Java Roadmap 🚀

---

1. Fundamentals (Beginner Level)

Setup & Basics

- Install **JDK** (Java Development Kit).
- Set up a development environment (e.g., **IntelliJ IDEA**, **Eclipse**, **Visual Studio Code**).
- Understand **Java architecture**.
- Learn basic **syntax** and **structure**.

Core Concepts

- **Variables** and **data types**.
- **Operators**.
- Control flow statements:
  - if-else
  - switch
- **Loops**:
  - for, while, do-while.
- **Arrays** and basic collections.

---

2. Object-Oriented Programming (OOP)

Core OOP Concepts

- **Classes** and **objects**.
- **Constructors**.
- Principles:
  - **Inheritance**
  - **Polymorphism**
  - **Encapsulation**
  - **Abstraction**
- **Interfaces** and **abstract classes**.

Advanced OOP

- **Method overloading** and **overriding**.
- **Composition** vs **inheritance**.
- **Access modifiers**.

---

3. Exception Handling

- **try-catch** blocks.
- **Checked** vs **unchecked exceptions**.
- Creating **custom exceptions**.
- **Exception propagation**.
- **finally** block.
- throw and throws keywords.

---

4. Java Collections Framework

Core Collections

- Lists: **ArrayList**, **LinkedList**.
- Sets: **HashSet**, **TreeSet**.
- Maps: **HashMap**, **TreeMap**.
- **Queue** and **Deque**.

Advanced Collections

- **Generics**.
- **Comparable** and **Comparator** interfaces.
- **Stream API**.
- **Lambda expressions** and **method references**.

---

5. Multithreading and Concurrency

- **Thread lifecycle**.
- Creating threads:
  - Extending Thread.
  - Implementing Runnable.
- **Thread synchronization**.
- **Executor framework**.
- **Concurrent collections**.
- **Thread pools**.
- wait, notify, notifyAll mechanisms.

---

6. File Handling and I/O

- File operations with File.
- **BufferedReader** and **BufferedWriter**.
- **FileInputStream** and **FileOutputStream**.
- **Serialization**.
- **NIO (New I/O) package**.

---

7. JDBC and Database Connectivity

- Database connection setup.
- Using **PreparedStatements**.
- **Transaction management**.
- **Connection pooling**.
- Basics of ORM (e.g., **Hibernate**).

---

8. Networking in Java

Fundamentals

- **Client/Server model**.
- **Socket programming** with Socket and ServerSocket.
- **TCP/IP** and **UDP** communication.

Advanced Networking

- Multi-threaded server design.
- Non-blocking I/O.
- Networking security.

---

9. Java Servlets

- Servlet lifecycle (init, service, destroy).
- HttpServlet and request-response handling.
- **Session management** with HttpSession and cookies.

---

10. Java Server Pages (JSP)

- **JSP directives** and **Expression Language (EL)**.
- Using **JSTL** (Standard Tag Library).

- **Model-View-Controller (MVC) pattern**.

---

11. Remote Method Invocation (RMI)

- **Object serialization** and **remote object invocation**.
- Implementing **Stub** and **Skeleton**.
- **Distributed computing basics**.

---

12. Java 8+ Features

- **Lambda expressions**.
- **Stream API**.
- **Optional class**.
- **Default methods** in interfaces.
- **Date and Time API**.

---

13. Design Patterns

Creational Patterns

- **Singleton**.
- **Factory**.
- **Builder**.
- **Prototype**.

Structural Patterns

- **Adapter**.
- **Decorator**.
- **Proxy**.

Behavioral Patterns

- **Observer**.
- **Strategy**.
- **Command**.

---

14. Advanced Java Concepts

- **Reflection**.

- **Annotations**.
- Advanced **Generics**.
- **Memory management**.
- **JVM internals**.
- **Performance optimization**.

---

15. Framework Knowledge

- Basics of the **Spring Framework**.
- **Spring Boot**.
- **Hibernate**.
- Building **Microservices with Java**.

---

16. Testing

- **JUnit**.
- **Mockito**.
- **Test-Driven Development (TDD)**.

---

17. Build Tools and DevOps

- **Maven** and **Gradle**.
- **Docker**.
- Continuous Integration (**CI**).

---

18. Dependency Injection and Important Topics

Dependency Injection (DI)

- **Definition:** A design pattern used to achieve Inversion of Control (IoC).
- **Benefits:**
  - Improves modularity and testability.
  - Reduces tight coupling between components.

Types of Dependency Injection

1. **Constructor Injection:**
   - Dependencies are provided through the class constructor.
2. **Setter Injection:**

    - Dependencies are provided through setter methods.
3. **Interface Injection:**
    - Dependencies are provided through interfaces.

DI Frameworks in Java

- **Spring Framework:**
    - Core container for IoC and DI.
- **Google Guice:**
    - Lightweight DI framework.
- **Dagger:**
    - Optimized for Android development.

---

19. Java Garbage Collection

- Understanding Memory Management in Java
    - Heap Memory Structure (Young Generation, Old Generation, Metaspace)
    - Object Lifecycle (Creation, Reachability, Finalization, Termination)
- Garbage Collection Basics
    - Mark-and-Sweep Algorithm
    - Generational Hypothesis
    - Stop-the-World Events
- Garbage Collector Types
    - Serial Collector
    - Parallel Collector
    - Concurrent Mark-Sweep (CMS) Collector
    - G1 (Garbage-First) Collector
- Garbage Collection Tuning and Optimization
    - JVM Command-Line Options (-Xms, -Xmx, -XX:+UseG1GC, etc.)
    - Monitoring Garbage Collection with JMX and JConsole
    - Profiling and Analyzing Garbage Collection Behavior
- Common Garbage Collection Issues
    - Memory Leaks
    - High CPU Utilization
    - Pauses and Latency
    - OutOfMemoryError Exceptions
- Best Practices for Efficient Garbage Collection
    - Minimizing Object Creation
    - Proper Object Lifecycle Management
    - Implementing Weak, Soft, and Phantom References
    - Using Finalization and Cleaners Judiciously
- Java 9+ Enhancements
    - Unified GC Logging
    - Ergonomic Improvements
    - Garbage-First Garbage Collector (G1) Enhancements

Important Topics in Dependency Injection

- Bean Configuration: XML vs Annotation-based.
- Autowired Annotation: Automatically injects dependencies.
- Qualifier Annotation: Resolves conflicts when multiple beans are available.
- Scope of Beans: Singleton, Prototype, Request, Session, etc.
- Bean Lifecycle: Initialization and destruction callbacks.
- Circular Dependencies: How DI frameworks handle them.

Hands-on Practice

- Create a Spring-based project using DI.
- Implement DI in an Android application.
- Use Google Guice or Dagger in small projects.

Learning Resources

Online Platforms

- **Codecademy**
- **Udemy**
- **Coursera**
- **edX**
- **PluralSight**

Books

- *Head First Java*.
- *Effective Java* by Joshua Bloch.
- *Clean Code* by Robert C. Martin.

Practice Platforms

- **LeetCode**
- **HackerRank**
- **CodeWars**
- **Project Euler**

Pro Tips

- Code **daily**.
- Build **projects**.
- Contribute to **open source**.
- Join **Java communities**.
- Stay updated with the latest **trends**.

Career Paths

- **Backend Developer**.
- **Full Stack Developer**.
- **Enterprise Application Developer**.
- **Android Developer**.
- **Cloud Solutions Architect**.
