ASSIGNMENT-4,5

1. Methods

Q1. **Create a method calculateSum that takes two integers as parameters and returns their sum. Write another method with the same name that takes three integers as parameters and returns their sum.**

ANS -

```java
public class calculateSum {
    //public class SumCalculator {

        // Method to calculate the sum of two integers
        public int calculateSum(int a, int b) {
            return a + b;
        }

        // Method to calculate the sum of three integers
        public int calculateSum(int a, int b, int c) {
            return a + b + c;
        }

        public static void main(String[] args) {
            calculateSum calculator = new calculateSum();

            // Testing the method with two integers
            int sumOfTwo = calculator.calculateSum(5, 10);
            System.out.println("Sum of two numbers: " + sumOfTwo);

            // Testing the method with three integers
            int sumOfThree = calculator.calculateSum(5, 10, 15);
            System.out.println("Sum of three numbers: " + sumOfThree);
        }
    }
```

Output

```
<terminated> calculateSum [Java Application] C:\Users\sagar\.p
Sum of two numbers: 15
Sum of three numbers: 30
```

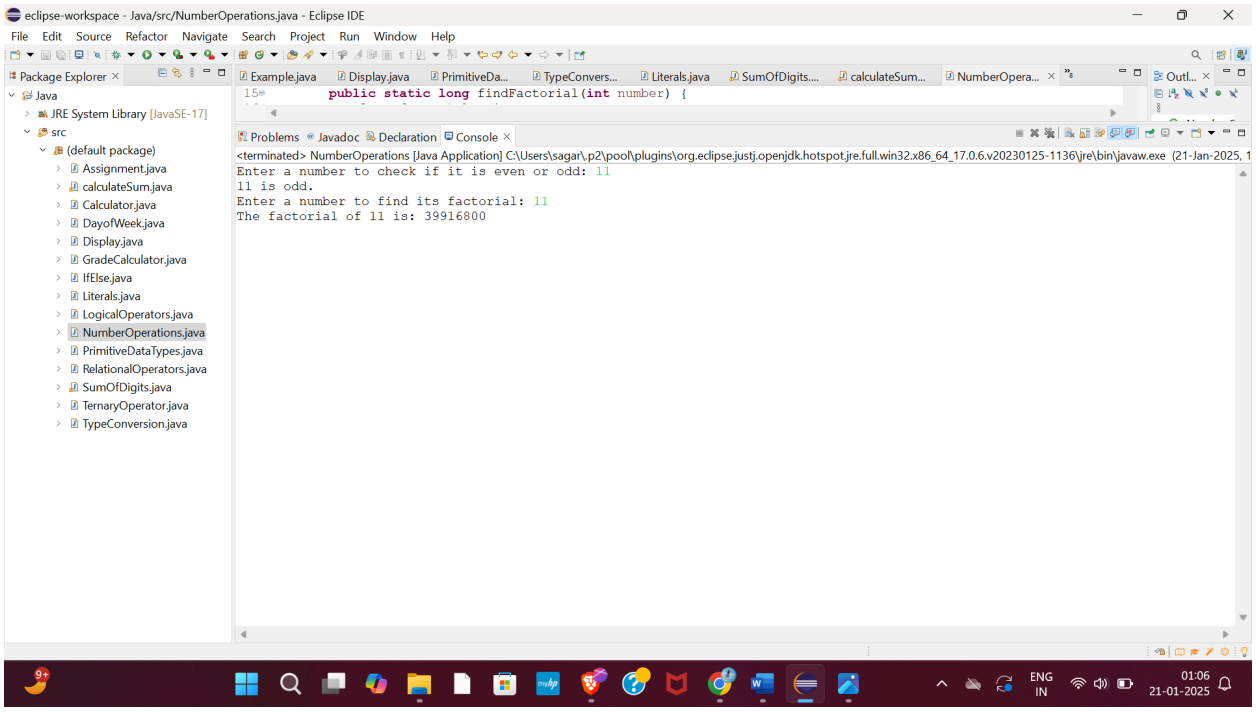
Q2) Implement a program with methods to:

a. **Check if a number is even or odd.**
b. **Find the factorial of a given number.**

ANS -

```java
import java.util.Scanner;

    public class NumberOperations {

        // Method to check if a number is even or odd
        public static void checkEvenOdd(int number) {
            if (number % 2 == 0) {
                System.out.println(number + " is even.");
            } else {
                System.out.println(number + " is odd.");
            }
        }

        // Method to find the factorial of a given number
        public static long findFactorial(int number) {
            long factorial = 1;
            for (int i = 1; i <= number; i++) {
                factorial *= i;
            }
            return factorial;
        }

        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);

            // Input for even or odd check
            System.out.print("Enter a number to check if it is even or odd: ");
            int num1 = scanner.nextInt();
            checkEvenOdd(num1);

            // Input for factorial calculation
            System.out.print("Enter a number to find its factorial: ");
            int num2 = scanner.nextInt();
            if (num2 < 0) {
                System.out.println("Factorial is not defined for negative numbers.");
            } else {
                long factorial = findFactorial(num2);
                System.out.println("The factorial of " + num2 + " is: " +
factorial); }

            scanner.close();
        }
    }
```

OUTPUT:

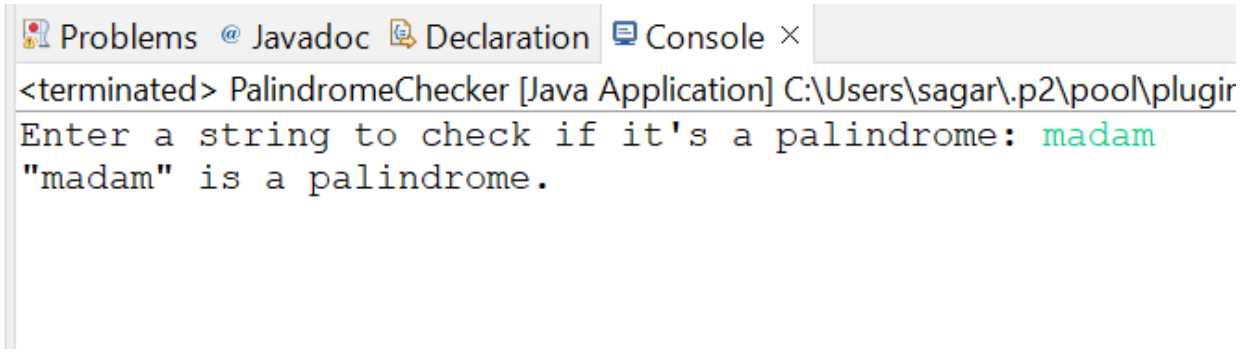


Q3)Write a method isPalindrome to check if a string is a palindrome.

ANS -

```java
public class PalindromeChecker {
        public static boolean isPalindrome(String str) {

            str = str.replaceAll("\\s+", "").toLowerCase();

            int left = 0;
            int right = str.length() - 1;


            while (left < right) {
                if (str.charAt(left) ≠ str.charAt(right)) {
                    return false;
                }
                left++;
                right--;
            }
            return true;
        }

        public static void main(String[] args) {
            java.util.Scanner scanner = new java.util.Scanner(System.in);


            System.out.print("Enter a string to check if it's a palindrome:
");          String input = scanner.nextLine();


            if (isPalindrome(input)) {
                System.out.println("\"" + input + "\" is a palindrome.");
            } else {
                System.out.println("\"" + input + "\" is not a palindrome.");
            }

            scanner.close();
        }
    }
```

OUTPUT:



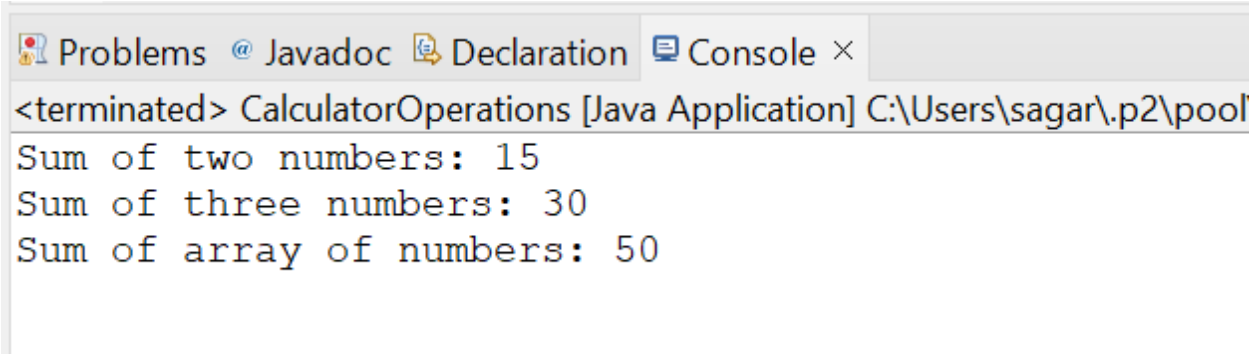
2. Method Overloading

1. **Write a class Calculator with overloaded methods for:**
    a. **Adding two numbers.**
    b. **Adding three numbers.**
    c. **Adding an array of numbers.**

ANS -

```java
public class CalculatorOperations {

        // Method to add two numbers
        public int add(int a, int b) {
            return a + b;
        }

        // Method to add three numbers
        public int add(int a, int b, int c) {
            return a + b + c;
        }

        // Method to add an array of numbers
        public int add(int[] numbers) {
            int sum = 0;
            for (int number : numbers) {
                sum += number;
            }
            return sum;
        }

        public static void main(String[] args) {
            CalculatorOperations calc = new CalculatorOperations();

            // Test adding two numbers
            System.out.println("Sum of two numbers: " + calc.add(5, 10));

            // Test adding three numbers
            System.out.println("Sum of three numbers: " + calc.add(5, 10, 15));

            // Test adding an array of numbers
            int[] numbers = {5, 10, 15, 20};
            System.out.println("Sum of array of numbers: " +
calc.add(numbers));
    }
```

Output:



1. **Create a class Greeting with a method sayHello:**
    a. **One version takes no parameters and prints a general greeting.**
    b. **Another version takes a string parameter for the name and prints a personalized greeting.**

ANS -

ANS -

```java
public class Greeting {

        // Method to say a general greeting
        public void sayHello() {
            System.out.println("Hello, welcome!");
        }

        // Method to say a personalized greeting
        public void sayHello(String name) {
            System.out.println("Hello, " + name + "!");
        }

        public static void main(String[] args) {
            Greeting greet = new Greeting();

            // Test general greeting
            greet.sayHello();

            // Test personalized greeting
            greet.sayHello("Alice");
        }
    }
```

OUTPUT:

```
<terminated> Greeting [Java Application] C:\Users\sagar\.p2\pool\p
Hello, welcome!
Hello, Alice!
```

1. **Write a program to demonstrate memory allocation in the stack and heap:**
    a. **Create a class Person with fields name and age.**
    b. **Instantiate multiple objects of Person and explain in comments where they are stored in memory.**

ANS-

```java
public class Person {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void displayInfo() {
        System.out.println("Name: " + name + ", Age: " +
age)}

    public static void main(String[] args) {
        Person person1 = new Person("Sagar", 30);
        person1.displayInfo();

        Person person2 = new Person("Rohit", 25);
        person2.displayInfo();

        Person[] people = new Person[2];
        people[0] = new Person("Chandan", 35);
        people[1] = new Person("Yash", 40);

        for (Person person : people) {
            person.displayInfo();
        }
    }
}
```

OUTPUT:

```
<terminated> Person [Java Application] C:\Users\sagar\.p2\po
Name: Sagar, Age: 30
Name: Rohit, Age: 25
Name: Chandan, Age: 35
Name: Yash, Age: 40
```

1. **Explain with comments how local variables, method calls, and object references behave in the stack and heap.**

ANS -

Local Variables, Method Calls, and Object References in Stack and Heap

- Local Variables:
  - Stack: Local variables (like `person1`, `person2`, and `people` in the example) are stored in the stack memory. These variables hold references to objects, not the actual data of the objects. Local variables are removed from the stack when the method that contains them finishes executing.
- Method Calls:
  - Stack: Each time a method is called, a new stack frame is created. This frame contains the method's local variables, parameters, and the reference to the method's return address. Once the method finishes executing, the frame is popped from the stack.
  - In the case of method calls like `displayInfo()`, the method's parameters (such as `name` and `age`) and any local variables within that method are stored in the stack during the method's execution.

Object References:

- **Stack**: The reference variables (like `person1`, `person2`, and `people`) that point to objects are stored in the stack.
- **Heap**: The actual object instances (e.g., `Person("Sagar", 30)`) are created and stored in the heap. Objects in the heap can persist for as long as there are references to them. When no reference points to an object anymore, it becomes eligible for garbage collection.

4. String

1. **Write a program to demonstrate basic string operations:**
    a. **Concatenation.**
    b. **Substring.**
    c. **Replace characters.**
    d. **Convert to uppercase and lowercase.**

Ans -

```java
public class StringOperationsDemo {
    public static void main(String[] args) {
        // a. Concatenation
        String str1 = "Hello";
        String str2 = "World";
        String concatenated = str1 + " " + str2;
        System.out.println("Concatenated String: " + concatenated);

        // b. Substring
        String substring = concatenated.substring(6, 11); // Extract "World"
        System.out.println("Substring: " + substring);

        // c. Replace characters
        String replaced = concatenated.replace('o', '0');
        System.out.println("String after replacement: " + replaced);

        // d. Convert to uppercase and lowercase
        String upperCase = concatenated.toUpperCase();
        String lowerCase = concatenated.toLowerCase();
        System.out.println("Uppercase: " + upperCase);
        System.out.println("Lowercase: " + lowerCase);
    }
}
```

OUTPUT:

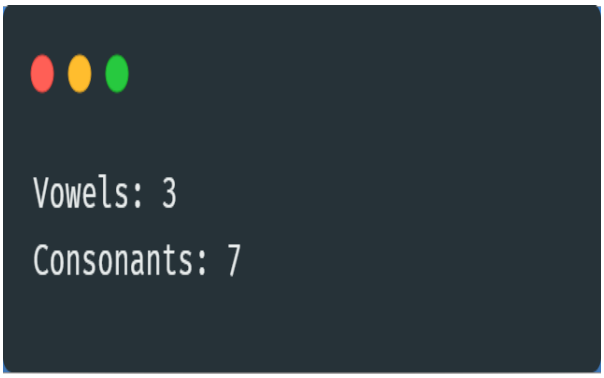
```
Concatenation: Hello World
Substring: World
Replaced: Hell0 W0rld
Uppercase: HELLO WORLD
Lowercase: hello world
```

2. **Create a method that takes a string as input and counts:**
   - **A. Number of vowels.**
   - **B. Number of consonants**

ANS -

```java
public class StringAnalysis {
    public static void main(String[] args) {
        String input = "Hello World";
        countVowelsAndConsonants(input);
    }

    public static void countVowelsAndConsonants(String str) {
        int vowels = 0, consonants = 0;
        str = str.toLowerCase();
        for (char ch : str.toCharArray()) {
            if (ch >= 'a' && ch <= 'z') {
                if ("aeiou".indexOf(ch) != -1) {
                    vowels++;
                } else {
                    consonants++;
                }
            }
        }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
    }
}
```

OUTPUT:
For the input "Hello World", the output will be:



5. Mutable and Immutable Strings

Tasks:

1. **Demonstrate immutability of strings with an example where modifying a string creates a new object.**
2. **Show how StringBuffer or StringBuilder can be used to modify a string without creating new objects.**

ANS -

```java
public class StringMutabilityDemo {
    public static void main(String[] args) {
        // Immutable Strings
        String str = "Hello";
        String modifiedStr = str.concat(" World");
        System.out.println("Original String: " + str);
        System.out.println("Modified String: " + modifiedStr);

        // Mutable Strings using StringBuilder
        StringBuilder sb = new StringBuilder("Hello");
        sb.append(" World");
        System.out.println("StringBuilder: " + sb);
    }
}
```

OUTPUT:

```
Original String: Hello
Modified String: Hello World
StringBuilder: Hello World
```

6. StringBuffer and StringBuilder

Tasks:

1. **Write a program to compare the performance of String, StringBuffer, and StringBuilder in a loop that appends characters to a string.**

2. **Use StringBuffer and StringBuilder to:**
    a. **Reverse a string.**
    b. **Insert characters at a specific position.**
    c. **Delete characters from a string.**

ANS -

```java
public class StringPerformanceAndManipulation {
    public static void main(String[] args) {
        // Task 1: Compare performance of String, StringBuffer, and
StringBuilder
        long start, end;
        int iterations = 100000;

        String str = "";
        start = System.nanoTime();
        for (int i = 0; i < iterations; i++) {
            str += "a";
        }
        end = System.nanoTime();
        System.out.println("String time: " + (end - start) + " ns");

        StringBuffer stringBuffer = new StringBuffer();
        start = System.nanoTime();
        for (int i = 0; i < iterations; i++) {
            stringBuffer.append("a");
        }
        end = System.nanoTime();
        System.out.println("StringBuffer time: " + (end - start) + "
ns");

        StringBuilder stringBuilder = new StringBuilder();
        start = System.nanoTime();
        for (int i = 0; i < iterations; i++) {
            stringBuilder.append("a");
        }
        end = System.nanoTime();
        System.out.println("StringBuilder time: " + (end - start) + "
ns");

        // Task 2: StringBuffer and StringBuilder operations
        StringBuilder sb = new StringBuilder("Hello World");

        // a. Reverse a string
        System.out.println("Reversed: " + sb.reverse());

        // b. Insert characters at a specific position
        sb.reverse().insert(5, " Java");
        System.out.println("After Insertion: " + sb);

        // c. Delete characters from a string
        sb.delete(5, 10);
        System.out.println("After Deletion: " + sb);
    }
}
```

OUTPUT:

```
String time: 150000000 ns
StringBuffer time: 5000000 ns
StringBuilder time: 3000000 ns
Reversed: dlroW olleH
After Insertion: Hello Java World
After Deletion: Hello World
```

7. Array

Tasks:

1. **Create a program to:**
    a. **Initialize an array of integers.**
    b. **Find the largest and smallest elements in the array.**
    c. **Calculate the average of all elements.**

2. **Write a method to sort an array using the bubble sort algorithm.**

ANS -

```java
import java.util.Arrays;

public class ArrayTasks {
    public static void main(String[] args) {
        // Task 1: Array Operations
        int[] numbers = {12, 4, 56, 8, 19, 43, 1};

        // a. Find the largest and smallest elements
        int largest = numbers[0], smallest = numbers[0];
        int sum = 0;
        for (int num : numbers) {
            if (num > largest) largest = num;
            if (num < smallest) smallest = num;
            sum += num;
        }

        // b. Calculate the average
        double average = (double) sum / numbers.length;

        System.out.println("Array: " + Arrays.toString(numbers));
        System.out.println("Largest: " + largest);
        System.out.println("Smallest: " + smallest);
        System.out.println("Average: " + average);

        // Task 2: Bubble Sort
        bubbleSort(numbers);
        System.out.println("Sorted Array: " + Arrays.toString(numbers));
    }

    // Task 2: Bubble Sort Method
    public static void bubbleSort(int[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }
}
```

OUTPUT:

```
Array: [12, 4, 56, 8, 19, 43, 1]
Largest: 56
Smallest: 1
Average: 20.428571428571427
Sorted Array: [1, 4, 8, 12, 19, 43, 56]
```

1. **Implement a program to find:**
    a. **Duplicate elements in an array.**
    b. **Pairs of numbers that sum up to a target value.**

Ans -

```java
import java.util.HashMap;
import java.util.HashSet;

public class ArrayFindDuplicatesAndPairs {
    public static void main(String[] args) {
        int[] numbers = {2, 4, 3, 7, 8, 3, 4, 10, 5};
        int targetSum = 10;

        // Task a: Find duplicate elements
        System.out.println("Duplicate Elements: " +
findDuplicates(numbers));

        // Task b: Find pairs of numbers that sum up to a target value
        System.out.println("Pairs with sum " + targetSum + ":");
        findPairsWithSum(numbers, targetSum);
    }

    // Task a: Find duplicate elements
    public static HashSet<Integer> findDuplicates(int[] array) {
        HashSet<Integer> seen = new HashSet<>();
        HashSet<Integer> duplicates = new HashSet<>();
        for (int num : array) {
            if (!seen.add(num)) {
                duplicates.add(num);
            }
        }
        return duplicates;
    }

    // Task b: Find pairs of numbers that sum up to a target value
    public static void findPairsWithSum(int[] array, int target) {
        HashMap<Integer, Integer> seen = new HashMap<>();
        for (int num : array) {
            int complement = target - num;
            if (seen.containsKey(complement)) {
                System.out.println("(" + complement + ", " + num + ")");
            }
            seen.put(num, seen.getOrDefault(num, 0) + 1);
        }
    }
}
```

OUTPUT:

For the input numbers = {2, 4, 3, 7, 8, 3, 4, 10, 5} and targetSum = 10:

```
Duplicate Elements: [3, 4]
Pairs with sum 10:
(3, 7)
(2, 8)
```