Module 4: Arrays & Strings

Single & Multi-dimensional Arrays

Single-dimensional Arrays:

- A single-dimensional array is a list of elements that are stored in contiguous memory locations and can be accessed using a single index.
- Syntax:
- int[] numbers = new int[5];
- numbers[0] = 10;
- Elements can be accessed and modified using the index.
- It is useful for linear data representation.

Example:

```
int[] arr = {1, 2, 3, 4, 5};

System.out.println(arr[2]); // Output: 3
```

Real World Example: *Storing and displaying marks of 5 subjects:*

```
public class MarksArray {
   public static void main(String[] args) {
      int[] marks = {85, 78, 90, 88, 76};
      for (int i = 0; i < marks.length; i++) {
         System.out.println("Subject " + (i+1) + ": " + marks[i]);
      }
   }
}
```

Output:

Subject 1: 85

Subject 2: 78

Subject 3: 90

Subject 4: 88

Subject 5: 76

Multi-dimensional Arrays:

- A multi-dimensional array is an array of arrays.

- The most common type is the two-dimensional array, which resembles a matrix.
- Syntax:
- int[][] matrix = new int[3][3];
- matrix[0][0] = 1;
- Accessed using multiple indices (e.g., matrix[i][j]).
- Useful in mathematical computations, grids, tables, etc.

Example:

```
int[][] matrix = {{1, 2}, {3, 4}};
System.out.println(matrix[1][0]); // Output: 3
```

Real World Example: *Storing seat arrangement in a classroom (2 rows and 3 columns):*

```
public class ClassroomSeats {
    public static void main(String[] args) {
        String[][] seats = {
            {"A1", "A2", "A3"},
            {"B1", "B2", "B3"}
        };
        for (int i = 0; i < seats.length; i++) {
            for (int j = 0; j < seats[i].length; j++) {
                System.out.print(seats[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
A1 A2 A3
B1 B2 B3
```

String Methods

- Strings in Java are objects of the String class.

- Java provides many built-in methods to manipulate strings:

Common Methods:

- length() - Returns the length of the string.
- charAt(int index) - Returns the character at the specified index.
- substring(int beginIndex, int endIndex) - Extracts a substring.
- equals(String another) - Compares two strings.
- equalsIgnoreCase(String another) - Compares two strings ignoring case.
- toLowerCase() / toUpperCase() - Converts string to lower/upper case.
- indexOf(String str) - Returns the index of the first occurrence of the specified substring.
- replace(char oldChar, char newChar) - Replaces all occurrences of a character.

Example:

```
String str = "Hello World";
System.out.println(str.length()); // Output: 11
System.out.println(str.substring(0, 5)); // Output: Hello
```

Real World Example: *Validating email format and formatting names:*

```
public class StringExample {
    public static void main(String[] args) {
        String name = "tOnY bAsKaR";
        String email = "tony@example.com";

        name = name.toLowerCase();
        name = name.substring(0, 1).toUpperCase() + name.substring(1);

        if (email.contains("@")) {
            System.out.println("Valid Email: " + email);
        } else {
            System.out.println("Invalid Email");
        }

        System.out.println("Formatted Name: " + name);
```

```
    }
}
```

Output:

Valid Email: tony@example.com

Formatted Name: Tony baskar

StringBuffer

- StringBuffer is a class used to create mutable (modifiable) strings.
- Unlike String, changes to StringBuffer objects do not create new objects.

Common Methods:

- append(String str) - Adds the given string to the end.
- insert(int offset, String str) - Inserts the string at the specified index.
- replace(int start, int end, String str) - Replaces characters in a substring.
- delete(int start, int end) - Deletes a portion of the string.
- reverse() - Reverses the contents of the string.

Example:

```
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");
System.out.println(sb); // Output: Hello World
sb.reverse();
System.out.println(sb); // Output: dlroW olleH
```

Real World Example: *Building and formatting a dynamic report message:*

```
public class ReportGenerator {
    public static void main(String[] args) {
        StringBuffer report = new StringBuffer();
        report.append("Student Report\n");
        report.append("Name: Tony\n");
        report.append("Marks: 92\n");
        report.insert(0, "*****\n");
        report.append("*****");
```

```
        System.out.println(report);
    }
}
```

Output:

```
*****
Student Report
Name: Tony
Marks: 92
*****
```

Use Case:

- Preferred when frequent modifications are required in a string (e.g., in loops or text editors).