First Simple Java Program Hello World

In this tutorial, we will learn how to write the first simple program in Java. Writing a simple program in Java is very easy as in other languages.

We need to create a class that contains the main method of valid signature which acts as an entry point for the program.

So, we will start with a very simple program that will print a line of text ("Hello Java") as output.

Before going to create a class, we will understand the requirements first.

Requirements for Simple Java Program

Following are four steps needed for creating and executing any program in java.

1. Install the JDK. If you don't have installed it, download the JDK and install it. Go to this tutorial: Download and Install JDK.
2. Set the path of the jdk/bin directory.
3. Writing the java program.
4. Compiling java program.
5. Running bytecode on JVM.

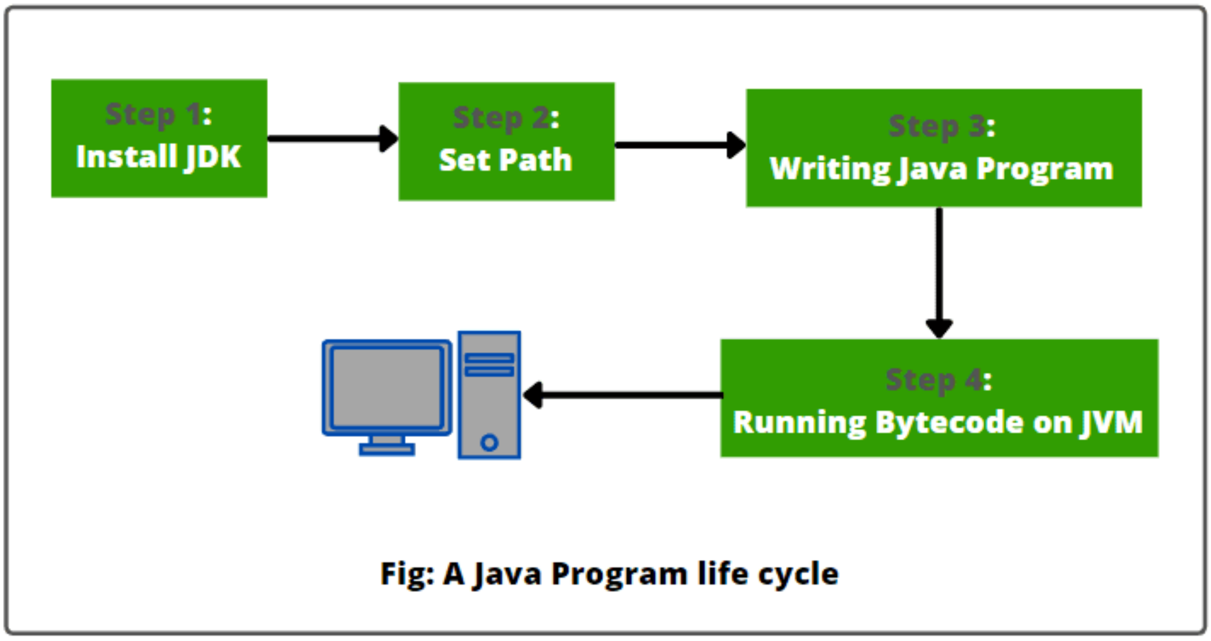


Fig: A Java Program life cycle

Simple Java Hello World Example

Let’s write a HelloWorld Program. To write a simple java program, we can use any text editor like Notepad on Windows and TextEdit on Mac. Now, open a text editor and write a simple program.

A simple Java program looks like as shown below. It contains one class that contains a main() method.

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

Save this file as Hello.java

A Java program must be saved with .java extension. The extension is case-sensitive. It is compiled using the Java compiler (javac.exe) and executed using Java interpreter (java.exe).

If Java program does not have valid extension (.java), Java compiler generates an error on compiling.

For example, suppose the above program is saved with .java extension then the compiler will generate an error because the file extension is case sensitive. A valid extension for Java program is .java.

Note: File name must be the same as the program name and the extension must be .java. This file is called source file or source code. Source code is Java code written by a programmer.

Now the next step is to compile Java program. So, let’s understand it step by step.

How to Compile Java Program?

To compile the above program, we need to call Java compiler to translate (converts) program source code into Java bytecode.

For this, open **Command prompt** (cmd) on Windows. If you are using Mac OS then open Terminal.

To compile the above program, write the following command and press enter.

> To compile: javac Hello.java

You may get this error while compiling this program: “**javac’ is not recognized as an internal or external command, operable program or batch file**“.

Generally, this error happens when the java path is not set properly in your system. If you get this error then you first set the path properly before going to compile the above program.

If everything is fine, the javac compiler will create a file called **Hello.class** on your disk. This file contains the byte code of the program.

The compiled java program source code is called bytecode. Java compiler automatically names the byte code file.

Note: If the program contains syntax errors then it will not compile. A syntax error generally occurs in a statement when we violate the rules of Java.

For example, all the java statements must end with a semicolon. A statement without a semicolon produces a syntax error during the compilation of program and prevents the compiler from generating bytecode for the program.

Now the next step is to run (execute) the Java program.

How to Run Java Program?

In the final step, we can run the program. For this, we need to use Java interpreter to execute a standalone program. At the command prompt, type the following command and press the enter key.

> To execute: java Hello

Now, Java interpreter will search the main method in the application program and starts the execution process from there.

When the program is executed in JVM, it will display the following line of text on your computer screen.

> Output:
>   Hello Java

Process of Java Program Compilation and Execution

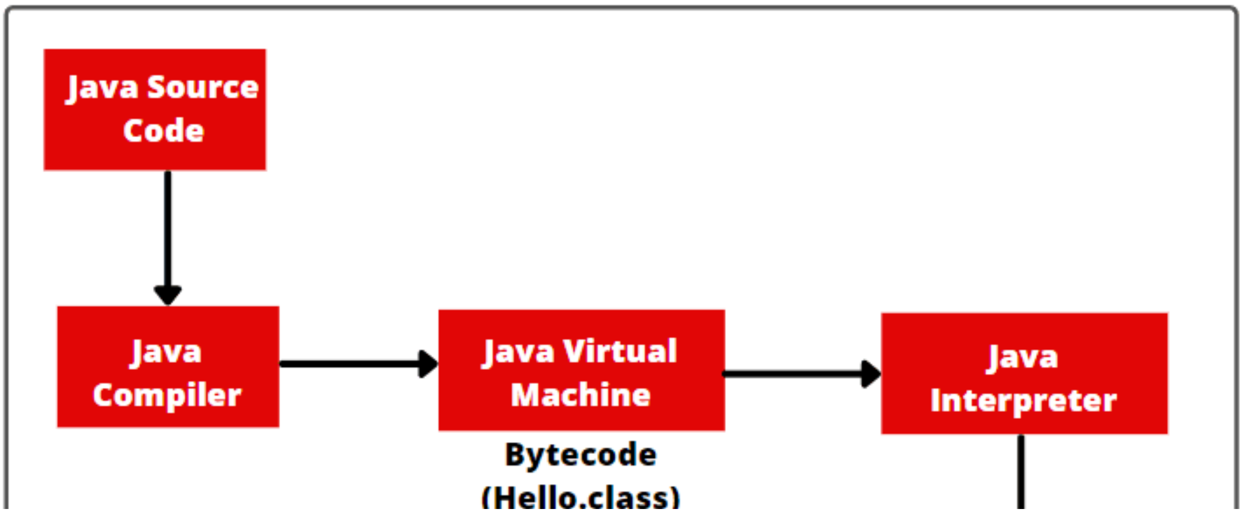
1. When we compile source code (Hello.java file) using javac compiler, Java compiler automatically generates bytecode (Hello.class) on the local disk.

A byte code is a binary code that is understood by Java interpreter with the help of JVM.

2. Java Interpreter takes byte code as an input and executes that code by converting it to native code with the underlying operating system. In simple words, byte code is processed by Java interpreter.

Since this bytecode is machine-independent, it can be run on any computer machine. In other words, a program compiled on one machine can be run on another machine.

Look at the compilation and execution flow diagram below.



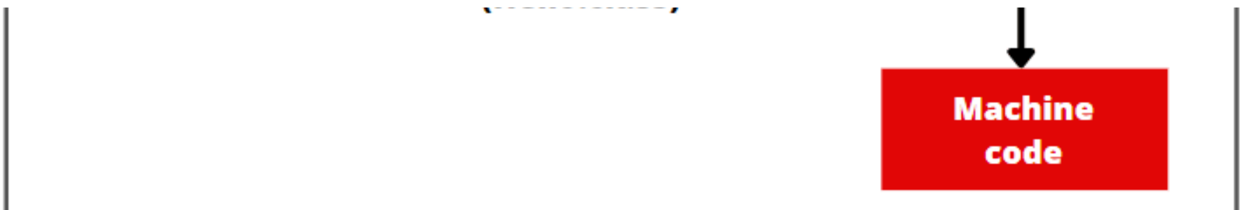


Fig: Java Translation and Execution Process

Important Statements used in Hello Program

Perhaps, the above program is one of the simplest of all Java programs. Let’s understand the statements of program line by line and the unique features provided by Java language.

1. Class Declaration:

The first line is

> public class Hello

This statement declares a class named Hello. Here, class is a keyword in Java that is used to define a class.

Hello is a Java identifier that represents the name of class to be defined. Since Java is a true object-oriented programming language, everything must be placed inside a class.

2. Opening Brace:

Every class definition starts with an opening brace ” { ” and ends with a matching closing brace ” { “. It is just like to C++ class construct. Remember that a class definition in C++ ends with a semicolon.

3. Main method (Main line):

The third line is

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

This statement defines the main() method. It is automatically run when the program is executed. It is similar to the main() function in C/C++.

Every Java program must have the main() method. It is the starting point for the interpreter to start the execution process of java program.

A Java program may contain any number of classes but only one of them must contain the main method to begin the execution.

This statement contains a number of keywords such as public, static, and void.

a) **public:** The keyword public is an access modifier that represents visibility. It means it is accessible to all other classes.

b) **static:** It is a keyword. If we define a method as static, it is called static method. The main advantage of the static method is that there is no need to create an object to call the static method.

The main method must always be declared as static because JVM executes the static method before the object creation. It means that JVM doesn't need to create an object to class the main method. Thus, it also saves memory.

c) **void:** It is a modifier that states that the main method does not return any value.

d) **main:** It represents the starting point for the execution of the program.

e) **String[] args:** It contains an array of objects of class type String. It is used for command-line arguments.

If you try out to make the main method upper case, Java compiler will generate compile-time error.

The valid “main” methods that can be used in a Java program are

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

At a time, only one of the above “main” method can be used in a class.

4. Output Line:

Inside the main method, the only executable statement is

```
System.out.println("Hello Java");
```

The println() method is used to print (display) “Hello World” on the console. It is similar to the print() statement of C or court << construct of C++.

Since Java is a pure object-oriented programming language, every method is a part of an object. The println() method is a part of out object, that is a static data member of System class.

The statement prints the string: “Hello World” to the screen.

5. Semicolon:

A program is composed of a set of instructions that is called statements. A semicolon is needed to indicate the end of a statement. Every Java statement must end with a semicolon.