Introduction to Java

What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the Oak name to Java.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

Java Example

Let's have a quick look at Java programming example. A detailed description of Hello Java example is available in next page.

```java
public class Simple {

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

Application

According to Oracle, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

https://www.zdnet.com/article/log4j-zero-day-flaw-what-you-need-to-know-and-how-to-protect-yourself/

Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called enterprise application. It has advantages of the high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

Java Platforms / Editions

There are 4 platforms or editions of Java:

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

It is a micro platform which is mainly used to develop mobile applications.

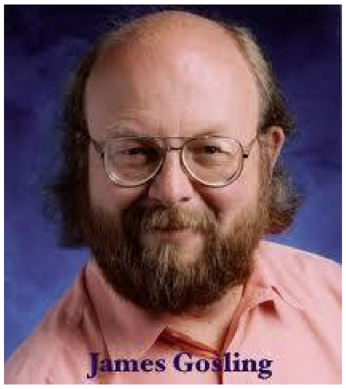
4) JavaFX

It is used to develop rich internet applications. It uses a light-weight user interface API.

History of Java

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. However, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". 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.



Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. There are given significant points that describe the history of Java.

1) **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
2) Initially designed for small, embedded systems in electronic appliances like set-top boxes.
3) Firstly, it was called **"Greentalk"** by James Gosling, and the file extension was .gt.
4) After that, it was called **Oak** and was developed as a part of the Green project.

Why Java named "Oak"?

5) **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.

6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Why Java Programming named "Java"?

7) **Why had they chosen java name for Java language?**
The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say. According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.

8) Java is an island of 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 coffee near his office.

9) Notice that Java is just a name, not an acronym.

10) Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 released in(January 23, 1996). After the first release of Java, there have been many additional features added to the language. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc. Each new version adds the new features in Java

Java Version History

Many java versions have been released till now. The current stable release of Java is Java SE 10.

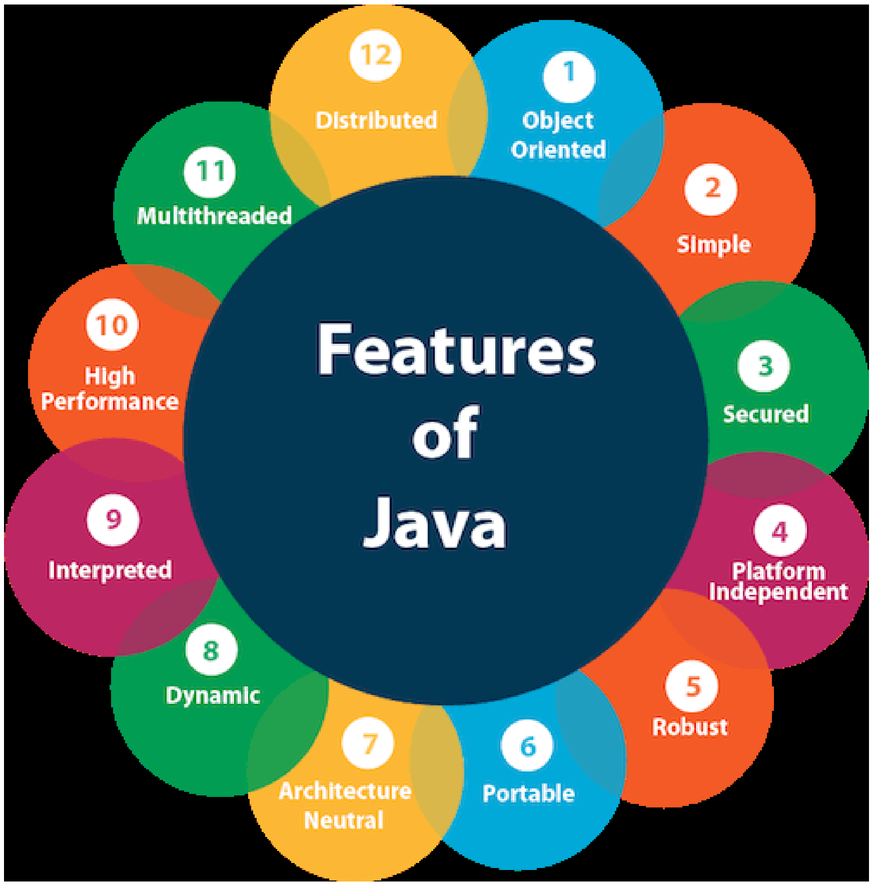
1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 LTS (18th Mar 2014)
11. Java SE 9 (21st Sep 2017)

12. Java SE 10 (20th Mar 2018)
13. Java SE 11 LTS (September 2018).
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)
16. Java SE 14 (March 2020)
17. Java SE 15 (September 2020)
18. Java SE 16 (March 2021)
19. Java SE 17 LTS (September 2021)
20. Java SE 18 (March 2022)

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*.

A list of most important features of Java language is given below.



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

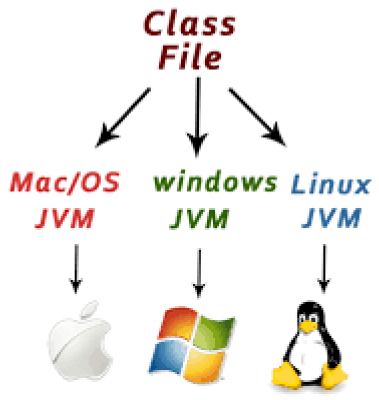
Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

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



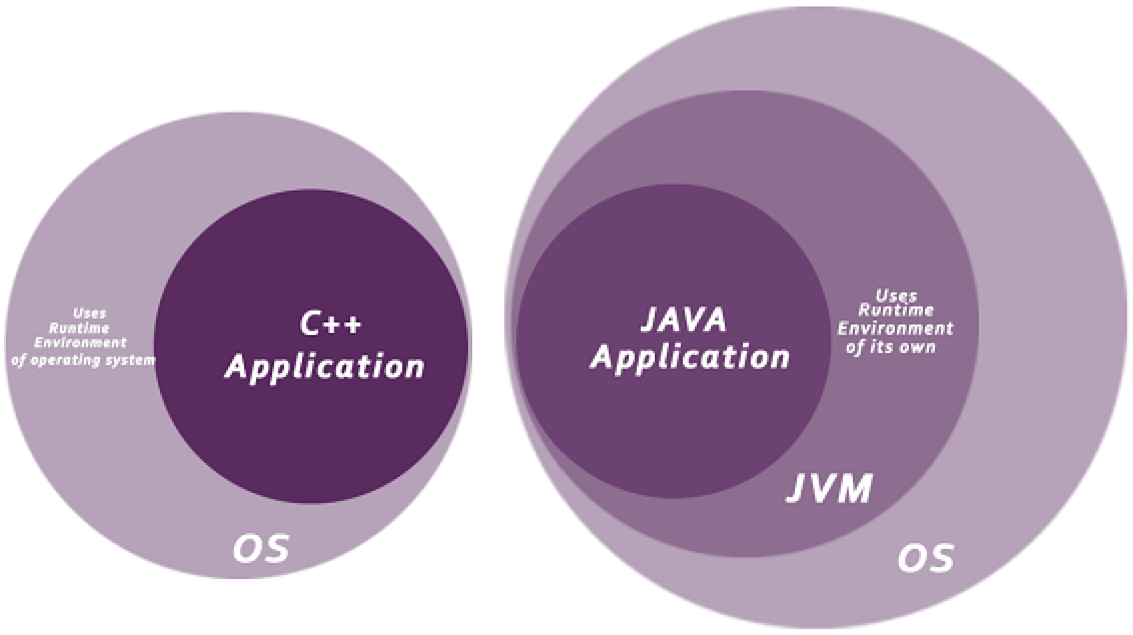
Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs. There are two types of platforms 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
2. API(Application Programming Interface)

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).


Uses Runtime Environment of operating system
C++ Application
OS
JAVA Application
Uses Runtime Environment of its own
JVM
OS

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- No explicit pointer
- Java Programs run inside a virtual machine sandbox

• **Classloader:** Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
• **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.
• **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

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.

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++.

Java supports dynamic compilation and automatic memory management (garbage collection).

Difference between JDK, JRE, and JVM

We must understand the differences between JDK, JRE, and JVM before proceeding further to Java. See the brief overview of JVM here.

JVM

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.

The JVM performs the following main tasks:

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

JRE

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 physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.

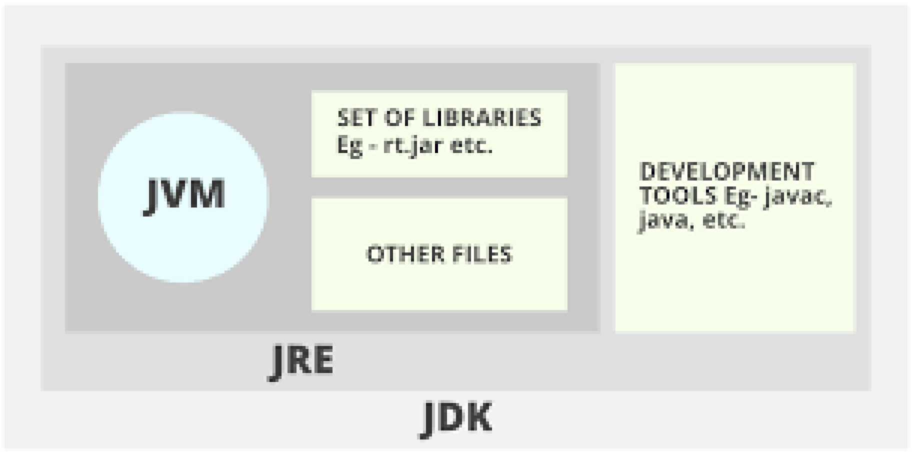
JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an
interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.


JVM
SET OF LIBRARIES Eg - rt.jar etc.
OTHER FILES
DEVELOPMENT TOOLS Eg- javac, java, etc.
JRE
JDK

How to set path in Java

The path is required to be set for using tools such as javac, java, etc.
If you are saving the Java source file inside the JDK/bin directory, the path is not required to be set because all the tools will be available in the current directory.
However, if you have your Java file outside the JDK/bin folder, it is necessary to set the path of JDK.
There are two ways to set the path in Java:

1. Temporary
2. Permanent

1) How to set the Temporary Path of JDK in Windows

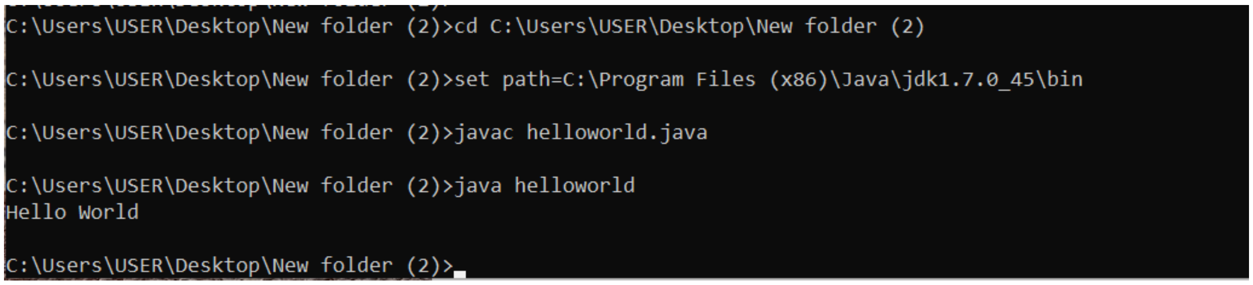
To set the temporary path of JDK, you need to follow the following steps:

- Open the command prompt
- Copy the path of the JDK/bin directory
- Write in command prompt: set path=copied_path

For Example:

set path=C:\Program Files(x86)\Java\jdk1.7.0_45\bin

Let's see it in the figure given below:



```
C:\Users\USER\Desktop\New folder (2)>cd C:\Users\USER\Desktop\New folder (2)

C:\Users\USER\Desktop\New folder (2)>set path=C:\Program Files (x86)\Java\jdk1.7.0_45\bin

C:\Users\USER\Desktop\New folder (2)>javac helloworld.java

C:\Users\USER\Desktop\New folder (2)>java helloworld
Hello World

C:\Users\USER\Desktop\New folder (2)>
```

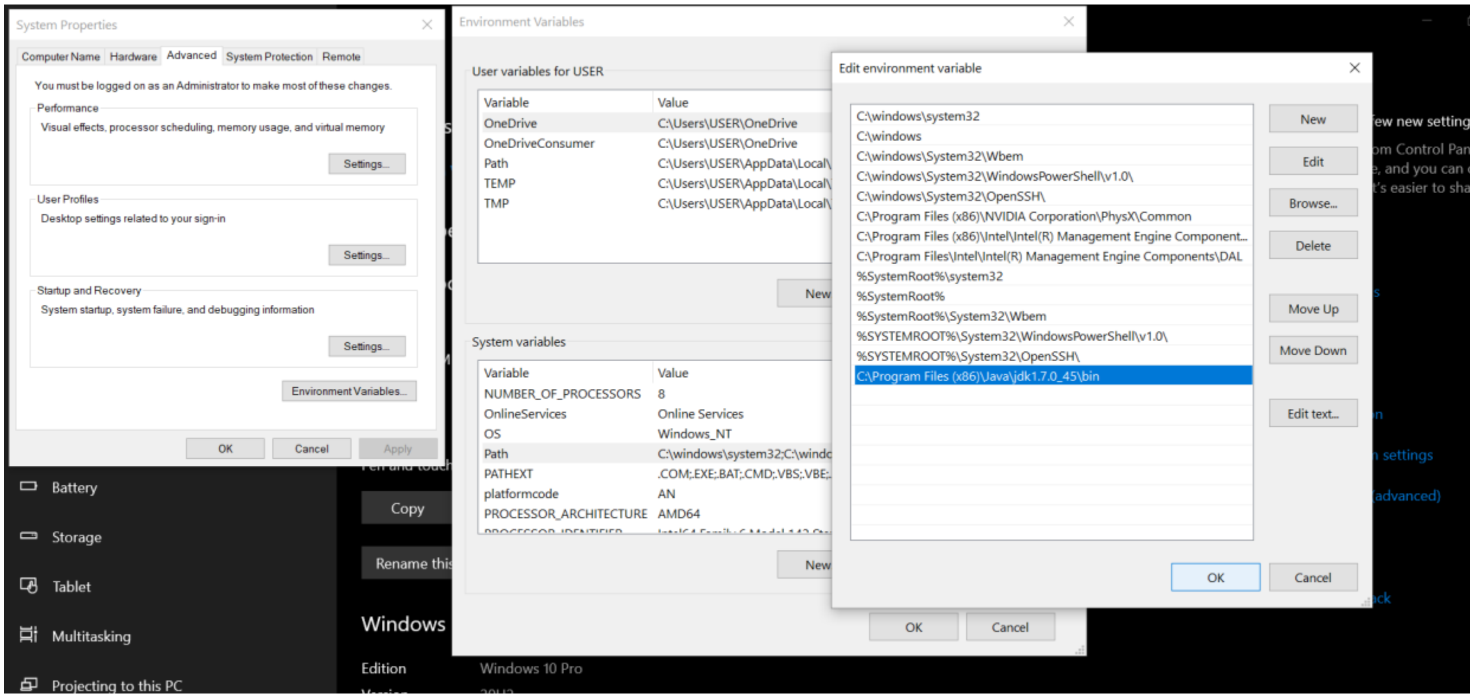
2) How to set Permanent Path of JDK in Windows

For setting the permanent path of JDK, you need to follow these steps:

• Copy the path of the JDK/bin directory

• Go to This Pc right click → Properties → Advance Settings → environment variables → System Variables →PathNew→ write path in variable name

For Example:



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

User variables for USER

Variable	Value
OneDrive	C:\Users\USER\OneDrive
OneDriveConsumer	C:\Users\USER\OneDrive
Path	C:\Users\USER\AppData\Local\
TEMP	C:\Users\USER\AppData\Local\
TMP	C:\Users\USER\AppData\Local\

System variables

Variable	Value
NUMBER_OF_PROCESSORS	8
OnlineServices	Online Services
OS	Windows_NT
Path	C:\windows\system32;C:\windc
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.
platformcode	AN
PROCESSOR_ARCHITECTURE	AMD64

Edit environment variable

```
C:\windows\system32
C:\windows
C:\windows\System32\Wbem
C:\windows\System32\WindowsPowerShell\v1.0\
C:\windows\System32\OpenSSH\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files (x86)\Java\jdk1.7.0_45\bin
```

First Java Program | Hello World Example

In this page, we will learn how to write the simple program of java. We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains the main method. Let's understand the requirement first.

The requirement for Java Hello World Example

For executing any java program, you need to

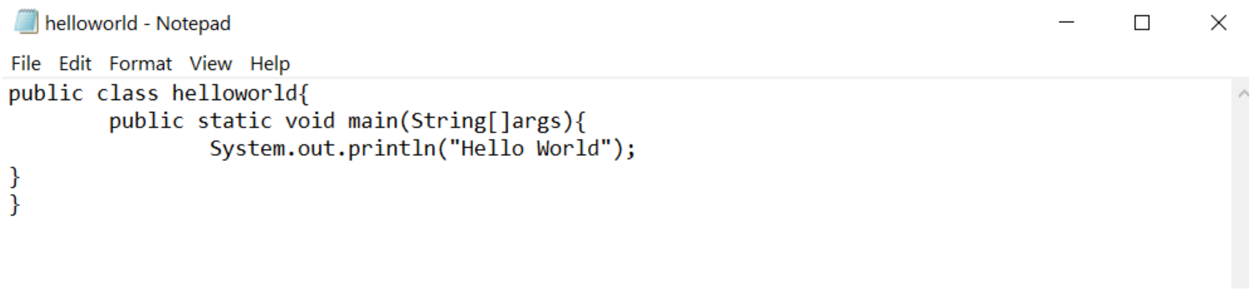
• Install the JDK if you don't have installed it, download the JDK and install it.

• Set path of the jdk/bin directory

• Create the java program

• Compile and run the java program

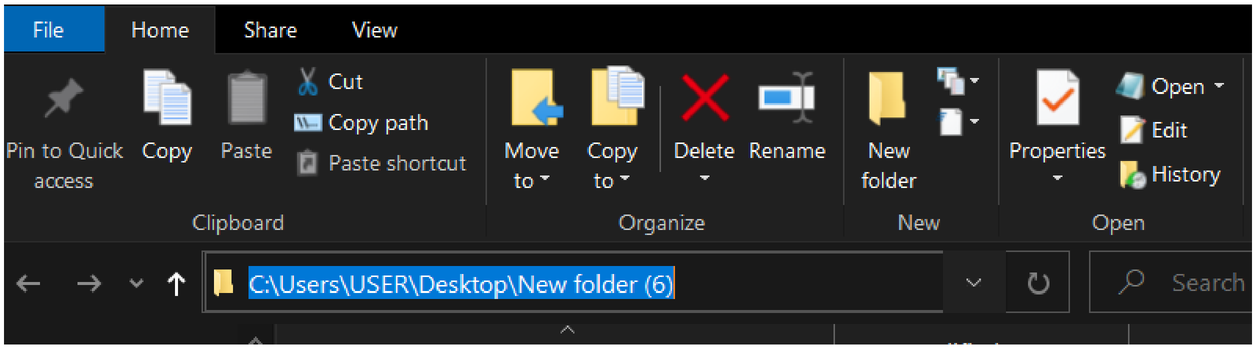
Creating Hello World Example
JAVA using Text document



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

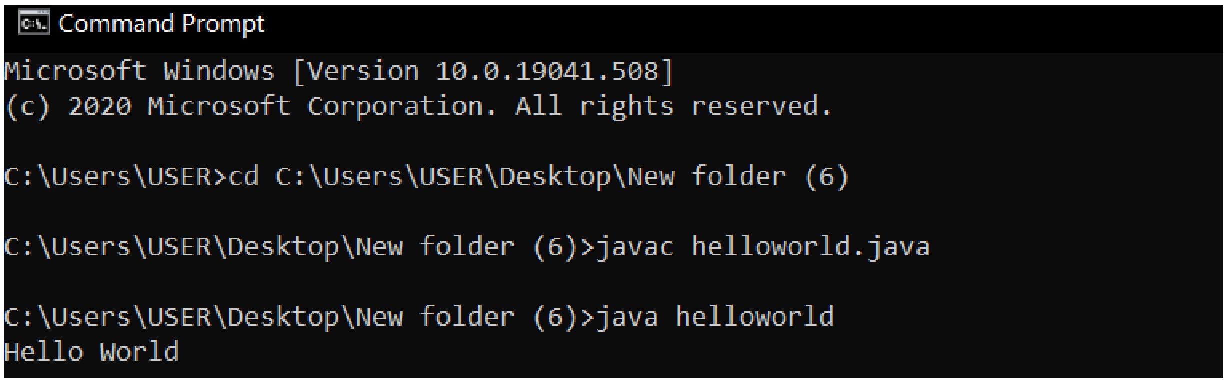
How to run the program,

1. Get the save location



C:\Users\USER\Desktop\New folder (6)

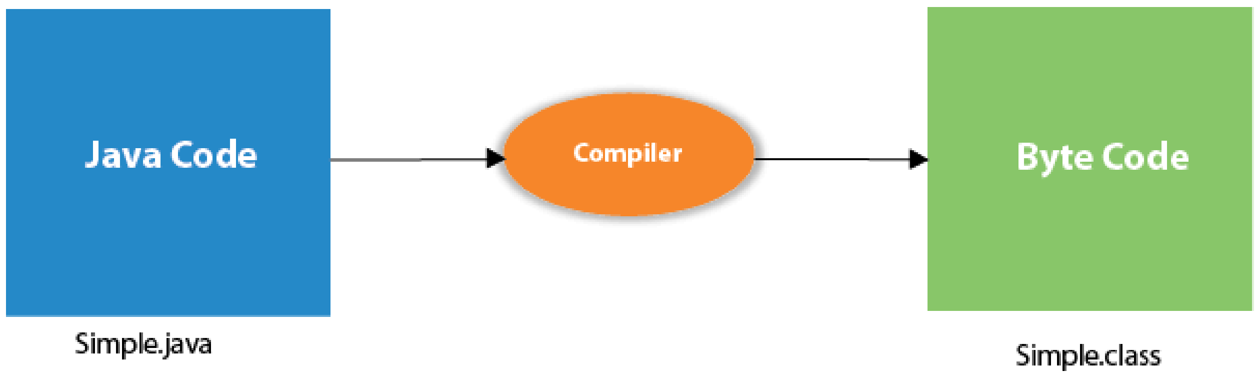
2. Change the path from the command prompt.(using cd command)
3. Then type javac filename.java (helloworld.java)
4. Finally type java filename (helloworld)



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\USER>cd C:\Users\USER\Desktop\New folder (6)

C:\Users\USER\Desktop\New folder (6)>javac helloworld.java

C:\Users\USER\Desktop\New folder (6)>java helloworld
Hello World
```

Compilation Flow:

When we compile Java program using javac tool, java compiler converts the source code into byte code.



Parameters used in First Java Program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().
• **class** keyword is used to declare a class in java.
• **public** keyword is an access modifier which represents visibility. It means it is visible to all.
• **static** is a keyword. If we declare any method as static, it is known as the static method. The core
advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
• **void** is the return type of the method. It means it doesn't return any value.
• **main** represents the starting point of the program.
• **String[] args** is used for command line argument. We will learn it later.
• **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println() is the method of PrintStream class. We will learn about the internal working of System.out.println statement later.

To compile and run the above program, go to your current directory first; my current directory is

c:\new.

Write here:
To compile: javac Simple.java
To execute: java Simple

How many ways can we write a Java program,

There are many ways to write a Java program. The modifications that can be done in a Java program are given below:

1) By changing the sequence of the modifiers, method prototype is not changed in Java.

Let's see the simple code of the main method.

1. static public void main(String args[])

2) The subscript notation in Java array can be used after type, before the variable or after the variable.

Let's see the different codes to write the main method.

1. public static void main(String[] args)
2. public static void main(String []args)
3. public static void main(String args[])

3) You can provide var-args support to the main method by passing 3 ellipses (dots)

Let's see the simple code of using var-args in the main method. We will learn about var-args later in Java New Features chapter.

1. public static void main(String... args)

Valid java main method signature

1. public static void main(String[] args)
2. public static void main(String []args)
3. public static void main(String args[])
4. public static void main(String... args)
5. static public void main(String[] args)
6. public static final void main(String[] args)
7. final public static void main(String[] args)
8. final strictfp public static void main(String[] args)

Invalid java main method signature

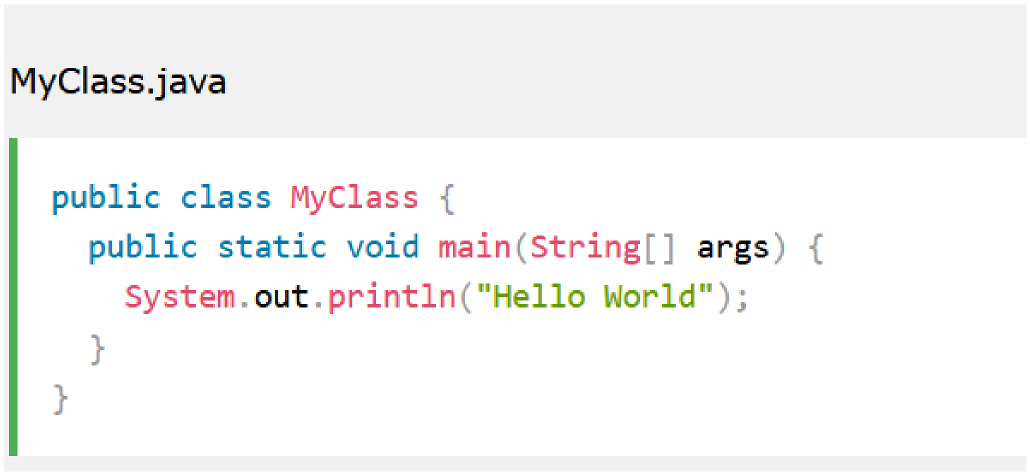
1. public void main(String[] args)
2. static void main(String[] args)
3. public void static main(String[] args)
4. abstract public static void main(String[] args)

Resolving an error "javac is not recognized as an internal or external command"?

If there occurs a problem like displayed in the below figure, you need to set path. Since DOS doesn't know javac or java, we need to set path. The path is not required in such a case if you save your program inside the JDK/bin directory. However, it is an excellent approach to set the path.

Java Syntax

• In the previous chapter, we created a Java file called Helloworld.java, and we used the following code to print "Hello World" to the screen:



MyClass.java

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

Example explained

• Every line of code that runs in Java must be inside a class. In our example, we named the class **MyClass**. A class should start with an uppercase first letter.
• **Note:** Java is case-sensitive: "MyClass" and "myclass" has different meaning.
• The name of the java file **must match** the class name. When saving the file, save it using the class name and add ".java" to the end of the filename. To run the example above on your computer, make sure that Java is properly installed: Go to the Get Started Chapter for how to install Java. The output should be:

The main Method

- The main() method is required and you will see it in every Java program:

public static void **main**(String[] args)

• Any code inside the main() method will be executed. You don't have to understand thekeywords before and after main. You will get to know them bit by bit while reading this tutorial.
• For now, just remember that every Java program has a class name which must match the filename, and that every program must contain the main() method.

System.out.println()

• Inside the main() method, we can use the println() method to print a line of text to the screen: