// Introduction to Java

// 1.

Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle) in the mid-1990s. It is known for its portability, security, and robustness, making it one of the most popular programming languages in the world. Java's significance in modern software development is immense, as it is widely used for building web applications, mobile applications (especially Android apps), enterprise software, scientific applications, and more. Its "write once, run anywhere" capability allows developers to create applications that can run on various devices and platforms without modification.

// 2.

Object-Oriented: Java is based on the principles of object-oriented programming (OOP), which allows for modular and reusable code.

Platform-Independent: Java code is compiled into bytecode, which can run on any device with a Java Virtual Machine (JVM), making it platform-independent.

Robust: Java has strong memory management and exception handling features, which help in creating reliable and error-free applications.

Secure: Java provides a secure environment through its runtime checks, memory management, and built-in security features, making it suitable for internet-based applications.

Multithreaded: Java supports multithreading, enabling the development of applications that can perform multiple tasks simultaneously.

High Performance: Java's Just-In-Time (JIT) compiler improves performance by compiling bytecode into native machine code at runtime.

Distributed: Java has built-in support for distributed computing, enabling the development of applications that can run on multiple machines.

// 3.

Compiled Languages: In compiled languages (e.g., C, C++), the source code is translated into machine code by a compiler before execution. The machine code is specific to the target platform and can be executed directly by the computer's hardware.

Interpreted Languages: In interpreted languages (e.g., Python, JavaScript), the source code is executed line-by-line by an interpreter at runtime. This makes them slower than compiled languages but allows for greater flexibility and easier debugging.

Java: Java is unique in that it combines both compilation and interpretation. Java source code is first compiled into bytecode by the Java compiler. This bytecode is platform-independent and can be executed on any device with a JVM. The JVM then interprets and/or compiles the bytecode into native machine code at runtime using the Just-In-Time (JIT) compiler.

// 4.

Platform independence refers to the ability of Java programs to run on any device or operating system that has a JVM, without requiring any modifications. This is achieved through the compilation of Java source code into bytecode, which is an intermediate, platform-independent representation of the code. When a Java program is executed, the JVM on the target platform interprets and/or compiles the bytecode into native machine code. This allows Java applications to be "write once, run anywhere" (WORA), making them highly portable and versatile.

// 5.

Web Applications: Java is widely used for developing dynamic web applications, using frameworks like Spring, Hibernate, and Struts.

Mobile Applications: Java is the primary language for Android app development, using the Android SDK.

Enterprise Applications: Java is used for building large-scale, robust, and scalable enterprise applications, often using Java EE (Enterprise Edition).

Scientific Applications: Java is used in scientific research and development for its robustness, security, and ability to handle large amounts of data.

Financial Services: Java is used in the finance industry for creating trading platforms, risk management systems, and other financial applications.

Embedded Systems: Java is used in embedded systems and Internet of Things (IoT) devices, as it can run on small and resource-constrained devices.

Gaming: Java is used to develop games and gaming engines, leveraging its multithreading capabilities and robustness.

// History Of Java

// 1.
Java was developed by James Gosling and his team at Sun Microsystems. It was officially introduced in 1995.

// 2.
Java was initially called "Oak," named after an oak tree that stood outside James Gosling's office. The name was changed to "Java" due to trademark issues and was inspired by Java coffee.

// 3.
Java 1.0 (1996)

Java 1.1 (1997)

Java 2 (1998)

Java 5.0 (2004)

Java 6 (2006)

Java 7 (2011)

Java 8 (2014)

Java 9 (2017)

Java 10 (2018)

Java 11 (2018)

Java 12 (2019)

Java 13 (2019)

Java 14 (2020)

Java 15 (2020)

Java 16 (2021)

Java 17 (2021)

// 4.
Pattern Matching

Records

Sealed Classes

Text Blocks

New Garbage Collectors

Foreign Function & Memory API (Preview)

Vector API (Incubator)

// 5.
C++:

Evolution: C++ has evolved with features like STL, smart pointers, and lambda expressions. However, it has a steeper learning curve due to its complexity and manual memory management.

Usability: Java is easier to learn and use, with automatic memory management (garbage collection) and a rich standard library. C++ offers more control over system resources, making it suitable for system programming and performance-critical applications.

Python:

Evolution: Python has seen rapid adoption and evolution, particularly in data science, machine learning, and web development. It emphasizes code readability and simplicity, with a large ecosystem of libraries and frameworks.

Usability: Python is known for its ease of use and readability, making it an excellent choice for beginners and rapid development. Java is more verbose but offers better performance, strong typing, and a more structured approach to object-oriented programming.

// Data Types in Java-

// 1.
Data types define the operations possible on data, ensure efficient memory management, and guarantee data integrity.

// 2.
Primitive Data Types: Predefined, stored directly in memory.

Non-primitive Data Types: User-defined, reference objects, stored in heap memory.

// 3.
byte: 8-bit integer.

short: 16-bit integer.

int: 32-bit integer.

long: 64-bit integer.

float: 32-bit floating-point.

double: 64-bit floating-point.

boolean: true or false.

char: 16-bit Unicode character.

```
// 4.
// Primitive data types
int number = 10;
char letter = 'A';
boolean isJavaFun = true;
double decimalNumber = 9.99;

// Non-primitive data types
String text = "Hello, Java!";
int[] array = {1, 2, 3, 4};
```

```
// 5.
```
Implicit-

```
int a = 5;
double b = a;
```

Explicit-

```
double x = 5.5;
int y = (int) x;
```

// 6.
Wrapper classes provide a way to use primitive data types as objects. Examples include Integer, Double, Boolean. They are useful for collections, performing operations on primitives requiring objects, and handling null values.

// 7.
Static Typing: Type checking at compile-time. Safer, optimized code.

Dynamic Typing: Type checking at runtime. More flexibility, potential runtime errors.

Java: Statically typed.

// Java Development Kit (JDK)-

// 1.
The Java Development Kit (JDK) is a software development kit used to develop Java applications. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java Virtual Machine, or JVM), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed for Java development.

JRE: The Java Runtime Environment provides the libraries, Java Virtual Machine (JVM), and other components to run applications written in Java.

JVM: The Java Virtual Machine is the engine that runs Java applications, converting Java bytecode into machine code that can be executed by the computer's CPU.

// 2.
The main components of the JDK include:

Java Compiler (javac): Converts Java source code into bytecode.

Java Virtual Machine (JVM): Executes Java bytecode.

Java Runtime Environment (JRE): Includes JVM and core libraries.

Java Archiver (jar): Packages related class files into a single archive file.

Java Debugger (jdb): Helps in debugging Java programs.

JavaDoc: Generates documentation from Java source code.

// 3.
Steps to install JDK and configure Java:

Download the JDK from the official Oracle website or OpenJDK.

Run the installer and follow the instructions.

Set the JAVA_HOME environment variable to the JDK installation path.

Add the JDK bin directory to the system PATH.

Verify the installation by running java -version and javac -version in the command prompt or terminal.

// 4.
```java
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, entry point of the program.

System.out.println("Hello, World!"): Prints the string "Hello, World!" to the console.

5.
PATH: An environment variable that tells the operating system where to find executable files. Including the JDK bin directory in the PATH allows you to run Java commands from any directory.

CLASSPATH: Specifies the location of user-defined classes and packages. It's essential for Java to find the necessary files to compile and run applications.

// 6.
OpenJDK: An open-source implementation of the Java Platform, Standard Edition.

Oracle JDK: The official JDK provided by Oracle, with additional commercial features and support.

// 7.
Java programs are compiled and executed in two steps:

Compilation: The javac compiler converts Java source code into bytecode, stored in .class files.

Execution: The JVM loads the bytecode and translates it into machine code, executing the program on the host machine.

// 8.
Just-In-Time (JIT) compilation is a part of the JVM that improves performance by compiling bytecode into native machine code at runtime. This allows frequently executed code to run faster, as it avoids the overhead of interpreting bytecode each time it's executed.

// 9.
The JVM plays a crucial role in executing Java programs:

It loads the compiled Java bytecode.

It verifies and interprets the bytecode.

It manages memory and resources.

It provides a platform-independent execution environment, allowing Java programs to run on any device with a compatible JVM.