# Java Programming Basic.

## Unit 1 - Introduction to Java.

1. What is Java? Explain its significance in modern software development.

→ Java is a high level objected-oriented & indeed platform independent programming language.
It has very much significance as it is most commonly used programming language with latest LTS versions it increases it's significance as it follows the principle of 'WORA' architecture i.e. write once Run anywhere.

2. List and explain key features of Java.

→ 1. follows WORA principle.
- this means that the wora principle is write once Run anywhere. It just needed to be written ~~anyt~~ one time and can run everywhere.

2. High-level programming language.
3. Platform independant → can run on any platform like mac os, Linux, & windows.
4. keeps ~~geett~~ getting update of latest versions.

3. What is the difference betw^n compiled & interpreted language? where does Java fit in?

→ compiled languages translate the entire source code into machine code before execution. (for ex. C, C++) which makes them faster but requires a compilation step.

Interpreted language on other hand translate and execute the code line by line for ex. Java making them more flexible but a bit slower.

with respect to java it is both it first compiled into bytecode (by the compiler) & then interpreted/executed by JVM (Java virtual machine). which makes it platform-independant.

4. Explain the concept of platform independance in java.
→ Platform independance in java means that the java programs can run on any operating system without any modification. which is achieved through (JVM) which executes the bytecode which can run on any operating system.
As long as a system has a proper JVM.

5. What are various applications of java in the real world?
→ The various applications of java are as follows-
1) web development
2) mobile application.
3) Game development
4) cloud computing.
5) cybersecurity.
6) IOT
7) scientific computing.

## [Part 2: History of Java]

1. Who developed java and when was it introduced?
→ Java was developed by James Arthur Gosling at sun-microsystem and was introduced in the year 1995.

2. What was java initially called? Why was its name changed?

→ Java was initially called 'oak' as there was an oak tree outside James Gosling office.
The name was changed to Java because oak was already trademarked of another company.

3. Describe the evolution of Java versions from its inception to the present?

→
- 1996 → first Java development kit (JDK 1.0) launched.
- 1997 → Java became the offical language for web development.
- 1999 → Java2 (J2SE, J2EE) introduced, bringing significant improve.
- 2006 → Sun microsystems made Java-open source & GPL.
- 2010 → oracle acquired sun-microsystem taking over Java development.
- 2014 → Java 8 released, introducing lambda expression & stream API.
- 2017 → oracle switched to a faster Java release cycle (every 6 month)
- 2018 → Java 11 became a long-term support (LTS) version.
- 2021 → Java 17 released as the next LTS version with modern features
- 2024 → Java 21 (latest LTS version) released, bringing virtual threads & pattern matching.

4. What are some of the major improvements introduced in recent Java versions?

→ There are some significant improvements.
- Java 21 → virtual threads for better concurrency, pattern matching, Record Patterns.
- Java 22 → Enhanced foreign function & memory API, improved vector API for performance.
- Java 23 → string templates for better string handling, variables for cleaner code.

5. How does Java compare with other programming language like c++ and python in terms of evolution & usability?

→ Java simplifies c++ by removing complexities like pointer and manual memory management while having platform independance via the JVM.

Compared to Python, Java is more structured and optimized for enterprise applications, while Python excels in simplicity, scripting and AI.

Java balances performance, portability and usability making it ideal for large-scale systems.

## Part 3: Data Types in Java

1. Explain the importance of data types in Java.

→ Data types in Java define the type of data a variable can hold, ensuring memory efficiency, type safety & optimized performance.

They help prevent errors, enabling efficient operations & improve code readability.

Java has primitive (eg. int, double, char) and reference (objects, arrays) data types ensuring structured & reliable programming.

2. Differentiate between primitive & non-primitive data types.

| → Primitive | Non-Primitive. |
|---|---|
| 1) Basic data types that stores values directly | Refrence types that store memory addresses of objects |
| 2) Ex. int, double, char, float | -string, array, class |
| 3) cannot be null | - can be null |
| 4) fixed size, faster performance | - Dynamic size more flexible. |
| 5) Requires less memory, stores actual value. | - Requires more memory, stores refrence. |

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

→ 1 byte = 8 bit

1. int data types. → 32 bytes.
   Range → $-2^{31}$ to $2^{31}-1$

   → 4 bytes.

2. Short → 16 bits.
   - Range → -32768 to 32767.

3. long → 64 bits.
   Range → $\pm 9.2E^{18}$

4. float → $\pm 34^{E38}$
   ↳ used where regular calculation & small calculations.

5. double → $\pm 1.7E^{308}$
   ↳ used where precision is important.

6. char → 0 to 65,535.

7. boolean → true/false.
   ↳ default value is considered false.

8. byte → stores 8-bit signed integers.
   Range → -128 to 127.

4. Provide examples of how to declare & initialize different data types.

→ . How to declare.

```
// primitive data types.
byte b = 100;
short s = 2000;
```

```
int i = 100000;
long l = 10000000L;
float f = 10.5f;
double d = 99.99;
char c = 'A';
boolean = true;

// Non-Primitive data types.
String str = "Hello, Java!";
int[] arr = {1,2,3,4,5}; // Array.
Integer objInt = 500; // wrapper class.
```

5. What is type casting in Java? Explain with example.

→ Type casting is converting one data type into another.

1. Implicit casting (Widening)
   - conversion from a smaller to a larger type.

2. Explicit casting (Narrowing)
   - Manual conversion from larger to a smaller type.

Ex.

```
public class TypeCastingEx {
    public static void main(String[] args) {

        int num = 100;
        double db = num;
        System.out.println("Widening: "+db);

        double val = 99.99;
        int intval = (int) val;
        System.out.println("Narrowing: "+ intval);
    }
}
```

6. Discuss the concept of wrapper classes & their usage in Java.

→ Wrapper classes convert primitive data types into objects. Each primitive type has a corresponding wrapper class. (e.g. → int → Integer, double → Double)

Usage

1. Autoboxing & unboxing → automatic conversion between primitives & objects.
2. collections framework → used in data structures like ArrayList which work with object, not primitive.
3. Utility methods → Wrapper classes provide useful methods.

Ex.

```
Integer num = 10;
int val = num;
```

7. What is the difference between static & dynamic typing? Where does Java stand?

→

| | Static typing | Dynamic typing. |
|---|---|---|
| Definition | Type checking at compile-time. | Type checking at runtime. |
| Flexibility | Less flexible, but safer | more flexible, but risky |
| Error detection | catches errors early. | Errors appear at runtime. |
| Ex. → language | Java, C, C++ | Python, Javascript, Ruby |

- Java is statically typed meaning variables must have a defined type at compile-time, ensuring better performance.

# [Part 4]

1. What is JDK? How does it differ from JRE & JVM?

→ JDK refers to Java development kit. which contains both JRE & JVM i.e. Java Runtime environment and Java virtual machine.

JDK is used for developing Java applications.

| JRE | JVM |
|---|---|
| used for running Java applications. | Executes Java bytecode. |
| ~~JRE~~ → JVM + core libraries. | Just the engine that runs bytecode. |
| End-users (to run Java applications) | Internally used by JRE to execute Program. |

2. Explain the main components of JDK.

→ Java development kit.

1. JRE → it includes JVM & libraries to run Java applications.
2. JVM → converts the bytecode into machine code for execution.
3. Java compiler (javac) → converts Java source code into bytecode to compile the Java code.

4. Java Debugger – Helps in debugging java programs.
5. Java Archive Tool (Jar) – Packages java classes into JAR files.
6. Javadoc (Java documentation) – generates documentation from comments.

3. Describe the steps to install JDK and configure Java on your system.

→ 1. Download JDK.
- visit the oracle JDK website
- select the version required as per system.
- Download & run the installer.

2. Installation JDK.
- follow all the steps on installation wizard & complete the setup.
- Note the installation path.

3. Configure environment variables.
- open sys. properties -> advanced -> environment var.
- under sys. variable find path & add the JDK bin.
files \ java \ jdk-xx \ bin

4. Create a new sys. variable.
- Variable name - Java_Home.
- Variable value: C\ program files \ java \ jdk-xx

5. Verify installation.
- java -version.
- javac -version.

4. Write a simple Java program to print "Hello World!" & 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, which is the entry point of the program.

- public — accessible from anywhere
- static — Runs without creating an object
- void — Does not return a value.
- String[] args — command line arguments.

* System.out.println("Hello, World!").
↓
Prints "Hello, World!"

- System.out → Standard output stream.
- println() → Prints a new line.

5. What is the significance of the path and CLASSPATH environment variables in Java?

→

- PATH Environmental variable.
  - Specifies the directory of Java executables (java, javac).
  - Allows running Java commands from any location in the terminal or command prompt.

- CLASSPATH Environmental variables.
  - Specifies the location of Java class file & libraries.
  - Ensure Java can find & load required classes during execution.

6. What are the differences between Open JDK & oracle JDK?

⇒

| | Open JDK | Oracle JDK. |
|---|---|---|
| License | open source. | - free for development, but requires a subscription for commercial use. |
| cost | completely free. | - Paid for commercial use. |
| Updates | community driven updates | - offical oracle support with security patches. |
| Support | No offical suppor community-based. | - Enterprise-level support from oracle. |

7. Explain how Java programs are compiled & executed.

→ 1. Write the Java code
   - Create .java file containing Java code.
2. Compilation (javac).
   - The Java compiler (javac) converts the source code into bytecode (.class) file.

3. Execution (JVM).
- The JVM reads & executes the bytecode

8. What is Just-In-Time (JIT) compilation, & how does it improve Java performance?
→ - JUST IN TIME compilation is a technique used by the Java Virtual machine to improve performance by compiling bytecode into native machine code at runtime.

JIT improves Java performance.
1. faster execution.
- converts frequently used bytecode into machine code reducing interpretation.
2. Optimization
- ~~Uses techniques like~~
3. Adaptive compilation
- Detects & optimize hot spots for better speed.

9. Discuss the role of the Java Virtual machine (JVM) in program execution.
→ JVM is responsible for running Java programs by converting bytecode into machine code. ~~Executes~~ Provides essential runtime services.

Key responsibilities of JVM.
1. loads Bytecode
2. Interprets & execute Bytecode
3. memory management
4. security & Exception handling.
5. Platform Independence

Execution process → compilation, class loading & execution.