**Java Basics**

**─**

Contents

[**1. Introduction to Java** 2](#_Toc182658299)

[# How Java code is executed? 3](#_Toc182658300)

[**2. Arrays** 5](#_Toc182658301)

[**3. Strings** 8](#_Toc182658302)

[**4. Interview Questions** 10](#_Toc182658303)

## **1. Introduction to Java**

Java is a powerful, high-level, object-oriented programming language widely used for building applications that are secure, robust, and platform-independent

Here are a few reasons why Java is a good choice for beginners and professionals:

* **Platform Independence:** Java code can run on any machine with a Java Virtual Machine (JVM), making it highly portable.
* **Object-Oriented:** Java encourages modular programming and code reuse, making it easier to manage and scale.
* **Rich API and Libraries:** Java provides an extensive API and libraries that simplify complex tasks.
* **Security:** Java has built-in security features, making it a preferred choice for developing secure applications.
* **Community and Support:** Java has a vast community and extensive documentation.

Java Use cases: -

1. **Desktop Apps**:- Some GUI toolkits in java

* AWT( Abstract Windowing Tool)
* Swing
* JavaFX(advanced, have animation too)

1. **Enterprise Java**: - Java EE(Enterprise Edition), advanced application with web and database etc. Java EE is set of enterprise application geared towards enterprise application, so that you don’t need hundreds of application. Key components are:-

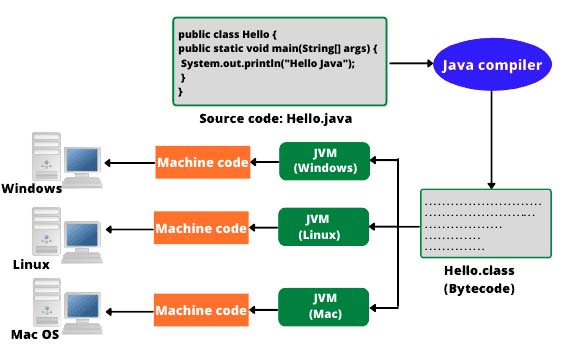
* **Java Server Pages (JSP):** Used to create dynamic web pages with embedded Java code, allowing for server-side processing and presentation logic.
* **Servlets:** Java classes that handle HTTP requests and responses, enabling dynamic web content generation.
* **Enterprise JavaBeans (EJBs):** Components that manage business logic, including transaction handling, security, and persistence, often used for complex business operations.
* **JPA (Java Persistence API):** A standard for managing object persistence in relational databases, simplifying database interactions.
* **JAX-RS (Java API for RESTful Web Services):** Enables the development of RESTful web services for interacting with applications via HTTP requests.
* **JMS (Java Message Service):** Facilitates asynchronous messaging between applications using message queues.

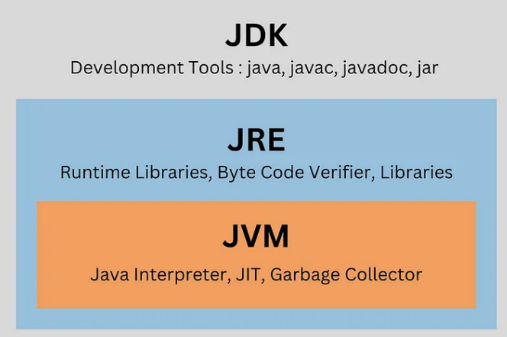
1. **Java Mobile Development**: - Using Android
2. **Java in cloud:** - web application development using microservices with frameworks like spring boot.

### # How Java code is executed?

Java code execution involves two main steps: compilation and interpretation:

* **Compilation**: The Java compiler (javac) translates the human-readable Java source code (.java file) into platform-independent bytecode, stored in a .class file.
* **Execution**: The Java Virtual Machine (JVM) reads the bytecode and converts it into machine-specific instructions using the Interpreter and Just-In-Time (JIT) compiler. This allows the program to run on any operating system with a JVM, ensuring Java's "write once, run anywhere" principle.



JDK vs JRE vs JVM: -

1. JDK is for developers to create Java applications.
2. JRE is for users to run Java applications.
3. JVM is the engine that powers Java applications, making them platform-independent.

**JDK (Java Development Kit):** Used for developing Java applications.

Components:

* Compiler (javac): Translates Java source code into bytecode.
* Interpreter (java): Executes the bytecode.
* Debugger: Helps identify and fix errors in code.
* Libraries (JAR files): Pre-written code for common tasks.
* Documentation: Provides information on Java APIs and language features.

**JRE (Java Runtime Environment):** Used for running Java applications.

Components:

* JVM (Java Virtual Machine): The core component that executes bytecode.
* Class libraries: Provides essential classes for input/output, networking, and other functionalities.

**JVM (Java Virtual Machine):** Executes Java bytecode

Key Features:

* Platform Independence: Allows Java programs to run on any platform with a JVM.
* Memory Management: Automatically handles memory allocation and deallocation.
* Security: Provides security features to protect against malicious code.
* Garbage Collection: Automatically reclaims unused memory.

First Java code: -

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

## **2. Arrays**

An array is a container object that holds a fixed number of values of a single data type. The length of an array is established when the array is created, and cannot be changed. (Arrays are immutable and not thread safe)

Declaration

// way to declare the array  
int[] arr;  
  
//ways to initialize the array  
int[] arr = new int[4];  
  
int[] arr = {1,2,3,4};

Default Values of Array Elements

* Numeric arrays are initialized to 0.
* boolean arrays are initialized to false.
* Reference type arrays (like String[]) are initialized to null.

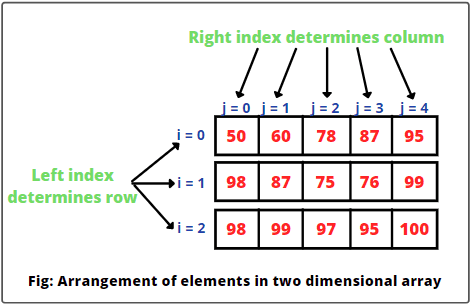
Basic functions

public class Array {  
 public static void main(String args[]){  
 int[] arr = {1,2,3,4};  
  
 //way to access the elements  
 System.*out*.println("Second Element of array is "+ arr[1]);  
  
 //Modifying the value  
 arr[0] = 24;  
  
 //printing the array  
 for(int i=0; i< arr.length; i++){ //output 24 2 3 4  
 System.*out*.println(arr[i]);  
 }  
  
 //For each loop  
 for(int ele:arr){  
 System.*out*.println(ele); //output 24 2 3 4  
 }  
 }  
}

Taking array as input from user

import java.util.\*;  
  
public class Array2 {  
 public static void main(String args[]){  
 Scanner scan = new Scanner(System.*in*);  
 int size = scan.nextInt();  
  
 int[] arr = new int[size];  
 //Getting array input from users  
 for(int i=0; i<size; i++){  
 arr[i] = scan.nextInt();  
 }  
 //Printing array  
 for(int ele: arr){  
 System.*out*.println(ele);  
 }  
  
 }  
}

2D Arrays



int[][] matrix = {  
 {1, 2, 3},  
 {4, 5, 6},  
 {7, 8, 9}  
};  
System.*out*.println(matrix[1][2]); // Output: 6

Array Utility class

Java provides the Arrays class with useful methods:

* sort(): Sorts the array in ascending order.
* binarySearch(): Searches for a specified value in a sorted array.
* copyOf(): Copies elements from an array.

import java.util.Arrays;  
  
public class LearnArrayUtil {  
 public static void main(String[] args) {  
  
 int[] arr = {5, 2, 8, 1};  
 Arrays.*sort*(arr); // Sorts array  
  
 //Convert array to string to print in 1 line  
 System.*out*.println(Arrays.*toString*(arr)); // Output: [1, 2, 5, 8]  
  
 int[] arr2 = {4, 3, 7};  
 int[] arr3 = {4, 3, 7};  
 System.*out*.println(Arrays.*compare*(arr,arr2)); //-1 arr is larger than arr2  
 System.*out*.println(Arrays.*compare*(arr2,arr3)); //0 > arrays are equal  
 }  
}

## **3. Strings**

* A String is a sequence of characters.
* Strings in Java are objects of the String class, which is part of java.lang package.
* Strings are immutable, meaning their content cannot be changed after they are created.

public class LearnString {  
 public static void main(String[] args) {  
  
 //Create string using literal  
 String s1 = "Hello";  
  
 //Creating string using new Keyword  
 String s2 = new String("World");  
  
 System.*out*.println(s1 + " " + s2);  
  
 }  
}

How String is stored in memory

* Java uses a String Pool to optimize memory (The String Pool is a special area in the Heap memory where Java stores string literals to optimize memory usage).
* If two strings have the same content and are created using literals, they share the same memory location in the pool.
* Strings created using the new keyword are stored in the heap.

public class LearnString {  
 public static void main(String[] args) {  
  
 String s1 = "Java";  
 String s2 = "Java";  
 String s3 = new String("Java");  
 String s4 = new String("Java");  
  
 System.*out*.println(s1 == s2); // true (both refer to the same object in the String pool)  
 System.*out*.println(s1 == s3); // false (different objects in memory)  
 System.*out*.println(s3 == s4); // false (different objects in memory)  
  
 }  
}

**Important methods of String: -**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| length() | Returns the length of the string | "Hello".length() → 5 |
| charAt(int index) | Returns the character at the specified index | "Java".charAt(1) → 'a' |
| substring(int start) | Extracts a substring from the string | "Hello".substring(2) → "llo" |
| substring(int start, int end) | Extracts a substring from start to end index | "Hello".substring(1, 4) → "ell" |
| equals(String s) | Compares strings for equality | "Java".equals("java") → false |
| equalsIgnoreCase(String s) | Compares strings, ignoring case | "Java".equalsIgnoreCase("java") → true |
| toLowerCase() | Converts the string to lowercase | "JAVA".toLowerCase() → "java" |
| toUpperCase() | Converts the string to uppercase | "java".toUpperCase() → "JAVA" |
| trim() | Removes leading/trailing whitespace | " Java ".trim() → "Java" |
| replace(oldChar, newChar) | Replaces characters in a string | "Hello".replace('l', 'p') → "Heppo" |

public class LearnString {  
 public static void main(String[] args) {  
 String str = "Java Programming";  
 String str2= new String("Java Programming");  
  
 System.*out*.println("Length: " + str.length()); //16  
 System.*out*.println("Substring: " + str.substring(2, 8)); //va Pro  
 System.*out*.println("Substring2: " + str.substring(5)); // Programming  
 System.*out*.println("Replaced: " + str.replace('a', 'o')); //Jovo Programming  
 System.*out*.println("Uppercase: " + str.toUpperCase()); //JAVA PROGRAMMING  
 System.*out*.println(str.equals(str2)); // true(as .equals check for content)  
 }  
}

## **4. Interview Questions**

**1. Why Are Strings Immutable?**

* Security: Prevents accidental or malicious modification of data (e.g., passwords).
* Thread Safety: Immutable strings can be shared between threads without synchronization issues.
* Memory Efficiency: Reusing strings in the pool saves memory.