DAY 1 – Java Basics + JVM

1 Java Features

1. **What are the main features of Java?**
   - Platform Independent
   - Object-Oriented
   - Simple
   - Secure
   - Robust
   - Multithreaded
   - High Performance
   - Distributed
   - Portable
   - Dynamic

2. **Why is Java called platform independent?**
   - Java follows the principle "Write Once, Run Anywhere (WORA)".
     - Java source code (.java) is compiled into bytecode (.class)
     - Bytecode is platform independent
     - Bytecode runs on the Java Virtual Machine (JVM)
     - Every OS (Windows, Linux, macOS) has its own JVM
     - JVM converts bytecode into machine-specific code
   - Because only the JVM changes (not the program), Java is platform independent.

3. **Is Java 100% object-oriented? Why or why not?**
   - Java is not 100% object-oriented because it supports primitive data types such as int, char, float, double, boolean, etc., which are not objects.

4. **Why is Java secure?**
   - Java is secure because it uses bytecode verification, JVM security architecture, no explicit pointers, automatic memory management, and a security manager, which together prevent memory corruption and unauthorized access.

5. **Why Java is robust?**
   - Java is robust because it provides strong memory management, automatic garbage collection, compile-time and runtime error checking, and a powerful exception-handling mechanism, which makes programs reliable and stable.

6. **What is bytecode? How Bytecode Works.**

- Bytecode is the .class file generated by the Java compiler, which is executed by the Java Virtual Machine (JVM) instead of directly running on the operating system.
- Bytecode Working:-
  - Java source code is written in a .java file
  - javac compiler converts it into bytecode (.class)
  - Bytecode is stored in class files
  - JVM reads and executes bytecode on any platform

7. Why Java uses garbage collection?

- Java uses garbage collection to automatically free memory by deleting unused objects, which improves performance, prevents memory leaks, and makes programs safer and easier to manage.

8. Difference between Java and C++.

Feature	Java	C++
Platform Independence	**Platform independent (runs on JVM)**	**Platform dependent**
Execution	**Compiled to bytecode, then run on JVM**	**Compiled directly to machine code**
Object-Oriented	**Not 100% OOP**	**Not 100% OOP**
Pointers	**No explicit pointers**	**Supports pointers**
Memory Management	**Automatic (Garbage Collection)**	**Manual (new / delete)**
Multiple Inheritance	**Not supported (class level)**	**Supported**
Security	**More secure (no pointers, JVM sandbox)**	**Less secure**
Performance	**Slightly slower (JVM overhead)**	**Faster (native execution)**
Operator Overloading	**Not supported (except + for String)**	**Supported**
Preprocessor	**No preprocessor**	**Uses preprocessor (#include)**
Thread Support	**Built-in multithreading support**	**No built-in thread support**
Exception Handling	**Strong exception handling**	**Less structured**

9. Why Java does not support pointers?

- Java does not support pointers because they can directly access memory, which may cause security issues, memory corruption, and program crashes.

10. Why Java supports multithreading?

- Java supports multithreading to perform multiple operations at the same time, which increases application performance and responsiveness.
- **Java Supports Multithreading:**
  - **i. Better Performance**
    - ✓ **Multiple threads run in parallel**
    - ✓ **Faster execution of programs**
  - **ii. Efficient Use of CPU**
    - ✓ **Utilizes CPU resources effectively**
    - ✓ **Reduces idle time**
  - **iii. Improved Responsiveness**
    - ✓ **Applications don't freeze**
    - ✓ **Especially useful in GUI and web apps**
  - **iv. Resource Sharing**
    - ✓ **Threads share memory**
    - ✓ **Less memory usage compared to processes**
  - **v. Built-in Support**
    - ✓ **Java provides Thread class and Runnable interface**
    - ✓ **Easy to implement multithreading**