

# CORE JAVA

In the previous chapters we installed JDK on our local machine and we also discussed the basic CMD commands. Then we learnt about Variables, keywords, datatypes and operators.

## First Java Program (Hello world)-

In this chapter, we are going to create our first java program. We will start learning about theoretical concepts from beginner level to advanced level from the next chapter.

Here, we will perform a practical demonstration to display "Hello World." After that, we will review the concepts used in this demonstration. So, let's get started.

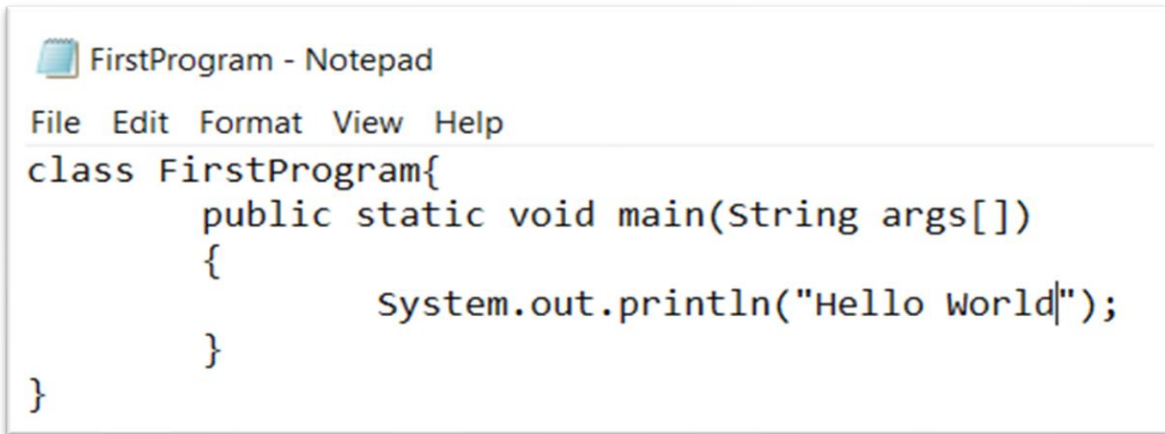
To compile and run any Java program, we need to have Java installed on our system. While you can use an online compiler, it is a good practice to install Java on your local machine.

To check java is properly installed in your local machine, open CMD and check Java version-

```
C:\Users\LENOVO\Documents\java_practical>java -version
java version "21.0.1" 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)
```

We are going to use notepad and CMD to develop, compile and run our Java program.

**Open notepad and write the following code-**



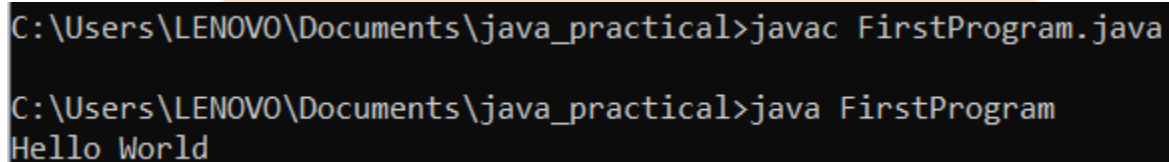
```
FirstProgram - Notepad
File Edit Format View Help
class FirstProgram{
    public static void main(String args[])
    {
        System.out.println("Hello World!");
    }
}
```

Save the above file with the same name as class name-

**FirstProgram.java**

HT EASY

**CMD- Compile and run our first program-**



```
C:\Users\LENOVO\Documents\java_practical>javac FirstProgram.java
C:\Users\LENOVO\Documents\java_practical>java FirstProgram
Hello World
```

To compile a java program we use command-

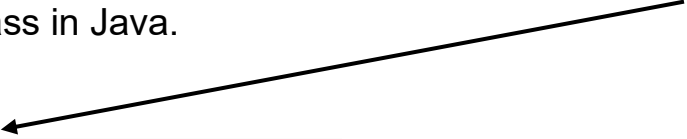
**javac name\_of\_the\_file.java**

To run a java program we use the command-

**java name\_of\_the\_file**

## Now let's understand what is happening in the above program.

1. We opened notepad to create our first java program.
2. We created a class named FirstProgram. class keyword is used to create a class in Java.




```
class FirstProgram{
```

Remember Java is case-sensitive. So, Class and class is considered as different variables. And class is a keyword in java.

The syntax to create a class is-

```
class name of the class{  
    // Body of the class  
}
```

3. Inside the class we created a function named main-



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

**public:** public is an access specifier/modifier that represents visibility. An access specifier is used to define the visibility of the variable or function it is

used with. Here, we used public that means the main function is accessible by all, within the class, outside of the class, and by other packages also.

**static:** static is a keyword in java. When we define a method with keyword static that means that method can be directly invoked without creating an object (instance of a class).

Basically, static method belongs to the class rather than the object.

The main method is the entry point for any Java application. When the Java Virtual Machine (JVM) starts, it looks for the main method to begin execution. Since no objects are created at the start, the JVM needs a way to call this method without any instance, hence it must be static.

**void:** This is the return type of the main() method. void means that the function does not return any value.

**main:** This is the name of the method. It is a predefined name in Java, and the JVM looks for this method signature to start the execution of a Java program.

A method is created with name\_of\_the\_method() and ends with brackets.

We will learn about methods later in our Java course series.

**NOTE:** In every java program the first thing that is executed when we run a java program is the main() method.

**String args[]:** This is the parameter of the main method, which is an array of String objects. It allows the method to accept a set of command-line arguments. We will learn about this later in detail.

Now every block of code is written within the {} curly braces.

#### 4. `System.out.println("Hello World");` :

In Java every statement ends with a ; (semicolon). In the above statement, `System` is a class, `out` is an object and `println()` is a method.

And inside the () brackets we provided string to be printed as it is on our screen. So, here we basically printed "Hello World". You can change the string to print something else.

In summary, `System.out.println("Hello World");` instructs the JVM to use the standard output stream (typically the console) to print the text "Hello World" followed by a new line.

The `println` method prints the specified string, followed by a new line.



#### 5. What happens when we compile our Java program-

##### Command to compile java code-

```
javac file_name.java
```

When you compile Java code, a series of processes transform your human-readable source code into bytecode that the Java Virtual Machine (JVM) can execute.

Everytime when you compile a java code, the compiler generates Java bytecode, which is a platform-independent code that the JVM can interpret. This bytecode is stored in .class files. For example, compiling `FirstProgram.java` creates a `FirstProgram.class` file containing the bytecode.

 FirstProgram.class	7/14/2024 4:15 PM	CLASS File	1 KB
 FirstProgram	7/14/2024 4:24 PM	Java Source File	1 KB

#### 6. To run Java program use the command-

```
java name_of_file
```

## Why class name and file name should be same?

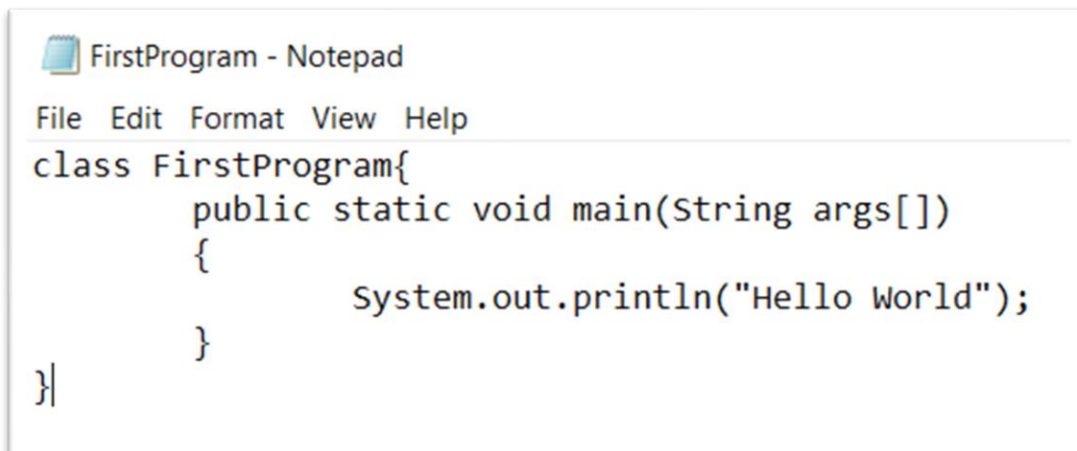
According to the Java Language Specification, if a class is declared as public, the name of the source file must match the name of the public class. For example, if you have a public class named HelloWorld, it must be in a file named HelloWorld.java.

The Java compiler (javac) expects this naming convention. If the file name doesn't match the public class name, the compiler will throw an error. This ensures consistency and helps avoid confusion during the compilation process.

Basically, in case of multiple classes the class that contain the main() method should be of the same name as the file name.

## Can we run a Java Program without creating a class?

No, you cannot run a traditional Java program without creating a class, as Java is fundamentally an object-oriented programming language where everything is encapsulated within classes. The main method, which serves as the entry point of the program, must be defined within a class.



```
FirstProgram - Notepad
File Edit Format View Help
class FirstProgram{
    public static void main(String args[])
    {
        System.out.println("Hello world");
    }
}
```

## Reasons for Needing a Class-

- **Object-Oriented Paradigm:** Java is designed around the concept of classes and objects. Everything in Java is either a class or part of a class.
- **Entry Point:** The main method, which serves as the starting point for program execution, must be placed inside a class.

The basic syntax of Java Programming Language, that we will use in every program-

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

## Default Imported Package

In every Java program, the java.lang package is automatically imported. This package contains fundamental classes that are essential for the Java programming language. Some commonly used classes in java.lang include:

- java.lang.Object
- java.lang.String
- java.lang.System

- `java.lang.Math`
- `java.lang.Integer`
- `java.lang.Thread`



**Join our growing community of tech enthusiasts! Subscribe to our YouTube channel, where we simplify programming languages in the easiest way possible. Gain a deeper understanding with our clear explanations and receive exclusive notes to enhance your learning journey. Don't miss out on valuable insights – hit that subscribe button now and embark on a programming adventure with us!**

**Subscribe to our channel:**  
**<https://www.youtube.com/@HTEASYLEARNING>**