# Java Basics & Architecture

**Why Should Someone Learn Java?**

Many beginners ask one simple question:

**Why should I learn Java when there are so many languages?**

Java is popular not because it is trendy, but because it is **reliable and practical**.

Java is used in:

- Banking applications
- Enterprise software
- Backend systems
- Large and long-running applications

Once Java fundamentals are clear:

- Understanding programming becomes easier
- Learning other languages becomes faster

That is why Java is still widely used today.

---

**Why Java is Needed**

Computers are powerful, but they cannot think on their own.

They only understand **instructions**.

Examples of instructions:

- Add two numbers
- Show a message on screen
- Store user details
- Run an application

To give instructions to a computer, we use **programming languages**.

Java is one such programming language that helps us communicate with the computer in a simple and safe way.

---

**What is Programming?**

Programming means **telling the computer what to do and how to do it**.

A computer:

- Does not understand human language

- Does not make decisions on its own

It follows instructions exactly as written.

Programming allows us to:

- Solve problems
- Automate tasks
- Build applications

These instructions are written using a programming language like Java.

---

**What is Java?**

Java is a **programming language**.

Using Java, we can:

- Write instructions for the computer
- Control how data is stored
- Decide how programs behave

Java was designed to be:

- Easy to understand
- Secure
- Reliable

Java programs are used to build:

- Backend applications
- Business software
- Large-scale systems

---

**How Java is Different from Other Languages**

Some programming languages work **directly with the operating system**.

Java works differently.

Java introduces an extra layer between:

- Your program
- The operating system

Because of this extra layer:

- Java programs are safer
- Errors are easier to manage
- The same program can run on different systems

This design makes Java suitable for large and critical applications.

---

**Why Java is Platform Independent**

Java follows one important idea:

**Write Once, Run Anywhere**

This means:

- Write Java code once
- Run the same code on Windows, Linux, or Mac

You do not need to rewrite the program for different systems.

This is one of the biggest advantages of Java.

---

**How Java Code Runs (Simple Flow)**

When you write Java code, it is saved in a file with .java extension.

Example: Basics.java

This code is **not directly understood** by the computer.

So Java uses a **compiler**.

The compiler converts: .java file → .class file

The .class file contains **Bytecode**.

---

**What is Bytecode?**

Bytecode is:

- Not machine-specific
- Same for all operating systems

It is an intermediate form of code.

Because Bytecode is the same everywhere, Java programs can run on any system.

---

**JVM (Java Virtual Machine)**

JVM stands for **Java Virtual Machine**.

Each operating system has its own JVM.

The JVM:

- Reads Bytecode
- Converts it into machine instructions
- Executes the program

Java programs **always run inside the JVM**.

---

**JRE (Java Runtime Environment)**

JRE is required to **run Java programs**.

It contains:

- JVM
- Required Java libraries

If you only want to run Java applications, JRE is enough.

---

**JDK (Java Development Kit)**

JDK is required to **write and run Java programs**.

It contains:

- JRE
- Java compiler
- Development tools

Developers install JDK on their system.

---

**Simple Java Program**

```
public class Basics {
   public static void main(String[] args) {
      System.out.println("Java is Platform Independent!");
   }
}
```

**Understanding the code:**

- public → accessible from anywhere
- class → keyword used to create a class
- Basics → class name
- main → program starts execution from here
- System.out.println → prints output on screen

---

**Execution Flow**

- Program starts from main()
- JVM executes the code

- Output is displayed
- Program ends

---

**Remember This**

Java does **not** run directly on the operating system.
Java **always runs through the JVM**.

Once this idea is clear, Java architecture becomes very easy to understand.