Java Language - Comprehensive Notes

1. Introduction to Java

Java is a high-level, object-oriented programming language developed by Sun Microsystems in 1995. Designed with the philosophy "Write Once, Run Anywhere," Java code is compiled into bytecode that runs on any device equipped with the Java Virtual Machine (JVM). Java is known for its portability, robustness, and security.

2. Features of Java

- Simple and Familiar: Syntax is similar to C++, but with simpler object models and no explicit pointers.
- Platform-Independent: Bytecode runs on the JVM, making Java code portable across systems.
- Object-Oriented: Everything is treated as objects and classes.
- Robust: Java handles memory management automatically and provides strong exception handling.
- Secure: Java provides runtime security checks, bytecode verification, and a restricted access model.
- Multithreaded: Built-in support for multithreading.
- High Performance: Just-In-Time (JIT) compilers boost execution speed.

3. Java Architecture

- Java Compiler: Converts source code to bytecode.
- Java Virtual Machine (JVM): Executes the bytecode on any platform.
- Java Runtime Environment (JRE): Provides libraries, JVM, and other components to run applications.
- Java Development Kit (JDK): Includes JRE plus development tools.

4. Java Basics

- Data Types: byte, short, int, long, float, double, char, boolean.
- Variables and Operators: Java supports arithmetic, relational, logical, bitwise, and assignment operators.

- Control Statements: if, if-else, switch, while, do-while, for, break, continue.

5. Object-Oriented Programming (OOP)

- Class and Object: Class is a blueprint; an object is an instance of the class.

- Inheritance: Mechanism by which one class can acquire the properties of another.

- Polymorphism: Ability to take many forms (method overloading and overriding).

- Abstraction: Hiding implementation details; shown via abstract classes and interfaces.

- Encapsulation: Wrapping data and methods into a single unit using access modifiers.

6. Exception Handling

Java provides a robust mechanism to handle runtime errors:

- try, catch, finally blocks

- throw and throws keyword

- Custom exceptions

7. Java Collections Framework

Java provides built-in classes and interfaces for data manipulation:

- List: ArrayList, LinkedList

- Set: HashSet, TreeSet

- Map: HashMap, TreeMap

- Queue: PriorityQueue, LinkedList

8. Input and Output (I/O)

Java provides I/O through java.io and java.nio packages:

- Byte Streams and Character Streams

- File Handling using FileReader, BufferedReader, FileWriter, etc.

- Serialization and Deserialization

9. Threads and Concurrency

Java supports multithreading:

- Thread class and Runnable interface

- Synchronization, wait(), notify(), and notifyAll()

- Executor framework for advanced thread management

10. Java GUI Programming

- AWT and Swing libraries for GUI development.

- JFrame, JPanel, JButton, JLabel, etc.

- Event handling using ActionListener, MouseListener, etc.

11. Java Applets and Web Start (legacy)

Java Applets were used for embedding Java programs in web browsers. Java Web Start allowed launching full-featured applications via the web. Both are now deprecated.

12. Java Networking

Java provides support for networking through java.net package:

- Socket Programming (ServerSocket and Socket)

- URL and HttpURLConnection for web communication

13. JDBC (Java Database Connectivity)

- Connecting to databases using JDBC drivers.

- Steps include: Load driver, establish connection, create statement, execute query, close connection.

- Supports PreparedStatement and ResultSet for efficiency and security.

14. Advanced Java Features

- Generics: Enable classes and methods to operate on objects of various types.

- Lambda Expressions: Provide a concise way to express anonymous functions.

- Streams API: Facilitates functional-style operations on collections.

15. Java 8 and Beyond

Java 8 introduced many major enhancements:

- Functional interfaces and default methods in interfaces.

- java.time package for improved date and time API.
- Modules (Java 9), local-variable syntax (Java 10), switch expressions (Java 14), pattern matching, records, and sealed classes (Java 17+).

16. Java Development Tools and IDEs

- Popular IDEs: Eclipse, IntelliJ IDEA, NetBeans.
- Build tools: Maven, Gradle, Ant.
- Debuggers and profilers help in efficient development.

17. Best Practices

- Follow naming conventions.
- Use meaningful variable and method names.
- Handle exceptions properly.
- Avoid memory leaks by releasing resources.
- Write unit tests for code validation.

18. Summary

Java remains one of the most popular languages in enterprise development, mobile apps (Android), cloud computing, and more. With a rich API, strong community support, and continuous evolution, Java is a language of both tradition and innovation.