**First Java Program in Notepad**

public class HelloWorld {

public static void main(String[] args) {

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

}

}

**📌 Step-by-Step Explanation (Every Word)**

**🔹 public**

* **Meaning**: Access modifier.
* **Purpose**: Makes the class or method **accessible to everyone** (i.e., the JVM can access it from anywhere).
* **In Context**: JVM needs to access main() method from outside, so it must be public.

**🔹 class**

* **Meaning**: Blueprint or template to create objects or programs.
* **Purpose**: Everything in Java resides inside a class.
* **In Context**: We are creating a class named HelloWorld.

**🔹 HelloWorld**

* **Meaning**: The **name of the class**.
* **Rule**: Should start with a **capital letter** (by convention).
* **In Notepad**: The filename must be **HelloWorld.java** (case-sensitive on some systems).

**🔹 { ... }**

* **Meaning**: Curly braces define the **body of the class or method**.
* **Use**: Everything inside { } is what the class or method will do.

**🔽 Inside the Class**

**🔹 public**

* Same as above — **needed** so the JVM can run the main() method.

**🔹 static**

* **Meaning**: Belongs to the class, not objects.
* **Why?**: JVM runs main() without creating an object of the class.
* **In Simple Terms**: We don’t create objects here, so static lets JVM call it directly.

**🔹 void**

* **Meaning**: No return value.
* **In Context**: The main() method doesn’t return anything to the JVM — it just runs.

**🔹 main**

* **Meaning**: Entry point of a Java program.
* **Rule**: JVM looks for this exact method name to start execution.

**🔹 (String[] args)**

* **Meaning**: It’s a parameter — an array of Strings.
* **Use**: It allows passing **command-line arguments** when the program starts.
* **In Real Life**: You can run your Java program with extra input like:

bash

CopyEdit

java HelloWorld Hello Siri

Then args[0] = "Hello" and args[1] = "Siri".

**🔸 Inside main() Method**

**🔹 System.out.println("Hello, World!");**

Let’s break it down:

1. **System**: A built-in Java class (in java.lang package).
2. **out**: A static object of PrintStream — used to send output to console.
3. **println**: Method that prints a line of text and moves to next line.
4. **"Hello, World!"**: The actual text being printed.
5. **;**: Every Java statement ends with a semicolon.

**✍️ Writing and Running in Notepad**

**📂 Steps:**

1. Open **Notepad**
2. Paste the code:

public class HelloWorld {

public static void main(String[] args) {

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

}

}

1. Save as **HelloWorld.java**
   * Make sure it’s **not** HelloWorld.java.txt
   * Use **Save as type: All Files**
2. Open **Command Prompt**
3. Compile:

javac HelloWorld.java

✅ This creates HelloWorld.class

1. Run:

java HelloWorld

✅ Output: Hello, World!

**🧠 Summary (What to Teach Students)**

| **Keyword** | **Meaning** | **Why it’s Important** |
| --- | --- | --- |
| public | Accessible from anywhere | So JVM can run it |
| static | Doesn’t need object | Called directly by JVM |
| void | No return | Just executes |
| main | Starting point | JVM looks for this method |
| String[] args | Command-line inputs | Useful for real-world programs |
| System.out.println() | Output command | Displays result on screen |

**THE main() METHOD IN JAVA**

public static void main(String[] args) {

System.out.println("Hello, world!");

}

This is the entry point of every Java application.

**✅ Why Do You Need main()?**

The Java Virtual Machine (JVM) starts execution from the main() method.

If main() is missing, the program won’t run unless you're writing:

A class used by another program (e.g., library)

A Java applet (obsolete now)

A Java Servlet (runs on web server)

A class with a static block (runs during class loading, not ideal for main logic)

Now, let's break it word by word:

🔷 1. **public**

✅ What it means:

It’s an access modifier.

public means the method is visible to the entire program and JVM can access it from anywhere.

**⚠️ What happens if you remove it?**

static void main(String[] args)

❌ Error at runtime:

Main method not found in class Hello

JVM looks specifically for a public main method.

🔷 2. **static**

✅ What it means:

static means the method belongs to the class, not to an object. JVM can call it without creating an object.

Think of it like a global method inside a class.

**⚠️ What happens if you remove static?**

public void main(String[] args)

❌ You’ll get:

Main method not found in class Hello

JVM does not create objects of your class to call main(). So main() must be static.

🔷 3. **void**

✅ What it means:

It tells the JVM that main() does not return anything.

It’s a return type.

If it returned something like int, JVM wouldn’t know what to do with it.

**What if you return something?**

public static int main(String[] args) {

return 0;

}

❌ JVM gives:

Error: Main method must return void

✅ So: void is mandatory for the JVM-recognized main().

🔷 4. **main**

✅ What it is:

The name main is fixed and predefined.

JVM looks only for a method named main.

**⚠️ Can you rename it?**

public static void start(String[] args)

❌ No. It compiles, but gives: Main method not found

✅ JVM will only run methods named exactly main.

**🔷 5. String[] args**

✅ What it means:

This is an array of String values.

Used to receive command-line arguments.

Example:

If you run:

java Hello HelloWorld GPT

Then inside main():

args[0] = "HelloWorld"

args[1] = "GPT"

**⚠️ What if you remove it?**

**public static void main()**

**❌ Compiles, but:**

Main method not found in class Hello

✅ JVM requires main(String[] args) — even if you don’t use args.

🧠 What are Command-Line Arguments?

The primary purpose of String[] args is to allow a Java program to receive external data or configuration parameters at runtime without requiring user input during program execution or hardcoding values within the program's source code. This makes programs more flexible and adaptable to different scenarios or inputs.

➤ Real Example:

public class Hello {

public static void main(String[] args) {

for(int i = 0; i < args.length; i++) {

System.out.println("Argument " + i + ": " + args[i]);

}

}

}

➤ Run from CMD:

java Hello Apple Banana Mango

➤ Output:

Argument 0: Apple

Argument 1: Banana

Argument 2: Mango

✅ Use cases:

Passing file names

Program configuration

Automating test inputs

**📌 Full Breakdown Summary:**

Keyword Meaning Mandatory for JVM? Notes

public Accessible from anywhere ✅ Yes Allows JVM to access it

static No object required to call ✅ Yes JVM calls main directly

void No return value ✅ Yes JVM doesn’t expect a return

main Entry point method name ✅ Yes Must be spelled exactly

String[] argsCommand-line inputs ✅ Yes Even if unused

**🚨 Final Note:**

You can technically run Java programs without a main() method in some special cases, like:

Static initializer blocks (runs during class loading)

JUnit/TestNG (testing frameworks)

Servlets (web apps)

Applets (legacy)

…but for any standard console application, main(String[] args) is absolutely essential

**System.out.println();**

System is a built-in Java class that contains useful members, such as out, which is short for "output". The println() method, short for "print line", is used to print a value to the screen (or a file).

**The Print() Method**

There is also a print() method, which is similar to println().

The only difference is that it does not insert a new line at the end of the output:

**Java Comments**

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Single-line Comments

Single-line comments start with two forward slashes (//).

Java Multi-line Comments

Multi-line comments start with /\* and ends with \*/.

Any text between /\* and \*/ will be ignored by Java.