LAB # 01

INTRODUCTION TO STRING POOL, LITERALS, AND WRAPPER CLASSES

OBJECTIVE: To study the concepts of String Constant Pool, String literals, String immutability and Wrapper classes.

LAB TASKS

1. Write a program that initialize five different strings using all the above mentioned ways

   a) string literals b) new keyword c) also use intern method and show string immutability.

Source code

```
public class StringExample { // Define a public class named StringExample
    public static void main(String[] args) { // Main method where the program execution begins

        // a) Initializing strings using string literals
        String str1 = "SHEIKH"; // Create a string literal "Hello" and assign it to str1
        String str2 = "FATIMA"; // Create a string literal "World" and assign it to str2
        String str3 = "DILSHAD"; // Create a string literal "World" and assign it to str3
        // b) Initializing strings using the new keyword
        String str4 = new String("Java"); // Create a new String object with value "Java" and assign it to str4
        String str5 = new String("Programming"); // Create a new String object with value "Programming" and assign it to str5

        // c) Using the intern method
        String str6 = str1.intern(); // Create a new String object with value "Intern", then intern it to reference the canonical
            string

        // Displaying the initialized strings
        System.out.println("String 1: " + str1); // Print the value of str1
        System.out.println("String 2: " + str2); // Print the value of str2
        System.out.println("String 3: " + str3); // Print the value of str3
        System.out.println("String 4: " + str4); // Print the value of str4
        System.out.println("String 5: " + str5); // Print the value of str5
        System.out.println("String 6: " + str6); // Print the value of str6

        // Illustrating string immutability
        str1 = str1.concat(" World!"); // Concatenate " World!" to str1 and assign the new string back to str1
        System.out.println("After modification, String 1: " + str1); // Print the modified value of str1
        System.out.println("Original String 2 remains unchanged: " + str2); // Print str2 to show it remains unchanged
    }
}
```

output

```
String 1: SHEIKH
String 2: FATIMA
String 3: DILSHAD
String 4: Java
String 5: Programming
String 6: SHEIKH
After modification, String 1: SHEIKH World!
Original String 2 remains unchanged: FATIMA

=== Code Execution Successful ===
```

2. Write a program to convert primitive data type Double into its respective wrapper object.

Source code

```
public class DoubleWrapperExample { // Define a public class named DoubleWrapperExample
    public static void main(String[] args) { // Main method where the program execution begins

        // Step 1: Declare a primitive double variable
        double primitiveDouble = 40.5; // Initialize a primitive double with a value of 10.5

        // Step 2: Convert the primitive double to its wrapper object Double
        Double wrapperDouble = Double.valueOf(primitiveDouble); // Use the valueOf method to convert primitive to wrapper

        // Step 3: Display the values
        System.out.println("Primitive double: " + primitiveDouble); // Print the primitive double value
        System.out.println("Wrapper Double: " + wrapperDouble); // Print the wrapper Double object
    }
}
```

output

```
Primitive double: 40.5
Wrapper Double: 40.5

=== Code Execution Successful ===
```

3. Write a program that initialize five different strings and perform the following operations. a. Concatenate all five stings. Convert fourth string to uppercase. Find the substring from the concatenated string from 8 to onward

Source code ***output***

```java
public class StringOperationsExample { // Define a public class named StringOperationsExample
    public static void main(String[] args) { // Main method where the program execution begins

        // Step 1: Initialize five different strings
        String str1 = "Java"; // First string
        String str2 = "is"; // Second string
        String str3 = "a"; // Third string
        String str4 = "powerful"; // Fourth string
        String str5 = "language"; // Fifth string

        // Step 2: Concatenate all five strings
        String concatenatedString = str1 + " " + str2 + " " + str3 + " " + str4 + " " + str5; // Concatenate with
        System.out.println("Concatenated String: " + concatenatedString); // Print the concatenated string

        // Step 3: Convert the fourth string to uppercase
        String upperCaseStr4 = str4.toUpperCase(); // Convert str4 to uppercase
        System.out.println("Fourth String in Uppercase: " + upperCaseStr4); // Print the uppercase string

        // Step 4: Find the substring from the concatenated string from index 8 onward
        String substring = concatenatedString.substring(8); // Get substring starting from index 8
        System.out.println("Substring from index 8 onward: " + substring); // Print the substring
    }
}
```

```
Concatenated String: Java is a powerful language
Fourth String in Uppercase: POWERFUL
Substring from index 8 onward: a powerful language

=== Code Execution Successful ===
```

4. You are given two strings word1 and word2. Merge the strings by adding letters in alternating order, starting with word1. If a string is longer than the other, append the additional letters onto the end of the merged string. Return *the merged string.*

Source code ***output***

```java
public class MergeStrings { // Define a public class named MergeStrings
    public static void main(String[] args) { // Main method where the program execution begins

        // Step 1: Initialize the input strings
        String word1 = "abc"; // First input string
        String word2 = "pqr"; // Second input string

        // Step 2: Call the merge function and store the result
        String mergedString = mergeAlternately(word1, word2); // Merge the strings
        System.out.println("Merged String: " + mergedString); // Print the merged string
    }

    // Method to merge two strings in alternating order
    public static String mergeAlternately(String word1, String word2) { // Define the merge method
        StringBuilder merged = new StringBuilder(); // Create a StringBuilder to hold the merged result
        int length1 = word1.length(); // Get the length of the first string
        int length2 = word2.length(); // Get the length of the second string
        int minLength = Math.min(length1, length2); // Find the minimum length of the two strings

        // Step 3: Merge characters from both strings up to the length of the shorter string
        for (int i = 0; i < minLength; i++) { // Loop through the characters
            merged.append(word1.charAt(i)); // Append character from word1
            merged.append(word2.charAt(i)); // Append character from word2
        }

        // Step 4: Append any remaining characters from the longer string
        if (length1 > minLength) { // If word1 is longer
            merged.append(word1.substring(minLength)); // Append remaining characters from word1
        } else if (length2 > minLength) { // If word2 is longer
            merged.append(word2.substring(minLength)); // Append remaining characters from word2
        }

        return merged.toString(); // Convert StringBuilder to String and return
    }
}
```

```
Merged String: apbqcr

=== Code Execution Successful ===
```

5. Write a Java program to find the minimum and maximum values of Integer**,** Float**,** and Double using the respective wrapper class constants.

Source code ***output***

```java
public class MinMaxValues { // Define the class
    public static void main(String[] args) { // Main method
        // Get the minimum and maximum values for Integer
        int minInt = Integer.MIN_VALUE; // Minimum value of Integer
        int maxInt = Integer.MAX_VALUE; // Maximum value of Integer

        // Get the minimum and maximum values for Float
        float minFloat = Float.MIN_VALUE; // Minimum value of Float (smallest positive value)
        float maxFloat = Float.MAX_VALUE; // Maximum value of Float

        // Get the minimum and maximum values for Double
        double minDouble = Double.MIN_VALUE; // Minimum value of Double (smallest positive value)
        double maxDouble = Double.MAX_VALUE; // Maximum value of Double

        // Print the results
        System.out.println("Integer Min: " + minInt); // Print minimum Integer value
        System.out.println("Integer Max: " + maxInt); // Print maximum Integer value
        System.out.println("Float Min: " + minFloat); // Print minimum Float value
        System.out.println("Float Max: " + maxFloat); // Print maximum Float value
        System.out.println("Double Min: " + minDouble); // Print minimum Double value
        System.out.println("Double Max: " + maxDouble); // Print maximum Double value
    }
}
```

```
Integer Min: -2147483648
Integer Max: 2147483647
Float Min: 1.4E-45
Float Max: 3.4028235E38
Double Min: 4.9E-324
Double Max: 1.7976931348623157E308

=== Code Execution Successful ===
```

HOME TASKS

1. Write a JAVA program to perform Autoboxing and also implement different methods of wrapper class.

```java
public class AutoboxingExample {
    public static void main(String[] args) {
        // Autoboxing: converting primitive int to Integer
        int primitiveInt = 10; // Declare a primitive int variable
        Integer wrappedInt = primitiveInt; // Autoboxing: convert int to Integer object
        System.out.println("Autoboxed Integer: " + wrappedInt); // Print the autoboxed Integer

        // Autoboxing: converting primitive double to Double
        double primitiveDouble = 20.5; // Declare a primitive double variable
        Double wrappedDouble = primitiveDouble; // Autoboxing: convert double to Double object
        System.out.println("Autoboxed Double: " + wrappedDouble); // Print the autoboxed Double

        // Unboxing: converting Integer back to int
        Integer anotherWrappedInt = 30; // Autoboxing: create an Integer object
        int unboxedInt = anotherWrappedInt; // Unboxing: convert Integer back to int
        System.out.println("Unboxed Integer: " + unboxedInt); // Print the unboxed int

        // Using Integer wrapper class methods
        System.out.println("Maximum Integer Value: " + Integer.MAX_VALUE); // Print max value of Integer
        System.out.println("Minimum Integer Value: " + Integer.MIN_VALUE); // Print min value of Integer
        System.out.println("Integer to String: " + Integer.toString(wrappedInt)); // Convert Integer to String
        System.out.println("String to Integer: " + Integer.parseInt("100")); // Convert String to Integer

        // Using Double wrapper class methods
        System.out.println("Maximum Double Value: " + Double.MAX_VALUE); // Print max value of Double
        System.out.println("Minimum Double Value: " + Double.MIN_VALUE); // Print min value of Double
        System.out.println("Double to String: " + Double.toString(wrappedDouble)); // Convert Double to String
        System.out.println("String to Double: " + Double.parseDouble("45.67")); // Convert String to Double

        // Comparing two Integer values
        Integer int1 = 100; // Create an Integer object with value 100
        Integer int2 = 200; // Create another Integer object with value 200
        // Compare int1 and int2 and print the result
        System.out.println("Comparison of " + int1 + " and " + int2 + ": " + Integer.compare(int1, int2));
    }
}
```

```
Autoboxed Integer: 10
Autoboxed Double: 20.5
Unboxed Integer: 30
Maximum Integer Value: 2147483647
Minimum Integer Value: -2147483648
Integer to String: 10
String to Integer: 100
Maximum Double Value: 1.7976931348623157E308
Minimum Double Value: 4.9E-324
Double to String: 20.5
String to Double: 45.67
Comparison of 100 and 200: -1

=== Code Execution Successful ===
```

2. Write a Java program to count the number of even and odd digits in a given integer using Autoboxing and Unboxing.

```java
import java.util.Scanner; // Import Scanner for user input

public class EvenOddDigitCounter { // Class definition
    public static void main(String[] args) { // Main method
        Scanner scanner = new Scanner(System.in); // Create Scanner object for input

        System.out.print("Enter an integer: "); // Prompt user for input
        Integer number = scanner.nextInt(); // Read input and autobox to Integer

        int evenCount = 0; // Initialize even digit counter
        int oddCount = 0; // Initialize odd digit counter

        // Process each digit in the number
        while (number != 0) { // Loop until number becomes 0
            int digit = number % 10; // Get the last digit
            number /= 10; // Remove the last digit

            // Check if the digit is even or odd
            if (digit % 2 == 0) { // If digit is even
                evenCount++; // Increment even counter
            } else { // If digit is odd
                oddCount++; // Increment odd counter
            }
        }

        // Display the results
        System.out.println("Even digits count: " + evenCount); // Print even count
        System.out.println("Odd digits count: " + oddCount); // Print odd count

        scanner.close(); // Close the scanner
    }
}
```

```
Enter an integer: 5674
Even digits count: 2
Odd digits count: 2

=== Code Execution Successful ===
```

3. Write a Java program to find the absolute value, square root, and power of a number using Math class methods, while utilizing Autoboxing and Wrapper classes.

```java
import java.util.Scanner; // Import Scanner for user input

public class MathOperations { // Class definition
    public static void main(String[] args) { // Main method
        Scanner scanner = new Scanner(System.in); // Create Scanner object

        System.out.print("Enter a number: "); // Prompt user for input
        Double number = scanner.nextDouble(); // Read input and autobox to Double

        // Calculate absolute value
        Double absoluteValue = Math.abs(number); // Use Math.abs() method

        // Calculate square root
        Double squareRoot = Math.sqrt(number); // Use Math.sqrt() method

        // Calculate power (number raised to 2)
        Double power = Math.pow(number, 2); // Use Math.pow() method

        // Display results
        System.out.println("Absolute Value: " + absoluteValue); // Print absolute value
        System.out.println("Square Root: " + squareRoot); // Print square root
        System.out.println("Power (number^2): " + power); // Print power

        scanner.close(); // Close the scanner
    }
}
```

```
Enter a number: -14
Absolute Value: 14.0
Square Root: NaN
Power (number^2): 196.0

=== Code Execution Successful ===
```

4. Write a Java program to **reverse only the vowels** in a string.

```java
import java.util.Scanner; // Import the Scanner class for user input

public class ReverseVowels { // Define the class named ReverseVowels
    public static void main(String[] args) { // Main method where program execution begins
        Scanner scanner = new Scanner(System.in); // Create a Scanner object to read input from the user
        System.out.print("Enter a string: "); // Prompt the user to enter a string
        String input = scanner.nextLine(); // Read the entire line of input from the user
        // Call the reverseVowels method and print the result
        System.out.println("Reversed vowels: " + reverseVowels(input));
        scanner.close(); // Close the scanner to prevent resource leaks
    }

    // Method to reverse the vowels in the given string
    public static String reverseVowels(String s) {
        char[] chars = s.toCharArray(); // Convert the input string to a character array for manipulation
        int left = 0; // Initialize the left pointer at the start of the array
        int right = chars.length - 1; // Initialize the right pointer at the end of the array
        String vowels = "aeiouAEIOU"; // String containing all vowels (both lowercase and uppercase)

        // Loop until the left pointer is less than the right pointer
        while (left < right) {
            // Check if the character at the left pointer is not a vowel
            if (vowels.indexOf(chars[left]) == -1) {
                left++; // Move the left pointer to the right
            }
            // Check if the character at the right pointer is not a vowel
            else if (vowels.indexOf(chars[right]) == -1) {
                right--; // Move the right pointer to the left
            }
            // If both pointers point to vowels, swap them
            else {
                // Swap the vowels at the left and right pointers
                char temp = chars[left]; // Store the left vowel in a temporary variable
                chars[left] = chars[right]; // Replace the left vowel with the right vowel
                chars[right] = temp; // Replace the right vowel with the left vowel
                left++; // Move the left pointer to the right
                right--; // Move the right pointer to the left
            }
        }
        return new String(chars); // Convert the modified character array back to a string and return it
    }
}
```

```
Enter a string: Sheikh Fatima Dilshad
Reversed vowels: Shaikh Fatima Dilshed

=== Code Execution Successful ===
```

5. Write a Java program to **find the longest word** in a sentence.

```java
import java.util.Scanner; // Import the Scanner class for user input

public class LongestWordFinder { // Define the class
    public static void main(String[] args) { // Main method
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
        System.out.print("Enter a sentence: "); // Prompt the user to enter a sentence
        String input = scanner.nextLine(); // Read the entire line of input
        String longestWord = findLongestWord(input); // Call the method to find the longest word
        System.out.println("The longest word is: " + longestWord); // Print the longest word
        scanner.close(); // Close the scanner to prevent resource leaks
    }

    // Method to find the longest word in a given sentence
    public static String findLongestWord(String sentence) {
        String[] words = sentence.split(" "); // Split the sentence into words using space as a delimiter
        String longest = ""; // Initialize an empty string to hold the longest word

        // Loop through each word in the array
        for (String word : words) {
            // Check if the current word is longer than the longest found so far
            if (word.length() > longest.length()) {
                longest = word; // Update the longest word
            }
        }
        return longest; // Return the longest word found
    }
}
```

```
Enter a sentence: Ezzah, Do You Want To Meet Software Enginweer ??
The longest word is: Enginweer

=== Code Execution Successful ===
```

LAB # 02

ArrayList and Vector in JAVA

OBJECTIVE: To implement ArrayList and Vector.

Lab Tasks

1. Write a program that initializes Vector with 10 integers in it. Display all the integers and sum of these integers.

CODE:

```java
import java.util.Vector;

public class VectorExample {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();
        int sum = 0;

        // Initializing Vector with 10 integers
        for (int i = 1; i <= 10; i++) {
            numbers.add(i * 10); // Adding multiples of 10 for variety
            sum += i * 10;
        }

        // Displaying all elements in the Vector
        System.out.println("Vector Elements: " + numbers);

        // Displaying the sum
        System.out.println("Sum of Vector Elements: " + sum);
    }
}
```

OUTPUT:

```
Vector Elements: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Sum of Vector Elements: 550
```

2. Create a ArrayList of string. Write a menu driven program which:

a. Displays all the elements
b. Displays the largest String

CODE:

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class StringListMenu {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Shahzaib");
        names.add("Jazbia");
        names.add("Fatima");
        names.add("Hashir");
        names.add("Kisa");

        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Display All Elements");
            System.out.println("2. Display the Largest String");
            System.out.println("0. Exit");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.println("Names in ArrayList: " + names);
                    break;
                case 2:
                    String largestString = Collections.max(names, (a, b) -> a.length() -
                        .length());
                    System.out.println("Largest String: " + largestString);
                    break;
                case 0:
                    System.out.println("Exiting.");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        } while (choice != 0);

        scanner.close();
    }
```

OUTPUT:

```
Menu:
1. Display All Elements
2. Display the Largest String
0. Exit
```

3. Create a Arraylist storing Employee details including Emp_id, Emp_Name, Emp_gender, Year_of_Joining (you can also add more attributes including these). Then sort the employees according to their joining year using Comparator and Comparable interfaces.

CODE:

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Employee implements Comparable<Employee> {
    int emp_id;
    String emp_name;
    String emp_gender;
    int year_of_joining;

    public Employee(int emp_id, String emp_name, String emp_gender, int year_of_joining)
        {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_gender = emp_gender;
        this.year_of_joining = year_of_joining;
    }

    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.year_of_joining, other.year_of_joining);
    }

    @Override
    public String toString() {
        return "Employee{" +
                "ID=" + emp_id +
                ", Name='" + emp_name + '\'' +
                ", Gender='" + emp_gender + '\'' +
                ", Year of Joining=" + year_of_joining +
                '}';
```

```java
    }
}

public class EmployeeSorting {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee(101, "Shahzaib", "Male", 2018));
        employees.add(new Employee(102, "Jazbia", "Female", 2020));
        employees.add(new Employee(103, "Fatima", "Female", 2017));
        employees.add(new Employee(104, "Hashir", "Male", 2019));
        employees.add(new Employee(105, "Kisa", "Female", 2015));

        Collections.sort(employees); // Sorting using Comparable interface

        System.out.println("Employees sorted by Year of Joining:");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

OUTPUT:

```
Employees sorted by Year of Joining:
Employee{ID=105, Name='Kisa', Gender='Female', Year of Joining=2015}
Employee{ID=103, Name='Fatima', Gender='Female', Year of Joining=2017}
Employee{ID=101, Name='Shahzaib', Gender='Male', Year of Joining=2018}
Employee{ID=104, Name='Hashir', Gender='Male', Year of Joining=2019}
Employee{ID=102, Name='Jazbia', Gender='Female', Year of Joining=2020}
```

4. Write a program that initializes Vector with 10 integers in it.

- Display all the integers
- Sum of these integers.
- Find Maximum Element in Vector

CODE:

```java
import java.util.Collections;
import java.util.Vector;

public class VectorTasks {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();
        int sum = 0;

        // Initializing Vector with 10 integers
        for (int i = 1; i <= 10; i++) {
            numbers.add(i * 5); // Adding multiples of 5
            sum += i * 5;
        }

        // Display all integers
        System.out.println("Vector Elements: " + numbers);

        // Display sum
        System.out.println("Sum of Vector Elements: " + sum);

        // Find Maximum Element
        int maxElement = Collections.max(numbers);
        System.out.println("Maximum Element in Vector: " + maxElement);
    }
}

```

OUTPUT:

```
Vector Elements: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
Sum of Vector Elements: 275
Maximum Element in Vector: 50
```

5. Find the k-th smallest element in a sorted ArrayList

CODE:

```java
import java.util.ArrayList;
import java.util.Collections;

public class KthSmallestElement {
    public static int findKthSmallest(ArrayList<Integer> list, int k) {
        Collections.sort(list); // Sort the list
        return list.get(k - 1); // k-th smallest (1-based index)
    }

    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        for (int i = 1; i <= 20; i++) {
            numbers.add(i * 3); // Adding multiples of 3 for variety
        }

        int k = 5; // 5th smallest
        System.out.println("The " + k + "-th smallest element is: " +
            findKthSmallest(numbers, k));
    }
}
```

OUTPUT:

```
The 5-th smallest element is: 15
```

6. Write a program to merge two ArrayLists into one.

CODE:

```java
import java.util.ArrayList;

public class MergeArrayLists {
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("Shahzaib");
        list1.add("Jazbia");
        list1.add("Fatima");

        ArrayList<String> list2 = new ArrayList<>();
        list2.add("Hashir");
        list2.add("Kisa");

        ArrayList<String> mergedList = new ArrayList<>(list1);
        mergedList.addAll(list2);

        System.out.println("Merged ArrayList: " + mergedList);
    }
}
```

OUTPUT:

```
Merged ArrayList: [Shahzaib, Jazbia, Fatima, Hashir, Kisa]
```

Home Tasks

1. Create a Vector storing integer objects as an input.
    a. Sort the vector
    b. Display largest number
    c. Display smallest number

CODE:

```java
import java.util.Collections;
import java.util.Vector;

public class VectorHomeTask {
    public static void main(String[] args) {
        Vector<Integer> numbers = new Vector<>();
        Collections.addAll(numbers, 15, 3, 25, 8, 14, 20, 5, 17, 1, 10
            );

        // Sorting Vector
        Collections.sort(numbers);
        System.out.println("Sorted Vector: " + numbers);

        // Display largest and smallest number
        int largest = Collections.max(numbers);
        int smallest = Collections.min(numbers);
        System.out.println("Largest Number: " + largest);
        System.out.println("Smallest Number: " + smallest);
    }
}
```

OUTPUT:

```
Sorted Vector: [1, 3, 5, 8, 10, 14, 15, 17, 20, 25]
Largest Number: 25
Smallest Number: 1
```

2. Write a java program which takes user input and gives hashcode value of those inputs using hashCode () method.

CODE:

```java
import java.util.Scanner;

public class HashCodeExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string to find its hash code: ");
        String userInput = scanner.nextLine();

        System.out.println("Hash Code of \"" + userInput + "\": " +
            userInput.hashCode());
        scanner.close();
    }
}
```

OUTPUT:

```
Enter a string to find its hash code: jazbia
Hash Code of "jazbia": -1166963721
```

3. Scenario based

Create a java project, suppose you work for a company that needs to manage a list of employees. Each employee has a unique combination of a name and an ID. Your goal is to ensure that you can track employees effectively and avoid duplicate entries in your system.

Requirements

a. Employee Class: You need to create an Employee class that includes:

- name: The employee's name (String).
- id: The employee's unique identifier (int).
- Override the hashCode() and equals() methods to ensure that two employees are considered equal if they have the same name and id.

b. Employee Management: You will use a HashSet to store employee records. This will help you avoid duplicate entries.
c. Operations: Implement operations to:

- Add new employees to the record.
- Check if an employee already exists in the records.
- Display all employees.

SENARIO A: (CODE)

```java
import java.util.HashSet;
import java.util.Objects;
import java.util.Scanner;

class Employee {
    String name;
    int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof Employee)) return false;
        Employee emp = (Employee) obj;
        return id == emp.id && name.equals(emp.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, id);
```

```
25      }
26
27      @Override
28      public String toString() {
29          return "Employee{ID=" + id + ", Name='" + name + "'}";
30      }
31  }
32
33  public class EmployeeManagement {
34      public static void main(String[] args) {
35          HashSet<Employee> employees = new HashSet<>();
36          Scanner scanner = new Scanner(System.in);
37
38          while (true) {
39              System.out.println("1. Add Employee\n2. Check if Employee
                    Exists\n3. Display Employees\n0. Exit");
40              int choice = scanner.nextInt();
41              scanner.nextLine();  // Consume newline
42
43              if (choice == 1) {
44                  System.out.print("Enter Employee Name: ");
45                  String name = scanner.nextLine();
46                  System.out.print("Enter Employee ID: ");
47                  int id = scanner.nextInt();
```

```
48
49              Employee emp = new Employee(name, id);
50              if (employees.add(emp)) {
51                  System.out.println("Employee added.");
52              } else {
53                  System.out.println("Employee already exists.");
54              }
55
56          } else if (choice == 2) {
57              System.out.print("Enter Employee Name to Check: ");
58              String name = scanner.nextLine();
59              System.out.print("Enter Employee ID to Check: ");
60              int id = scanner.nextInt();
61
62              Employee empToCheck = new Employee(name, id);
63              if (employees.contains(empToCheck)) {
64                  System.out.println("Employee exists in the
                        records.");
65              } else {
66                  System.out.println("Employee does not exist in
                        the records.");
67              }
68          } else if (choice == 3) {
69              System.out.println("Employee Records:");
70              for (Employee emp : employees) {
71                  System.out.println(emp);
72              }
73          } else if (choice == 0) {
74              System.out.println("Exiting.");
75              break;
76          } else {
77              System.out.println("Invalid choice. Please try again
                    .");
78          }
79      }
80
81      scanner.close();
82  }
83 }
84
85
```

OUTPUT:

```
javac EmployeeManagement.java && java -Xmx1024M -Xms256MEmployeeManagement
1. Add Employee
2. Check if Employee Exists
3. Display Employees
0. Exit
```

4.Create a Color class that has red, green, and blue values. Two colors are considered equal if their RGB values are the same

CODE:

```java
import java.util.HashSet;
import java.util.Objects;
import java.util.Scanner;

class Color {
    private int red;
    private int green;
    private int blue;

    // Constructor
    public Color(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }

    // Overriding equals to compare based on RGB values
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Color color = (Color) obj;
        return red == color.red && green == color.green && blue == color
            .blue;
    }
```

```
24      }
25
26      // Overriding hashCode to generate hash based on RGB values
27      @Override
28      public int hashCode() {
29          return Objects.hash(red, green, blue);
30      }
31
32      @Override
33      public String toString() {
34          return "Color { R: " + red + ", G: " + green + ", B: " + blue + " }"
                ;
35      }
36  }
37
38  public class RGBColorProgram {
39      public static void main(String[] args) {
40          HashSet<Color> colors = new HashSet<>();
41          Scanner scanner = new Scanner(System.in);
42          int choice;
43
44          do {
45              System.out.println("\nMenu:");
46              System.out.println("1. Add new color");
47              System.out.println("2. Check if a color exists");
48              System.out.println("3. Display all colors");
49              System.out.println("4. Exit");
50              System.out.print("Enter your choice: ");
51              choice = scanner.nextInt();
52
53              switch (choice) {
54                  case 1:
55                      System.out.print("Enter red value (0-255): ");
56                      int red = scanner.nextInt();
57                      System.out.print("Enter green value (0-255): ");
58                      int green = scanner.nextInt();
59                      System.out.print("Enter blue value (0-255): ");
60                      int blue = scanner.nextInt();
61
62                      Color newColor = new Color(red, green, blue);
63                      if (colors.add(newColor)) {
64                          System.out.println("Color added successfully.");
65                      } else {
66                          System.out.println("Color already exists.");
67                      }
68                      break;
69
70                  case 2:
71                      System.out.print("Enter red value (0-255): ");
```

```
72                  int checkRed = scanner.nextInt();
73                  System.out.print("Enter green value (0-255): ");
74                  int checkGreen = scanner.nextInt();
75                  System.out.print("Enter blue value (0-255): ");
76                  int checkBlue = scanner.nextInt();
77
78                  Color checkColor = new Color(checkRed, checkGreen,
                        checkBlue);
79                  if (colors.contains(checkColor)) {
80                      System.out.println("Color exists.");
81                  } else {
82                      System.out.println("Color not found.");
83                  }
84                  break;
85
86              case 3:
87                  System.out.println("All Colors:");
88                  if (colors.isEmpty()) {
89                      System.out.println("No colors available.");
90                  } else {
91                      for (Color color : colors) {
92                          System.out.println(color);
93                      }
94                  }
95                  break;
96
97              case 4:
98                  System.out.println("Exiting...");
99                  break;
100
101             default:
102                 System.out.println("Invalid choice. Please try again.");
103         }
104     } while (choice != 4);
105
106     scanner.close();
107 }
```

OUTPUT:

```
java -cp /tmp/b1NEi8PIcQ/RGBColorProgram

Menu:
1. Add new color
2. Check if a color exists
3. Display all colors
4. Exit
Enter your choice: 1
Enter red value (0-255): 1
Enter green value (0-255): 1
Enter blue value (0-255): 1
Color added successfully.

Menu:
1. Add new color
2. Check if a color exists
3. Display all colors
4. Exit
Enter your choice:
```

LAB # 03 RECURSION

OBJECTIVE: To understand the complexities of the recursive functions and a way to reduce these complexities.

LAB TASK

1. Write a program which takes an integer value (k) as input and prints the sequence of numbers from k to 0 in descending order.

CODE:

```java
import java.util.Scanner;

public class DescendingSequence {
    public static void displayDescending(int jazbia) {
        if (jazbia < 0) return;
        System.out.print(jazbia + " ");
        displayDescending(jazbia - 1);
    }

    public static void main(String[] args) {
        Scanner fatima = new Scanner(System.in);
        System.out.print("Enter an integer (k): ");
        int jazbia = fatima.nextInt();
        System.out.print("Sequence from " + jazbia + " to 0: ");
        displayDescending(jazbia);
        fatima.close();
    }
}
```

OUTPUT:

```
java -cp /tmp/l1zCo6n1ou/DescendingSequence
Enter an integer (k): 87
Sequence from 87 to 0: 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70
    69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46
    45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22
    21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
=== Code Execution Successful ===
```

2. Write a program to reverse your full name using Recursion.

CODE:

```java
import java.util.Scanner;

public class NameReversal {
    public static String reverse(String fatima) {
        if (fatima.length() == 0) return "";
        return reverse(fatima.substring(1)) + fatima.charAt(0);
    }

    public static void main(String[] args) {
        Scanner jazbia = new Scanner(System.in);
        System.out.print("Enter your full name: ");
        String fatima = jazbia.nextLine();
        String reversedName = reverse(fatima);
        System.out.println("Reversed Name: " + reversedName);
        jazbia.close();
    }
}

```

OUTPUT:

```
java -cp /tmp/GMcZWsJjav/NameReversal
Enter your full name: jazbia
jazbia
Reversed Name: aibzaj

=== Code Execution Successful ===
```

3. Write a program to calculate the sum of numbers from 1 to N using recursion. N should be user input.

CODE:

```java
import java.util.Scanner;

public class SumUpToN {
    public static int sumToN(int jazbia) {
        if (jazbia <= 0) return 0;
        return jazbia + sumToN(jazbia - 1);
    }

    public static void main(String[] args) {
        Scanner fatima = new Scanner(System.in);
        System.out.print("Enter a positive integer (N): ");
        int jazbia = fatima.nextInt();
        System.out.println("Sum from 1 to " + jazbia + ": " + sumToN
            (jazbia));
        fatima.close();
    }
}
```

OUTPUT:

```
java -cp /tmp/i8nc7xVzie/SumUpToN
Enter a positive integer (N): 17
Sum from 1 to 17: 153

=== Code Execution Successful ===
```

4. Write a recursive program to calculate the sum of elements in an array.

CODE:

```java
public class ArraySumRecursion {
    public static int sumElements(int[] jazbia, int fatima) {
        if (fatima < 0) return 0;
        return jazbia[fatima] + sumElements(jazbia, fatima - 1);
    }

    public static void main(String[] args) {
        int[] jazbia = {2, 4, 6, 8, 10};
        int total = sumElements(jazbia, jazbia.length - 1);
        System.out.println("Sum of array elements: " + total);
    }
}
```

OUTPUT:

```
java -cp /tmp/zmVQBl9EWT/ArraySumRecursion
Sum of array elements: 30

=== Code Execution Successful ===
```

5. Write a recursive program to calculate the factorial of a given integer n
CODE:

```java
import java.util.Scanner;

public class RecursiveFactorial {
    public static int factorial(int jazbia) {
        if (jazbia <= 1) return 1;
        return jazbia * factorial(jazbia - 1);
    }

    public static void main(String[] args) {
        Scanner fatima = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        int jazbia = fatima.nextInt();
        System.out.println("Factorial of " + jazbia + ": " + factorial
            (jazbia));
        fatima.close();
    }
}
```

OUTPUT:

```
java -cp /tmp/4Mpk2lODBi/RecursiveFactorial
Enter a positive integer: 98
Factorial of 98: 0

=== Code Execution Successful ===
```

6. Write a program to count the digits of a given number using recursion.
CODE:

```java
import java.util.Scanner;

public class DigitCounter {
    public static int countDigits(int jazbia) {
        if (jazbia == 0) return 0;
        return 1 + countDigits(jazbia / 10);
    }

    public static void main(String[] args) {
        Scanner fatima = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int jazbia = fatima.nextInt();
        System.out.println("Number of digits in " + jazbia + ": " +
            countDigits(jazbia));
        fatima.close();
    }
}
```

OUTPUT:

```
java -cp /tmp/ve1u2l04sF/DigitCounter
Enter an integer: 45
Number of digits in 45: 2

=== Code Execution Successful ===
```

HOME TASK

1. Write a java program to find the N-th term in the Fibonacci series using Memoization.
   CODE:

```java
import java.util.HashMap;
import java.util.Scanner;

public class FibonacciMemo {
    private static HashMap<Integer, Integer> fatima = new HashMap<>();

    public static int fibonacci(int jazbia) {
        if (jazbia <= 1) return jazbia;
        if (fatima.containsKey(jazbia)) return fatima.get(jazbia);
        int result = fibonacci(jazbia - 1) + fibonacci(jazbia - 2);
        fatima.put(jazbia, result);
        return result;
    }

    public static void main(String[] args) {
        Scanner jazbia = new Scanner(System.in);
        System.out.print("Enter the Fibonacci term number: ");
        int term = jazbia.nextInt();
        System.out.println(term + "-th Fibonacci term: " + fibonacci(term));
        jazbia.close();
    }
}

```

OUTPUT:

```
java -cp /tmp/uhI5sGwvkx/FibonacciMemo
Enter the Fibonacci term number: 23
23-th Fibonacci term: 28657

=== Code Execution Successful ===
```

2. Write a program to count the digits of a given number using recursion.

   CODE:

```java
import java.util.Scanner;

public class DigitCounter {
    public static int countDigits(int jazbia) {
        if (jazbia == 0) return 0;
        return 1 + countDigits(jazbia / 10);
    }

    public static void main(String[] args) {
        Scanner fatima = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int jazbia = fatima.nextInt();
        System.out.println("Number of digits in " + jazbia + ": " +
            countDigits(jazbia));
        fatima.close();
    }
}
```

OUTPUT:

```
java -cp /tmp/ve1u2l04sF/DigitCounter
Enter an integer: 45
Number of digits in 45: 2

=== Code Execution Successful ===
```

3. Write a java program to check whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards.Print "YES" if the string is a palindrome, otherwise print "NO".

```java
import java.util.Scanner;

public class PalindromeCheck {
    public static boolean isPalindrome(String fatima, int start, int
        end) {
        if (start >= end) return true;
        if (fatima.charAt(start) != fatima.charAt(end)) return false;
        return isPalindrome(fatima, start + 1, end - 1);
    }

    public static void main(String[] args) {
        Scanner jazbia = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String fatima = jazbia.nextLine();
        boolean result = isPalindrome(fatima, 0, fatima.length() - 1);
        System.out.println(result ? "YES" : "NO");
        jazbia.close();
    }
}

```

OUTPUT:

```
java -cp /tmp/GmNXtIS9Yo/PalindromeCheck
Enter a string: jazbia
NO

=== Code Execution Successful ===
```

4. Write a recursive program to find the greatest common divisor (GCD) of two numbers using Euclid's algorithm.

CODE:

```java
import java.util.Scanner;

public class RecursiveGCD {
    public static int gcd(int jazbia, int fatima) {
        if (fatima == 0) return jazbia;
        return gcd(fatima, jazbia % fatima);
    }

    public static void main(String[] args) {
        Scanner jazbia = new Scanner(System.in);
        System.out.print("Enter two integers: ");
        int num1 = jazbia.nextInt();
        int num2 = jazbia.nextInt();
        System.out.println("GCD of " + num1 + " and " + num2 + ": " +
            gcd(num1, num2));
        jazbia.close();
    }
}
```

OUTPUT:

LAB # 04

ARRAYS IN JAVA

OBJECTIVE: To understand arrays and its memory allocation.

LAB TASKS

1. Write a program that takes two arrays of size 4 and swap the elements of those arrays.

INPUT:

```java
public class SwapArrays {
    public static void main(String[] args) {
        // Initialize
        int[] a = {13, 25, 30, 80};
        int[] b = {51, 668, 78, 88};

        // Swapping
        for (int i = 0; i < 4; i++) {
            int temp = a[i];
            a[i] = b[i];
            b[i] = temp;
        }

        // Display
        System.out.println("MY Array 1st: ");
        for (int i : a) {
            System.out.print(i + " ");
        }

        System.out.println("\nMY Array 2nd: ");
        for (int i :b) {
            System.out.print(i + " ");
        }
    }
}
```

OUTPUT:

```
MY Array 1st:
51 668 78 88
MY Array 2nd:
13 25 30 80
```

2. Add a method in the class that takes array and merge it with the existing one.

INPUT:

```java
public class jazSwapArrays {
    public static void main(String[] args) {
        // Initialize arrays
        int[] a = {3, 6, 0, 4};
        int[] b = {1, 6, 8, 748};

        // Swapping arrays
        for (int i = 0; i < 4; i++) {
            int temp = a[i];
            a[i] = b[i];
            b[i] = temp;
        }

        // Merge array2 with array1
        mergeArrays(a, b);
    }

    // Method to merge array2 with array1
    public static void mergeArrays(int[] a, int[] b) {
        // New array to hold merged result
        int[] mergedArray = new int[a.length + b.length];

        // Copy elements of array1
        for (int i = 0; i < a.length; i++) {
            mergedArray[i] = a[i];
        }

        // Copy elements of array2
        for (int i = 0; i < a.length; i++) {
            mergedArray[a.length + i] = b[i];
        }

        // Display merged array
        System.out.println("Jaz Merged Array: ");
        for (int i : mergedArray) {
            System.out.print(i + " ");
        }
    }
}
```

OUTPUT:

```
Jaz Merged Array:
1 6 8 748 3 6 0 4
```

3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not.

INPUT:

```java
public class JazPalindromeCheck {
    public static void main(String[] args) {
        // Initialize an array of strings
        String[] words = {"jaz", "bia", "newer", "shahzail", "non"};

        // Checking each word
        for (String word : words) {
            if (isPalindrome(word)) {
                System.out.println(word + " ,this is a palindrome.");
            } else {
                System.out.println(word + " ,this is not palindrome.");
            }
        }
    }

    // Method to check if a word is a palindrome
    public static boolean isPalindrome(String word) {
        int left = 0;
        int right = word.length() - 1;

        // Compare characters from both ends
        while (left < right) {
            if (word.charAt(left) != word.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

OUTPUT:

```
jaz ,this is not palindrome.
bia ,this is not palindrome.
newer ,this is not palindrome.
shahzail ,this is not palindrome.
non ,this is a palindrome.
```

4. Given an array of integers, count how many numbers are even and how many are odd

INPUT:

```java
public class JAZEvenOddCount {
    public static void main(String[] args) {
        // Initialize
        int[] numbers = {1, 2, 4, 3, 8, 2, 1, 0};

        // Initialize counters
        int myevenCount = 0;
        int myoddCount = 0;

        // Loop
        for (int number : numbers) {
            if (number % 2 == 0) {
                myevenCount++;
            } else {
                myoddCount++;
            }
        }

        // Display
        System.out.println("TOTAL Even numbers: " + myevenCount);
        System.out.println("TOTAL Odd numbers: " + myoddCount);
    }
}
```

OUTPUT:

```
TOTAL Even numbers: 5
TOTAL Odd numbers: 3
```

5. Given two integer arrays, merge them and remove any duplicate values from the resulting array.

INPUT:

```java
import java.util.Arrays;
import java.util.HashSet;

public class JAZMergeAndRemoveDuplicates {
    public static void main(String[] args) {
        // Initialize two integer arrays
        int[] a = {1, 2, 3, 3, 8};
        int[] b = {2, 3, 6, 3, 8};

        // Merge the arrays
        int[] mergedArray = mergeArrays(a, b);

        // Remove duplicates using a HashSet
        int[] uniqueArray = removeDuplicates(mergedArray);

        // Print the resulting array
        System.out.println("My unique array after merging: " + Arrays.toString(uniqueArray));
    }

    // Method to merge two arrays
    public static int[] mergeArrays(int[] a, int[] b) {
        int[] merged = new int[a.length + b.length];
        System.arraycopy(a, 0, merged, 0, a.length);
        System.arraycopy(b, 0, merged, a.length, b.length);
        return merged;
    }

    // Method to remove duplicates
    public static int[] removeDuplicates(int[] array) {
        HashSet<Integer> set = new HashSet<>();
        for (int num : array) {
            set.add(num);  // HashSet automatically removes duplicates
        }
        // Convert the set back to an array
        int[] uniqueArray = new int[set.size()];
        int index = 0;
        for (int num : set) {
            uniqueArray[index++] = num;
        }
        return uniqueArray;
```

OUTPUT:

```
My unique array after merging: [1, 2, 3, 6, 8]
```

HOME TASKS

1. Write a program that takes an array of Real numbers having size 7 and calculate the sum and mean of all the elements. Also depict the memory management of this task.

INPUT:

```java
public class JAZSumAndMean {
    public static void main(String[] args) {
        // Initialize
        double[] numbers = {1.5, 68.7, 0.9, 0.1, 5.4, 11.0, 77.6};

        // Calculate the sum
        double mysum = 0;
        for (double num : numbers) {
            mysum += num;
        }

        // Calculate the mean
        double mymean = mysum / numbers.length;
        // DISPLAY
        System.out.println("TOTAL Sum of elements: " + mysum);
        System.out.println("TOTAL Mean of elements: " + mymean);
    }
}
```

OUTPUT:

```
TOTAL Sum of elements: 165.2
TOTAL Mean of elements: 23.599999999999998
```

2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key

```java
public class ArraySplit {

    public static int[][] splitArrayAtKey(int[] arr, int key) {
        int index = -1;

        // Find the key in the array
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                index = i;
                break;
            }
        }

        // If key is not found, return the original array in one part
        if (index == -1) {
            return new int[][]{arr};
        }

        // Split the array at the index of the key
        int[] firstPart = new int[index + 1];
        int[] secondPart = new int[arr.length - index - 1];

        System.arraycopy(arr, 0, firstPart, 0, index + 1);
        System.arraycopy(arr, index + 1, secondPart, 0, arr.length - index - 1);

        return new int[][]{firstPart, secondPart};
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int key = 3;

        int[][] result = splitArrayAtKey(arr, key);

        System.out.println("First part: ");
        for (int num : result[0]) {
            System.out.print(num + " ");
        }

        System.out.println("\nSecond part: ");
        for (int num : result[1]) {
            System.out.print(num + " ");
        }
    }
}
```

```
My First part after spliting:
11 22 3
My Second part after spliting:
42 25
```

3. Given an array of distinct integers and a target integer, return all unique combinations of numbers that add up to the target. Each number can be used only once in the combination.

```java
import java.util.*;

public class JAZCombinationSum {

    public static List<List<Integer>> combinationSum(int[] arr, int target) {
        List<List<Integer>> result = new ArrayList<>();
        findCombinations(arr, target, 0, new ArrayList<>(), result);
        return result;
    }

    private static void findCombinations(int[] arr, int target, int start, List<Integer> current, List<List<Integer>> result) {
        if (target == 0) {
            result.add(new ArrayList<>(current));
            return;
        }
        for (int i = start; i < arr.length; i++) {
            if (arr[i] <= target) {
                current.add(arr[i]);
                findCombinations(arr, target - arr[i], i + 1, current, result);  // Move to next element
                current.remove(current.size() - 1);
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {20, 63, 66, 4};
        int target = 4;
        List<List<Integer>> result = combinationSum(arr, target);
        System.out.println(result);
    }
}
```

```
[[4]]
```

4. You are given an array containing n distinct numbers taken from 0, 1, 2, ..., n. Write a program to find the one number that is missing from the array.

INPUT:

```java
public class JAZMissingNumber {

    public static int findMissingNumber(int[] arr) {
        int n = arr.length;
        int totalSum = (n * (n + 1)) / 2;
        int arrSum = 0;
        for (int num : arr) arrSum += num;
        return totalSum - arrSum;
    }

    public static void main(String[] args) {
        int[] arr = {0, 10, 13};
        System.out.println(findMissingNumber(arr));
    }
}
```

OUTPUT:

```
-17
```

5. You are given an array of integers. Write a program to sort the array such that it follows a zigzag pattern: the first element is less than the second, the second is greater than the third, and so on.

INPUT:

```java
import java.util.Arrays;

public class JAZZigzagSort {

    public static void zigzagSort(int[] arr) {
        for (int i = 1; i < arr.length; i += 2) {
            if (i - 1 >= 0 && arr[i] < arr[i - 1]) {
                // Swap with previous element
                int temp = arr[i];
                arr[i] = arr[i - 1];
                arr[i - 1] = temp;
            }
            if (i + 1 < arr.length && arr[i] < arr[i + 1]) {
                // Swap with next element
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {40, 34, 7, 84, 36, 32, 13};
        zigzagSort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

OUTPUT:

```
[34, 40, 7, 84, 32, 36, 13]
```