

Java Fundamentals

1. App Development with Java

- Java is widely used for building various applications, particularly mobile (Android), desktop, and backend applications.
- Below are the essentials:

I. Mobile Development (Android)

- Android Studio is the official IDE for Android development.
- Java plays a pivotal role in Android apps, where you define activities, services, and UI components.
- Android apps follow the ****Model-View-Controller (MVC)** architecture, dividing the code into:

Model: Data management, typically handled using SQLite or Room Database.

View: XML files to define the user interface.

Controller: Java classes (`Activity` and `Fragment`) managing the app's logic.

Backend and Web Development with Java :

- Java is also used for building backend APIs using frameworks like:
 - Spring Boot: Simplifies backend development with embedded servers and easy setup.
 - Java EE (Enterprise Edition): For scalable enterprise applications.
 - Servlets and JSP are traditional Java technologies for web applications.

2. Full-Stack Development in Java

- Full-stack development involves both front-end and back-end technologies. With Java, here's how you can become a full-stack developer:

Backend (Java) :

- Spring Boot Framework for REST APIs.

Database Integration: Connect with MySQL, PostgreSQL, or MongoDB using Java Database Connectivity (JDBC) or Spring Data JPA.

Authentication: Use JWT (JSON Web Tokens) or OAuth for secure API access.

Frontend (Complementary Technologies)

- HTML, CSS, and JavaScript for the UI part.
- Thymeleaf or JSP for rendering dynamic content.
- Alternatively, React or Angular can serve as frontend frameworks that consume backend APIs developed in Java.

Tools and Build Management

- Maven or Gradle for managing dependencies and builds.
- Git and GitHub for version control and collaboration.

Deployment and DevOps

- Deploy Java web apps on Tomcat or use Cloud Platforms like AWS, Google Cloud, or Azure.
- CI/CD pipelines using Jenkins or GitHub Actions streamline the deployment process.

What is a Java Developer?

A Java developer is a software engineer specializing in designing, developing, and maintaining applications using Java, one of the most popular programming languages. Java developers can work across multiple domains, such as web, mobile, backend, enterprise, and cloud-based applications.

Roles and Responsibilities of a Java Developer:

1. Developing Applications:

- Build mobile apps (Android), web applications, or backend systems.

2. Writing Code:

- Use Java frameworks like Spring Boot or Hibernate to create scalable software.

3. Database Management:

- Integrate apps with relational databases (MySQL, PostgreSQL) or NoSQL (MongoDB).

4. Testing and Debugging:

- Write unit tests using tools like JUnit and identify application bugs.

5. Deploying Applications:

- Deploy apps on servers or cloud platforms such as AWS or Google Cloud.

6. Collaboration:

- Work in teams using GitHub for version control and follow Agile practices.

Type of Code

- If a set of instructions is in human-readable format, then it is called high-level code.
- If the set of instructions is in binary then it is called low-level code.
- If a set of instructions is partially human readable and partially machine readable then it is called as Intermediate code.

Keywords

- Keywords are basically a collection of predefined words where every keyword is unique(different) from another.
- Without keywords we cannot write any Java program.
- Keywords will help us in building a Java program and also help us provide the program structure.
- Java supports single-purpose keywords as well as multiple-purpose keywords.
- All the keywords in Java have to be in lowercase.

Syntax format Structure of Java Program

```
class ClassName  
{  
    // class members  
}
```

- Class is Keyword

Class Members

1. State/Properties

2. Behaviour/Members
3. Constructors
4. Blocks
5. Main-method

JDK

- JDK Stands for Java Development Kit.
- JDK is an Application.
- JDK internally contains development tools[Small Applications].
- Development tools will help us develop Java applications.
- This development tool contains javac.
- Javac[compiler] is an application that converts high-level code to intermediate code.
- JDK internally contains JRE.

JRE

- JRE Stands for Java Runtime Environment.
- JRE is also an Application and it contains Library.
- Java Library is a collection of JAR Files->refers to Java Archive.
- Rt[Run-time] contains all the necessary inbuilt Java programs.
- JRE provides the necessary conditions to execute any Java programs.
- JRE internally contains JVM.

JVM

- JVM Stands for Java Virtual Machine.
- JVM is a Translator or interpreter which helps us to convert intermediate code to low-level code.
- JVM contains JIT Compiler[JUST IN TIME] which converts intermediate code to low-level code.
- JVM contains JVM Memory, Heap Memory, Class Memory, Stack Memory, PC Registers, Negative Method Area, etc.

Execution

- Execution of a Java program is a two-step process
 1. Compilation
 2. Interpretation

1. Compilation

- Compilation is the process of converting high-level code to Intermediate code and these can be archived using an application called Javac compiler.
- Once we write a Java program it has to be saved in the destination folder with .java format or extension.
- This program will be checked by the compiler, if the compiler finds any syntax mistake then we get an error called a compilation error.
- If not the byte code will be generated.

2. Interpretation

- Interpretation is the process of converting intermediate code to low-level code and these can be archived using a translator or interpreter called (JVM) Java Virtual Machine.
 - Once the byte code is generated it will be verified by JVM.
 - If it finds any logical mistake then we get the run-time error, if not machine code will be generated which will be processed by the platform in order to get the required output.
-
- Compilation---> javac FileName.java
 - Interpretation--> java ClassName