloat = input.nextFloat();
        System.out.println("Float entered = " + myFloat);

        // Getting double input
        System.out.print("Enter double: ");
        double myDouble = input.nextDouble();
        System.out.println("Double entered = " + myDouble);

        // Getting String input
        System.out.print("Enter text: ");
        String myString = input.next();
        System.out.println("Text entered = " + myString);
    }
}
```

**Output**:

```
Enter float: 2.343
Float entered = 2.343
Enter double: -23.4
Double entered = -23.4
```

```
Enter text: Hey!
Text entered = Hey!
```

As mentioned, there are other several ways to get input from the user. To learn more about Scanner, visit Java Scanner.

## Java "Hello, World!" Program

```
// Your First Program

class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

## Java program to find the sum of two numbers

```
public class AddTwoNumbers {

   public static void main(String[] args) {

      int num1 = 5, num2 = 15, sum;
      sum = num1 + num2;

      System.out.println("Sum of these numbers: "+sum);
   }
}
```