Writing First Program and Basic Input/Output in Java

**First Program - "Hello World"**

The below-given program is the simplest program of Java, printing "Hello World" to the screen. Let us try to understand every bit of the code step by step.

Java

**import** **java.io.**\*;

// This **is** a simple Java program.

**class** **HelloWorld**

{

// Your program begins **with** a call to main().

// Prints "Hello, World" to the terminal window.

public static void main(String args[])

{

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

}

}

**Output**:

Hello, World

The "Hello World!" program consists of three primary components: the HelloWorld class definition, the main method, and the source code comments. The following explanation will provide you with a basic understanding of the code:

1. **Class definition:**This line uses the keyword **class**to declare that a new class is being defined.

class HelloWorld

1. **HelloWorld** is an identifier that is the name of the class. The entire class definition, including all of its members, will be between the opening curly brace  **{**  and the closing curly brace**}** .
2. **main method:**In Java programming language, every application must contain a main method whose signature is:

public static void main(String[] args)

**public**: So that [JVM](https://www.cdn.geeksforgeeks.org/jvm-works-jvm-architecture/) can execute the method from anywhere.

**static**: Main method is to be called without object.

The modifiers public and static can be written in either order.

**void**: The main method doesn't return anything.

**main()**: Name configured in the [JVM](https://www.cdn.geeksforgeeks.org/jvm-works-jvm-architecture/).

**String[]**: The main method accepts a single argument:

an array of elements of type String.

1. Like in C/C++, main method is the entry point for your application and will subsequently invoke all the other methods required by your program.
2. The next line of code is shown here. Notice that it occurs inside main( ).

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

1. This line outputs the string "Hello, World" followed by a new line on the screen. Output is actually accomplished by the built-in println(*)* method. **System** is a predefined class that provides access to the system, and **out** is the variable of type output stream that is connected to the console.
2. Comments: They can either be multi-line or single line comments.

/\* This is a simple Java program.

Call this file "HelloWorld.java". \*/

1. This is a multiline comment. This type of comment must begin with /\* and end with \*/. For single line you may directly use // as in C/C++.

**Important Points**:

* The name of the class defined by the program is HelloWorld, which is the same as the name of the file(*HelloWorld.java*). This is not a coincidence. In Java, all codes must reside inside a class and there is at most one public class which contains the main() method.
* By convention, the name of the main class(a class which contain the main method) should match the name of the file that holds the program.

**Java Basic Console Input/Output**

In Java, there are three different ways of reading input from the user in the command line environment(console).

**1.Using Buffered Reader Class**

This is the classical Java method to take input, Introduced in JDK1.0. This method is used by wrapping the System.in (standard input stream) in an InputStreamReader which is wrapped in a BufferedReader. We can read input from the user in the command line.

* The input is buffered for efficient reading.

* The wrapping code is hard to remember.

**Program:**

Java

// Java program to demonstrate BufferedReader

**import** **java.io.BufferedReader**;

**import** **java.io.IOException**;

**import** **java.io.InputStreamReader**;

public **class** **Test**

{

public static void main(String[] args) throws IOException

{

//Enter data using BufferReader

BufferedReader reader =

new BufferedReader(new InputStreamReader(System.**in**));

// Reading data using readLine

String name = reader.readLine();

// Printing the read line

System.out.println(name);

}

}

Input: 

Geek

Output:

Geek

Note: To read other types, we use functions such as Integer.parseInt(), Double.parseDouble(). To read multiple values, we use split().

**2. Using Scanner Class**

This is probably the most preferred method to take input. The main purpose of the Scanner class is to parse primitive types and strings using regular expressions; however, it can also be used to read input from the user in the command line.

* Convenient methods for parsing primitives (nextInt(), nextFloat(), …) from the tokenized input.
* Regular expressions can be used to find tokens.

* The reading methods are not synchronized.

To see more differences, please see [this](https://www.cdn.geeksforgeeks.org/difference-between-scanner-and-bufferreader-class-in-java/) article.

Java

// Java program to demonstrate working of Scanner **in** Java

**import** **java.util.Scanner**;

**class** **GetInputFromUser**

{

public static void main(String args[])

{

// Using Scanner **for** Getting Input **from** **User**

Scanner **in** = new Scanner(System.**in**);

String s = **in**.nextLine();

System.out.println("You entered string "+s);

int a = **in**.nextInt();

System.out.println("You entered integer "+a);

float b = **in**.nextFloat();

System.out.println("You entered float "+b);

}

}

Input:

GeeksforGeeks

12

3.4

Output:

You entered string GeeksforGeeks

You entered integer 12

You entered float 3.4

**3. Using Console Class**

It has been becoming a preferred way for reading user’s input from the command line. In addition, it can be used for reading password-like input without echoing the characters entered by the user; the format string syntax can also be used (like System.out.printf()).  
**Advantages:**

* Reading password without echoing the entered characters.
* Reading methods are synchronized.
* Format string syntax can be used.

* Does not work in non-interactive environment (such as in an IDE).

Java

// Java program to demonstrate working of System.console()

// Note that this program does **not** work on IDEs **as**

// System.console() may require console

public **class** **Sample**

{

public static void main(String[] args)

{

// Using Console to input data **from** **user**

String name = System.console().readLine();

System.out.println(name);

}

}