Java

Assignment No-23

Q.1] What are the features of Java programming languages.

→ 1] Java follows object-oriented programming principles like encapsulation, Inheritance, polymorphism and abstraction.

2] Java code is compiled into bytecode, which can run on any system with a Java Virtual Machine making it "Write Once, Run Anywhere"

3] Java has a simple syntax similar to C++ but removes complex features like pointers and operator overloading.

4] Java provides built-in security mechanisms such as bytecode verification, security Manager, and lack of explicit pointer to prevent unauthorized memory access.

5] Java supports multithreading, allowing multiple tasks to run simultaneously, improving performance in concurrent application.

Q.2] What is mean by platform independent & Architecture Neutral.

→ Platform Independent.

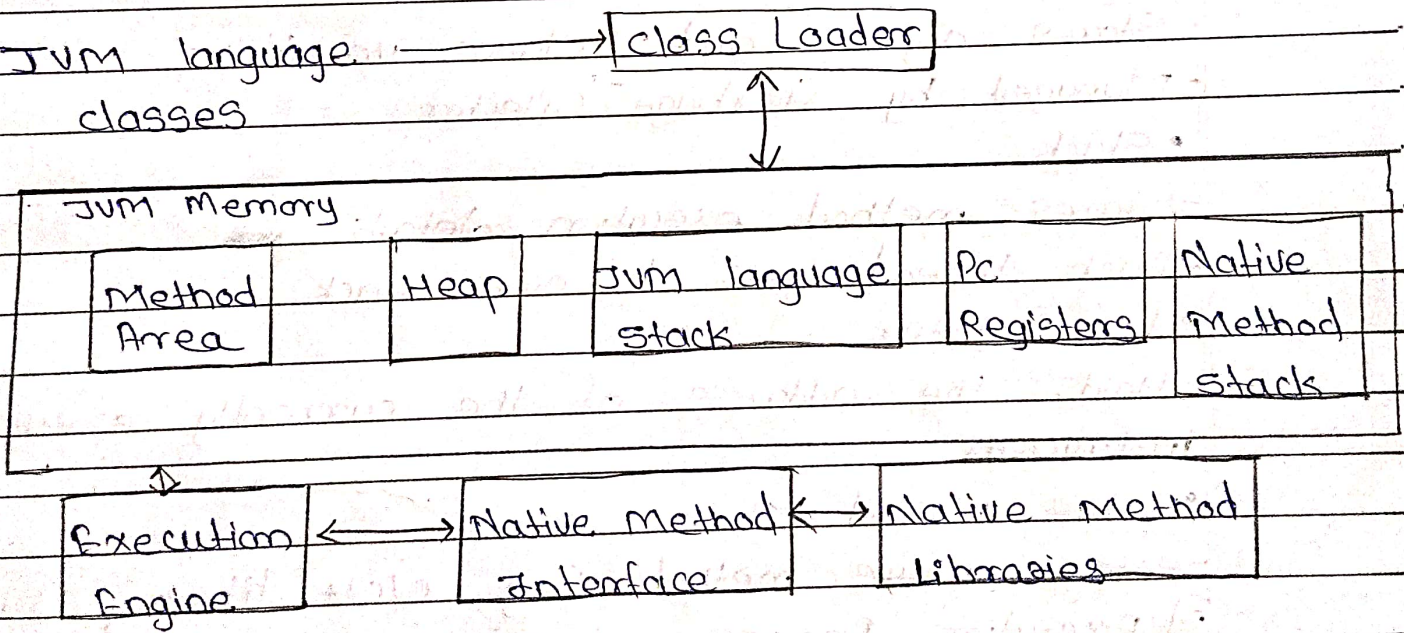
- Java is platform-independent because java program are not directly compiled into machine code specific to a single operating system.
- Instead, Java source code is compiled into an intermediate form called bytecode which can be executed on any device that has a Java Virtual Machine.
- This allows Java programs to run on different operating systems without modification.

• Architecture Neutral

- Java is architecture neutral because it does not depend on the underlying hardware.
- The JVM acts as an abstraction layer between the java program and the hardware, ensuring that the same Java bytecode can run on different hardware architecture.
- Unlike C or C++, where compiled code is specific to a particular CPU architecture, Java remains portable across different systems.

Q3] Explain JVM architecture in details.

→



The Java Virtual Machine is the engine that run Java applications by converting bytecode into machine code. It provides platform independent and manages memory, execution and security.

• Main Components of JVM architecture.

1] Class Loader Subsystem -

- loads .class file into memory.
- Responsibilities -
  • Loading - finds and loads class files.
  • Linking - Verifies and prepares classes for execu
  • Initialization - Executes static blocks and assigns values to static variables.

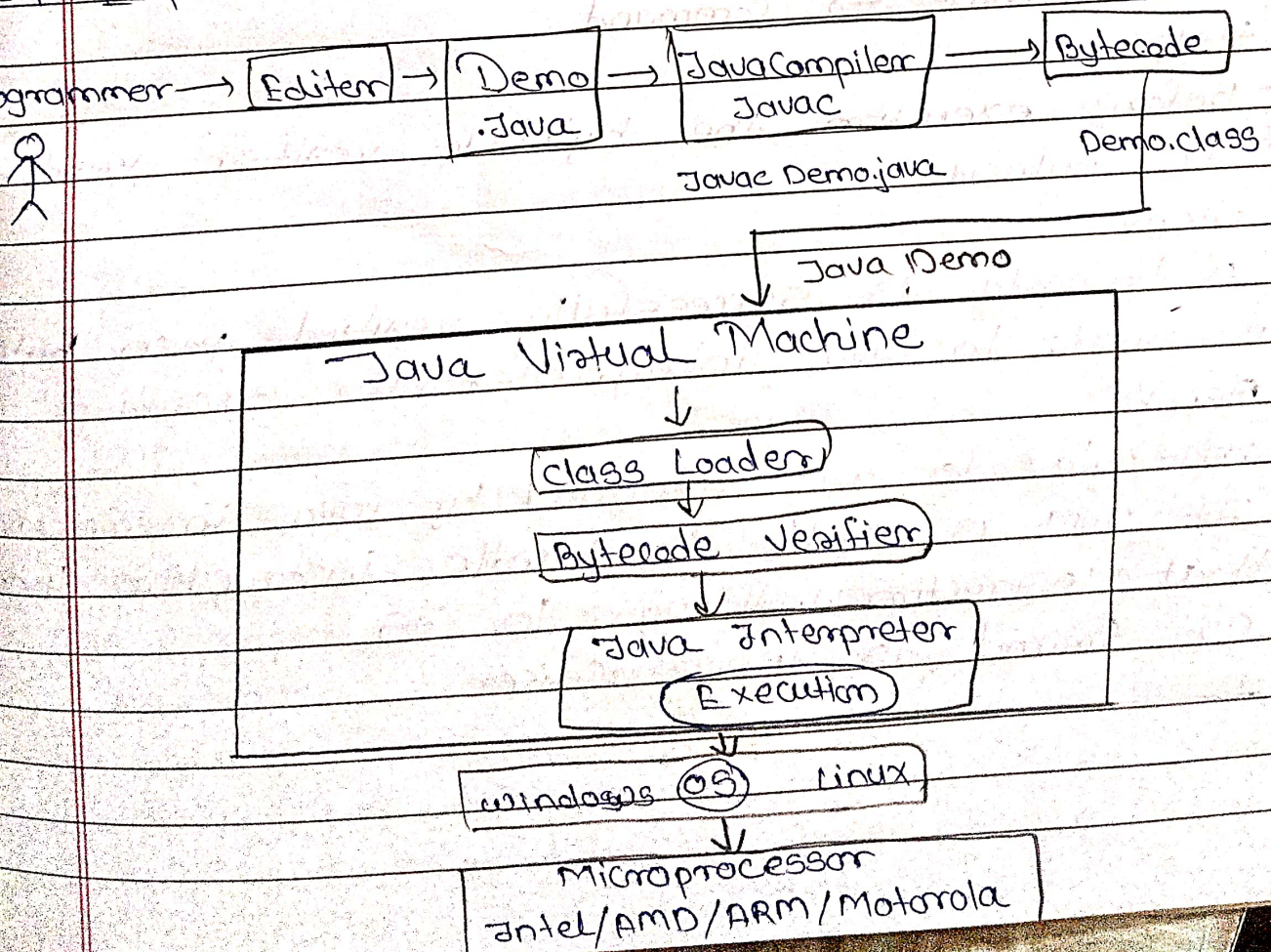
2) JVM Memory (Runtime Memory Areas)
JVM divides memory into several areas for efficient execution.
• Method Area (Class Area)
- Stores class structures like method code, Constan pool, and static variables.
- Shared among all threads.
• Heap
- Stores objects and instance variables
- Managed by Garbage Collector
• Stack
- Stores method execution details
- Each thread gets its own stack.
• Pc Register
- Holds the address of the currently executing instruction.
• Native Method Stack
- Stores native method (e.g. c/c++ library functions)

3) Execution Engine
- Interpreter - Reads and executes bytecode line by line
- Just In Time Compiler - Converts frequently used bytecode into native code for faster execution.
- Garbage Collector (GC) - Automatically removes unused objects from memory.

4] Java Native Interface (JNI)
- Allows java to call native method for system specific operations.
5] Java Native Libraries
Includes platform-specific libraries for OS interation

Java execution process
① Java code (.java) → compiled to bytecode (.class) using javac.
② JVM loads the class file using class loader.
③ Execution Engine reads the bytecode and executes it using the Interpreter or JIT Compiler.
④ Garbage Collector cleans up unused objects from memory.

Q.4 Explain Tool chain of Java.



- Step I -
Java programmer uses any IDE or editors to write the java program
After writing java code we have to save the file with the extension .java (Demo.java)
- Step II
Now we passed our java file to java compiler
Javac → Name of compiler
Javac Demo.java → Command
- Step III
After successful compilation bytecode of java program gets generated with is understandable to java machine.
The bitcode get stored inside .class file (Demo.cl
- Step IV
Now the .class file get loaded inside JVM.
Java filename ↔ Command
Help
- Before execution the bytecode verifier verified the bytecode.
- Step V
If bytecode is successfully verified then bytecode is sends to java interpreter for execution
- Step VI
Now byte code will executed by java interpreter.
If JVM is unable to handle instruction then that instruction will pass to OS
OS forward to microprocessor

Q5] What is mean by JDK, JRE, JVM?

→ • JVM (Java Virtual Machine)

JVM is runtime environment that executes Java bytecode.

- It provides platform independence by converting bytecode into machine specific instructions.
- It includes -
  - Class loader (loads .class file)
  - Executes Engine (Interpreter + JIT Compiler)
  - Garbage collector (Automatic memory management)

* It run java programs by interpreting bytecode.

2] JRE (Java Runtime Environment)

- JRE is a package that contains JVM + essential libraries required to run java applicat ion.

* Provides an environment to run Java applicati ons but cannot compile code.

3] JDK (Java Development kit)

JDK is complete Java development package that includes -

- JRE (JVM + libraries)
- Development Tools (Javac, jdb, jar, javadoc, etc)

- Needed for writing, compiling and running Java programs

* Required for developing and running Java applications.

Q6] Why java is considered as pure object oriented programming language.

→

Java follows object-oriented Programming principles making it one of the most object-oriented languages.
However, whether Java is purely oop is debated because of its support for primitive data type.

- Reason Why Java considered Object-oriented

① Encapsulation
Java bundles data and methods into objects. Access to data is controlled using access specifiers

② Inheritance
Java supports code reuse by allowing classes to inherit properties and behaviours from other classes using extends.

③ Polymorphism
Java allows method overloading and method overriding, enabling multiple behaviours under a single interface.

④ Abstraction
Java allows defining abstract classes and interfaces promoting modularity and reducing complexity.

⑤ Everything is in object
In java (except primitive types like int, char, double) everything is an object
Even primitives can be wrapped into objects using wrapper class.

⑥ No direct use of pointer
Unlike C++, Java does not allow direct access to memory using pointers, making it more object oriented and secure.

Q7] What is mean by 'write once Run Anywhere' strategy of Java.

→ WORA strategy means that Java programs can be written and compiled once and then executed on any platform without modification. This is achieved through Java bytecode and the Java Virtual Machine (JVM).

- How WORA works:

1] Write Java code
- Developers write Java programs using .java source file.

2] Compile to Bytecode
- The java compiler (javac) converts the .java file into bytecode (.class file).
- Bytecode is an intermediate representation that is not platform specific.

3] Run on Any platform with a JVM
- The JVM interprets and execute bytecode on any operating system.
- Since JVM implementations exist for different platforms, the same bytecode can run anywhere.

Q8] What is difference between compiler and Interpreter?

→

Compiler	Interpreter
- Translate the entire program at once	Translates and executes line by line
- Faster	Slower

Compiler	Interpreter
- Detects all errors at once after compilation	Stops at the first error & requires fixing before Contin
- Generates an Independent executable file	Does not generate an independ file, runs the program directly.
- C, C++	Python, Javascript

Q9] What is difference between Native language and virtual machine based language?

→

	Native language	virtual machine based
1]	Directly executed by the hardware (CPU)	Runs on virtual machine inste of directly on hardware
2]	Compiled into machine code specific to an OS/CPU.	Compiled into bytecode, which is interpreted by a VM
3]	Not portable (os & hard-ware dependent).	Highly portable (can run on any system with the VM
4]	Faster (runs directly on hardware)	Slightly slower due to the VM overhead.
5]	ex- C, C++ compiled into machine code	ex- Java, python, C#.

Q10] How java is considered as secure language?

→ Java is designed with several security features that make it safer than many other programming languages.

It prevents direct memory access, protects against vulnerabilities and provides a secure execution environment.

1] No explicit memory manipulation (No pointers)

- Unlike c/c++, Java does not use pointers, which prevents direct memory access.
- This eliminates security issues like buffer overflow and memory corruption.

2] Automatic Memory Management (Garbage Collector)

Java has an automatic Garbage collector that removes unused objects from memory.

- Prevents memory leaks and dangling pointers

3] Bytecode Verification

Java code is compiled into bytecode (.class file) which is then executed by the JVM

- Before execution, the bytecode verifier checks for: Invalide code, Malicious code, violation of access rule.
- This prevents unauthorized access and corruption.