Name: Atharva Kishor Pimple (Juhu)

PRN: 250240520014

Part-1:

1. **What is Java? Explain its significance in modern software development.**
   Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle). It is widely used for building cross-platform applications, including web, mobile, and enterprise applications. Its significance lies in its portability, security, scalability, and strong community support.

2. **List and explain the key features of Java.**
   Platform Independence – Java programs run on any OS with a Java Virtual Machine (JVM).
   Object-Oriented – Supports OOP concepts like inheritance, encapsulation, and polymorphism.
   Automatic Memory Management – Uses Garbage Collection for memory efficiency.
   Multithreading – Supports concurrent execution of multiple threads.
   Security – Has built-in security features like bytecode verification and a security manager.
   Rich API – Provides a vast set of libraries for networking, database handling, etc.

3. **What is the difference between compiled and interpreted languages? Where does Java fit in?**
   Compiled Languages (e.g., C, C++): Translated directly into machine code, leading to faster execution.
   Interpreted Languages (e.g., Python, JavaScript): Executed line by line, making them slower.
   Java: Uses a hybrid approach—code is first compiled into bytecode and then interpreted by the JVM, making it both compiled and interpreted.

4. **Explain the concept of platform independence in Java.**
   Java programs are compiled into an intermediate bytecode, which can run on any system that has a JVM, making Java platform-independent under the "Write Once, Run Anywhere" (WORA) principle.

5. **What are the various applications of Java in the real world?**
   Web Applications (Spring, Hibernate, Servlets)
   Mobile Applications (Android Development)
   Enterprise Applications (Banking and Finance Systems)
   Cloud Computing (AWS, Google Cloud)
   Big Data and Machine Learning (Hadoop, Apache Spark)
   Game Development (Minecraft)

Part-2: History of Java

1. **Who developed Java and when was it introduced?**
   Java was developed by James Gosling and his team at Sun Microsystems and was officially released in 1995.

2. **What was Java initially called? Why was its name changed?**
   Java was originally called "Oak", named after an oak tree outside James Gosling's office. The name was later changed to Java due to trademark issues.

3. **Describe the evolution of Java versions from its inception to the present.**
   Java 1.0 (1995) – Initial release
   Java 2 (1998-1999) – Introduced Swing and Collections Framework
   Java 5 (2004) – Introduced generics, enhanced for-loop
   Java 8 (2014) – Introduced Lambda Expressions, Stream API
   Java 11 (2018) – Long-term support (LTS) version
   Java 17 (2021) – Latest LTS version with performance improvements
   Java 21 (2023) – Introduced virtual threads, pattern matching

4. **What are some of the major improvements introduced in recent Java versions?**
   Java 8: Lambda expressions, Stream API, functional interfaces
   Java 11: var keyword, HTTP Client API
   Java 17: Sealed classes, foreign function API
   Java 21: Virtual threads, Pattern matching for switch.
5. **How does Java compare with other programming languages like C++ and Python in terms of evolution and usability?**
   Java vs. C++: Java is more portable, secure, and manages memory automatically, whereas C++ provides more direct memory control.
   Java vs. Python: Java is faster and more scalable, whereas Python is more concise and beginner-friendly due to its simpler syntax.

Part-3:

1. **Explain the importance of data types in Java.**
   Data types in Java define the kind of data that a variable can hold. They help in:

   Memory efficiency: Allocating only the required memory.
   Type safety: Preventing unintended operations between incompatible types.
   Performance optimization: Primitive types are more efficient than objects.
   Code readability and maintainability.

2. **Differentiate between primitive and non-primitive data types.**

Primitive Data type	Non Primitive Data type
Stored in stack memory	Stored in heap memory
Predefined in Java	User-defined
int, double, char, boolean	String, Array, Class, Interface
Directly store values	Store references to objects

3. **List and briefly describe the eight primitive data types in Java.**

   byte (1 byte) – Stores whole numbers from -128 to 127.
   short (2 bytes) – Stores whole numbers from -32,768 to 32,767.
   int (4 bytes) – Stores whole numbers from -2^31 to 2^31-1.
   long (8 bytes) – Stores whole numbers from -2^63 to 2^63-1.
   float (4 bytes) – Stores fractional numbers (single-precision floating point).
   double (8 bytes) – Stores fractional numbers (double-precision floating point).
   char (2 bytes) – Stores a single character (Unicode).
   boolean (1 bit) – Stores true or false.

4. **Provide examples of how to declare and initialize different data types.**
   byte b = 100;
   short s = 32000;
   int i = 100000;
   long l = 10000000000L;
   float f = 10.5f;
   double d = 99.99;
   char c = 'A';
   boolean isJavaFun = true;

5. **What is type casting in Java? Explain with an example**.
   Type casting is converting one data type into another. It is of two types:
   Implicit (Widening) Casting: Smaller to larger type (automatically done).
   int num = 10;
   double d = num;
   Explicit (Narrowing) Casting: Larger to smaller type).
   double pi = 3.14;
   int intPi = (int) pi;

6. **Discuss the concept of wrapper classes and their usage in Java.**
   Wrapper classes are used to convert primitive types into objects and vice versa. Java provides wrapper classes for all primitive types:
   Examples: Integer, Double, Character, Boolean, etc.

7. **What is the difference between static and dynamic typing? Where does Java stand?**
   Java is a statically typed language, meaning that variable types must be declared explicitly and are checked at compile-time.

Feature	Static Typing	Dynamic Typing
Type Checking	At compile-time	At runtime
Performance	Faster, optimized	Slower due to runtime checks
Errors	Caught at compile-time	Caught during execution
Examples	Java, C, C++	Python, JavaScript

Part-5:

1. **What is JDK? How does it differ from JRE and JVM?**
   JDK (Java Development Kit): A software development kit that includes tools needed to develop, compile, and run Java applications.
   JRE (Java Runtime Environment): Provides libraries and components to run Java programs but lacks development tools.
   JVM (Java Virtual Machine): A runtime engine that executes Java bytecode on any platform.
   Difference:
   JDK = JRE + development tools (compiler, debugger, etc.)
   JRE = JVM + libraries required to run Java applications
   JVM = Engine that interprets bytecode and runs Java programs

2. **Explain the main components of JDK.**
   Compiler (javac) – Converts Java source code into bytecode.
   Java Runtime Environment (JRE) – Allows Java programs to run.
   Java Virtual Machine (JVM) – Executes Java bytecode.
   Libraries – Predefined classes and packages.
   Debugger (jdb) – Helps debug Java applications.
   Java Archive Tool (jar) – Packages Java classes into .jar files.
   Documentation Tool (javadoc) – Generates documentation from Java comments.

3. **Describe the steps to install JDK and configure Java on your system.**
   Download JDK from Oracle or OpenJDK.
   Install JDK by running the installer and following setup instructions.
   Set Environment Variables :
   Add JDK path to PATH variable: C:\Program Files\Java\jdk\bin
   Verify Installation by running:
   java –version
   javac –version

4. **Write a simple Java program to print "Hello, World!" and explain its structure.**

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World!");
   }
}
```

public class HelloWorld – Defines a class named HelloWorld.
public static void main(String[] args) – The main method, the entry point of execution.
main- is a method name.
void- is a return type.
Static- for jvm to access without object creation.
String args[]- for command line args.
System.out.println("Hello, World!"); – Prints "Hello, World!" to the console.

5. **What is the significance of the PATH and CLASSPATH environment variables in Java?**
PATH: Specifies where the system should look for Java executables (java, javac).
CLASSPATH: Specifies where Java should look for user-defined classes and JAR files.

6. **What are the differences between OpenJDK and Oracle JDK?**

Feature	OpenJDK	Oracle JDK
License	Open-source (GPL)	Commercial with free LTS versions
Performance	Similar to Oracle JDK	Optimized for enterprise use
Updates	Community-driven	Oracle provides updates & support
Usage	Free for all users	Requires a subscription for commercial use

7. **Explain how Java programs are compiled and executed.**
Write Code: Create a .java file.
Compile: Convert Java code to bytecode using javac:
javac File.java

Execute: Run the bytecode using java:

java File

8. **What is Just-In-Time (JIT) compilation, and how does it improve Java performance?**
   Just-In-Time (JIT) compilation is a technique used by the Java Virtual Machine (JVM) to improve the performance of Java applications. Instead of interpreting bytecode line by line, the JIT compiler translates frequently used bytecode sequences into native machine code at runtime. This reduces the overhead of repeated interpretation and allows Java programs to run faster. JIT compilation enhances performance by optimizing code execution, eliminating redundant computations, and leveraging processor-specific instructions.

9. **Discuss the role of the Java Virtual Machine (JVM) in program execution.**
   The Java Virtual Machine (JVM) is a crucial component of the Java runtime environment. It provides a platform-independent execution environment for Java programs by converting Java bytecode into machine-specific instructions. The JVM handles memory management, garbage collection, security enforcement, and exception handling. It enables features like platform independence through the "Write Once, Run Anywhere" principle and optimizes execution using techniques like JIT compilation. Overall, the JVM plays a vital role in ensuring efficient, secure, and portable execution of Java applications.