JAVA BASICS

Java Tutorial

Last Updated : 26 Dec 2025

Our Core Java programming tutorial is designed for students and working professionals. Java is an object-oriented, class-based, concurrent, secured and general-purpose computer programming language. It is a widely used robust technology.

What is Java?

Java is a programming language and a platform. Java is a high-level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now a subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

Platform: Any hardware or software environment in which a program runs is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

Java Example

Let's have a quick look at the Java programming example. A detailed description of the Hello World! example is available on the next page.

Hey, Compile and Run Java!

```
public class Main{
   public static void main(String args[]){
    System.out.println("Hello, World!");
   }
}
```

Getting Started

Before diving into coding, we will need to set up your development environment. Java development typically requires the Java Development Kit (JDK), which includes the Java compiler and other essential tools. You can download the JDK from the official Oracle website and follow the installation instructions for your operating system.

Once we have the JDK installed, you can use a text editor or an Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans to write and run your Java code. IDEs provide features such as code completion, debugging, and project management, making them invaluable tools for developers.

Application

According to Sun Microsystems, 3 billion devices run Java. There are various devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as Acrobat Reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, tpointtech.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

Types of Java Applications

There are the following 4-types of applications that can be created using Java programming:

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone applications are Media players, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

An application that is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

Java Platforms / Editions

There are four platforms or editions of Java:

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

It is a micro platform that is dedicated to mobile applications.

4) JavaFX

It is used to develop rich Internet applications. It uses a lightweight user interface API.

Prerequisite

To learn Java, you must have a basic knowledge of C/C++ programming language.

Audience

Our Java programming tutorial is designed to help beginners and professionals.

Problem

We assure you that you will not find any problems in this Java tutorial. However, if there is any mistake, please post the problem in the contact form.

What does *concurrent* actually mean?

Concurrency =
➡️ *Multiple tasks are in progress at the same time*
➡️ They may run **simultaneously** (on multiple CPU cores) **or**
➡️ They may take **turns very fast** (on a single core)

So concurrency is about **managing many tasks at once**, not necessarily running them at the exact same instant.

What does it mean in Java?

When we say **Java is concurrent**, we mean:

1️⃣ Java supports multithreading

Java allows you to create multiple **threads** inside a single program.

Example:

- One thread handles user input
- Another thread processes data
- Another thread writes to a file

All within **one Java application**.

```
class MyTask extends Thread {
    public void run() {
        System.out.println("Task running in parallel");
    }
}

public class Main {
    public static void main(String[] args) {
        MyTask t1 = new MyTask();
        MyTask t2 = new MyTask();
        t1.start();
        t2.start();
    }
}
```

2️⃣ Java provides concurrency APIs

Java gives powerful tools to **manage threads safely**:

- `Thread`
- `Runnable`
- `ExecutorService`
- `Callable & Future`
- `ForkJoinPool`
- `CompletableFuture`
- `synchronized`

- `Locks`
- `Atomic` classes

These help avoid problems like:

- Race conditions
- Deadlocks
- Inconsistent data

3️⃣ Java can use multiple CPU cores

On modern systems, Java can run threads **truly in parallel** using multiple cores.

Example:

- Web server handling 1000 requests
- Each request handled by a separate thread

Concurrency vs Parallelism (important interview point)

Concept	Meaning
Concurrency	Many tasks in progress (structure & design)
Parallelism	Many tasks executing at the exact same time

➡️ **Java is concurrent by design**
➡️ **Java can be parallel if hardware allows**

Real-world analogy 🛣️

- **Concurrency**:
  A single chef cooking multiple dishes by switching between them.

- **Parallelism**:
  Multiple chefs cooking different dishes at the same time.

Java supports **both**.

How to say this in an interview (perfect answer)

> “Java is called a concurrent language because it provides built-in support for multithreading and concurrency utilities that allow multiple tasks to execute simultaneously or be managed efficiently within a single application.”

Why is Java called a *robust* language?

1️⃣ Strong Exception Handling

Java forces you to **handle errors properly** using exceptions.

Example:

```
try {
    int x = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Error handled safely");
}
```

➡️ Program doesn’t crash unexpectedly.

2️⃣ Automatic Memory Management (Garbage Collection)

- Java **does not allow direct memory access**
- Garbage Collector automatically removes unused objects

➡️ Prevents:

- Memory leaks
- Dangling pointers
- Invalid memory access

3️⃣ Strong Type Checking

Java checks:

- Variable types at **compile time**
- Type safety at **runtime**

Example:

```
int a = "hello"; // compile-time error
```

➡️ Errors are caught early.

4️⃣ No Pointer Arithmetic

Unlike C/C++:

- Java **does not support pointer manipulation**

➡️ Avoids:

- Memory corruption
- Security vulnerabilities
- System crashes

5️⃣ Bytecode Verification

Before execution:

- Java bytecode is **verified by JVM**

➡️ Ensures:

- No illegal instructions
- No stack overflow abuse
- No unauthorized memory access

6️⃣ Runtime Checks

Java performs runtime checks like:

- Array index bounds
- Null pointer checks

```
int[] arr = new int[3];
arr[5] = 10; // ArrayIndexOutOfBoundsException
```

➡️ Error detected safely, not silent corruption.

One-line interview definition ✅

> **Java is robust because it provides strong memory management, exception handling, type checking, and runtime safety mechanisms that make applications reliable and less prone to failure.**

What does *general-purpose* actually mean?

➡️ The language is not limited to a single domain
➡️ It can solve a wide variety of problems
➡️ It is flexible and reusable across industries

Why is Java called a general-purpose language?

Java can be used to build:

- **🌐 Web applications (Spring, JSP, Servlets)**
- **🖥️ Desktop applications (JavaFX, Swing)**
- **📱 Mobile apps (Android)**
- **⚙️ Enterprise systems (banking, ERP)**
- **☁️ Cloud & microservices**
- **🧠 AI / ML tools**
- **🎮 Games**
- **🔌 Embedded & IoT systems**

➡️ One language, many uses.

General-purpose vs Special-purpose language

General-Purpose	Special-Purpose
Used for many tasks	Used for one specific task
Flexible	Limited
Example: Java, C, Python	SQL, HTML, MATLAB

Example:

- **Java → build backend, desktop apps, Android**
- **SQL → only database querying**
- **HTML → only webpage structure**

Simple real-world analogy 🛠️

- **General-purpose language → Swiss Army knife**
- **Special-purpose language → One specific tool (like a screwdriver)**

One-line interview answer ✅

A general-purpose language is a programming language designed to solve a wide range of problems and develop different types of applications across multiple domains.

10-second interview version 🎯

“Java is a general-purpose language because it can be used for web, desktop, mobile, enterprise, and cloud applications, rather than being limited to a single domain.”

Why is Java called a *secure* language?

1️⃣ JVM Security (Sandbox Model)

Java programs run inside the Java Virtual Machine (JVM), not directly on the OS.

➡️ JVM acts like a security guard

- **Restricts file access**
- **Restricts network access**
- **Prevents illegal system operations**

2️⃣ No Direct Memory Access

- **Java does not allow pointers**
- **No pointer arithmetic**

➡️ Prevents:

- **Memory corruption**
- **Buffer overflow attacks**

3️⃣ Bytecode Verification

Before execution, JVM verifies the bytecode to ensure:

- **No illegal memory access**
- **No stack underflow/overflow**
- **No forged bytecode**

➡️ Prevents execution of malicious code.

4️⃣ ClassLoader Security

Java uses ClassLoaders to load classes securely.

➡️ Ensures:

- **Trusted classes are loaded first**
- **Untrusted code cannot override core Java classes**

Example:

```
java.lang.String // cannot be replaced by user code
```

5️⃣ Built-in Security APIs

Java provides strong security libraries:

- **Encryption & decryption**
- **Secure hashing**
- **Digital signatures**
- **Authentication**

Packages:

```
java.security
javax.crypto
```

6️⃣ Access Control (Encapsulation)

Java uses:

- `private`
- `protected`
- `public`

➡️ **Prevents unauthorized access to data.**

```
private int balance; // cannot be accessed directly
```

7️⃣ Automatic Memory Management

Garbage Collection prevents:

- **Memory leaks**
- **Use-after-free attacks**

Secure vs Robust (important difference)

Secure	Robust
Protects from attacks	**Handles errors safely**
Prevents unauthorized access	**Prevents crashes**
Focus on security threats	**Focus on reliability**

One-line interview definition ✅

Java is secure because it runs programs inside the JVM sandbox, prevents direct memory access, verifies bytecode, enforces access control, and provides strong security APIs.