**8. Arrays and Strings**

• **Theory:**

**1 . One-Dimensional and Multidimensional Arrays: -**

**One-Dimensional Arrays:-** A one-dimensional array in Java is a simple list of elements of the same type.

Declaration and Initialization.

**Eg:-** // Declaration

int[] array = new int[5]; // Array of integers with size 5

// Initialization

int[] numbers = {1, 2, 3, 4, 5}; //

**Multidimensional Arrays:-**

A **multidimensional array** in Java is essentially an array of arrays. The most common type is a two-dimensional array.

**Eg:-** // Declaration

int[][] matrix = new int[3][3]; // 3x3 matrix

// Initialization

int[][] grid = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

**2. String Handling in Java: String Class, StringBuffer, StringBuilder :-**

* **String handling** in Java involves several classes, each designed to work with strings in different ways.
* **String Class:-** Instances of the String class are immutable, meaning once created, their values cannot be changed. Any modification creates a new String object.

**Eg:-** String str1 = "Hello"; String str2 = new String("World");

* **StringBuffer** :- StringBuffer is mutable, meaning you can modify the contents without creating a new object. It is synchronized, making it safe for use in multithreaded environments.

**Eg:-** StringBuffer sb = new StringBuffer("Hello");

* **StringBuilder :-** Like StringBuffer, StringBuilder is also mutable. It is not synchronized, making it faster than StringBuffer for single-threaded scenarios.

**Eg:-** StringBuilder sb = new StringBuilder("Hello");

**3.Array of Objects:-** An array of objects is a collection of instances of a class stored in a single array structure. Each element of the array is a reference to an object of that class, allowing you to manage multiple objects conveniently.

**Eg:-** ClassName[] arrayName = new ClassName[size];

**4. String Methods (length, charAt, substring, etc.):-**

* **length():-** Returns the number of characters in the string.

**Eg:-** String str = "Hello";

int length = str.length(); // length = 5

* **2. charAt(int index):-** Returns the character at the specified index. Indexing starts at 0.

**Eg:-** String str = "Hello";

char ch = str.charAt(1); // ch = 'e'

* **3. substring(int beginIndex):-** Returns a new string that is a substring starting from the specified index to the end of the string.

**Eg:-** String str = "Hello, World!";

String sub = str.substring(7); // sub = "World!"

* **4. substring(int beginIndex, int endIndex):-** Returns a substring from the specified beginIndex (inclusive) to endIndex (exclusive).

**Eg:-** String str = "Hello, World!";

String sub = str.substring(0, 5); // sub = "Hello"

* **5. toUpperCase():-**Converts all characters in the string to uppercase.

**Eg:-** String str = "Hello";

String upper = str.toUpperCase(); // upper = "HELLO"

* **6. toLowerCase():-** Converts all characters in the string to lowercase.

**Eg:-** String str = "Hello";

String lower = str.toLowerCase(); // lower = "hello"

* **7. trim():-** Removes leading and trailing whitespace from the string.

**Eg:-** String str = " Hello ";

String trimmed = str.trim(); // trimmed = "Hello"

* **8. concat(String str):-** Concatenates the specified string to the end of this string.

**Eg:-** String str1 = "Hello";

String str2 = " World!";

String result = str1.concat(str2); // result = "Hello World!"

* **9. equals(Object obj):- Compares this string to the specified object for equality.**

**Eg:-** String str1 = "Hello";

String str2 = "Hello";

boolean isEqual = str1.equals(str2); // isEqual = true

* **10. equalsIgnoreCase(String anotherString):-** Compares this string to another string, ignoring case considerations.

**Eg:-** String str1 = "hello";

String str2 = "HELLO";

boolean isEqual = str1.equalsIgnoreCase(str2); // isEqual = true

* **11. indexOf(String str):-** Returns the index of the first occurrence of the specified substring or -1 if it does not occur.

**Eg:-** String str = "Hello, World!";

int index = str.indexOf("World"); // index = 7

* **12. replace(char oldChar, char newChar):-** Returns a new string with all occurrences of oldChar replaced by newChar.

**Eg:-** String str = "Hello, World!";

String replaced = str.replace('o', 'a'); // replaced = "Hella, Warld!"

* **13. split(String regex):-** Splits the string around matches of the given regular expression and returns an array of strings.

**Eg:-** String str = "apple,banana,cherry";

String[] fruits = str.split(","); // fruits = ["apple", "banana", "cherry"]