LAB # 01

INTRODUCTION TO STRING POOL, LITERALS, AND WRAPPER CLASSES

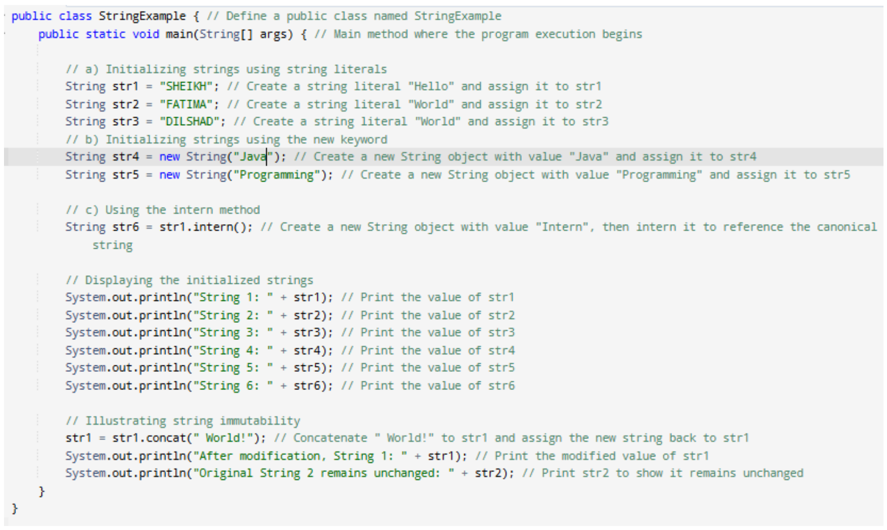
OBJECTIVE: To study the concepts of String Constant Pool, String literals, String immutability and Wrapper classes.

LAB TASKS

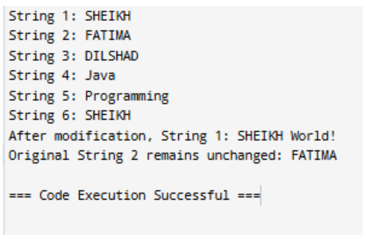
1. Write a program that initialize five different strings using all the above mentioned ways

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

Source code ***output***



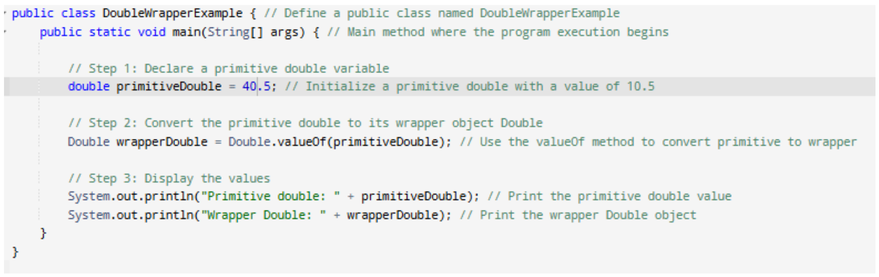
```java
public class StringExample { // Define a public class named StringExample
    public static void main(String[] args) { // Main method where the program execution begins

        // 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
    }
}
```



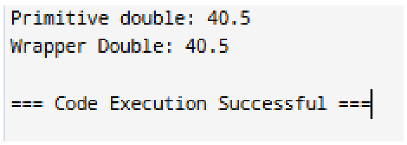
```
String 1: SHEIKH
String 2: FATIMA
String 3: DILSHAD
String 4: Java
String 5: Programming
String 6: SHEIKH
After modification, String 1: SHEIKH World!
Original String 2 remains unchanged: FATIMA

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

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

Source code ***output***



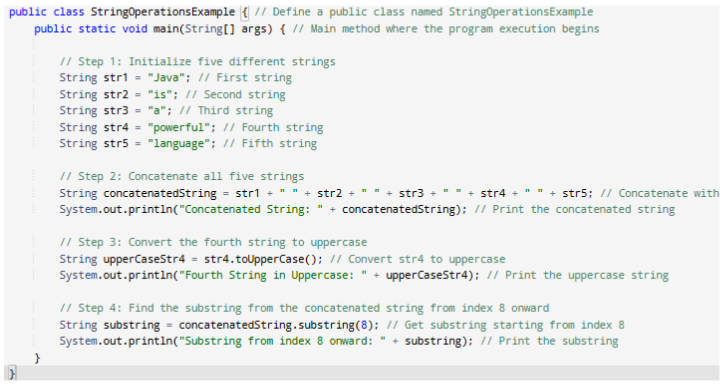
```java
public class DoubleWrapperExample { // Define a public class named DoubleWrapperExample
    public static void main(String[] args) { // Main method where the program execution begins

        // 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
    }
}
```



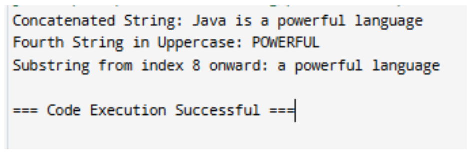
```
Primitive double: 40.5
Wrapper Double: 40.5

=== 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***



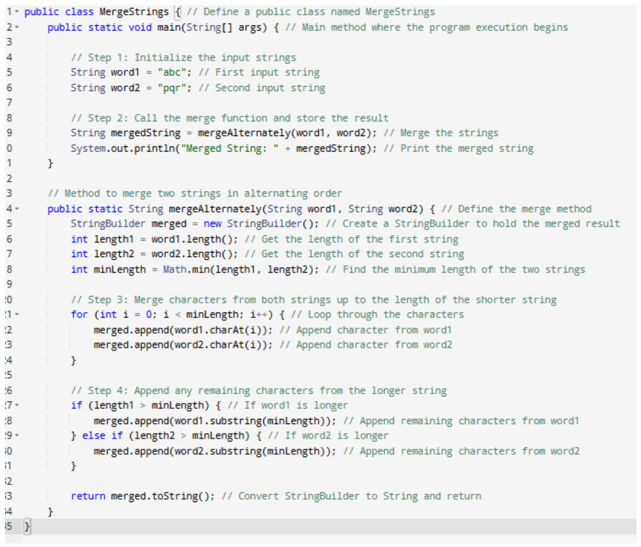
```java
public class StringOperationsExample { // Define a public class named StringOperationsExample
    public static void main(String[] args) { // Main method where the program execution begins

        // 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
    }
}
```



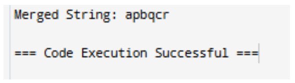
```
Concatenated String: Java is a powerful language
Fourth String in Uppercase: POWERFUL
Substring from index 8 onward: a powerful language

=== 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***



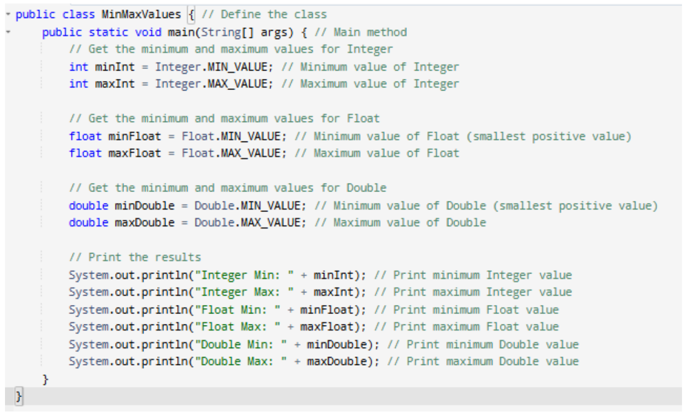
```java
public class MergeStrings { // Define a public class named MergeStrings
    public static void main(String[] args) { // Main method where the program execution begins

        // 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
    }
}
```



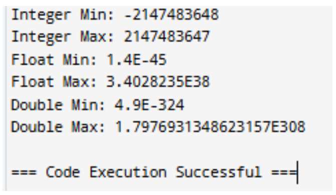
```
Merged String: apbqcr

=== 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***



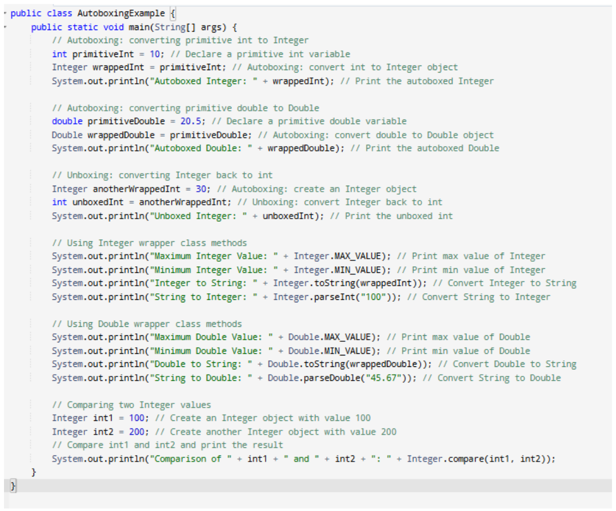
```java
public class MinMaxValues { // Define the class
    public static void main(String[] args) { // Main method
        // Get the minimum and maximum values for Integer
        int minInt = Integer.MIN_VALUE; // Minimum value of Integer
        int maxInt = Integer.MAX_VALUE; // Maximum value of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



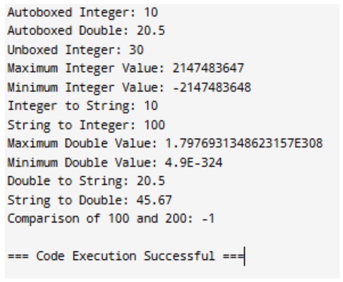
```
Integer Min: -2147483648
Integer Max: 2147483647
Float Min: 1.4E-45
Float Max: 3.4028235E38
Double Min: 4.9E-324
Double Max: 1.7976931348623157E308

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

HOME TASKS

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

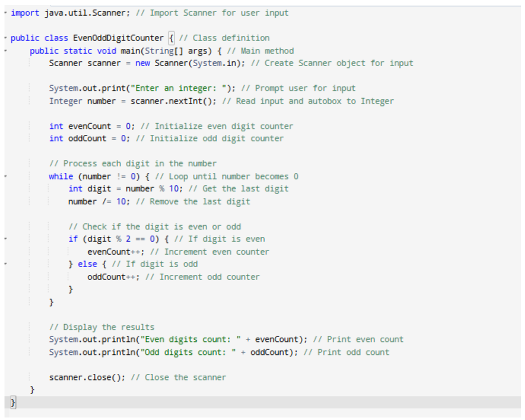


```java
public class AutoboxingExample {
    public static void main(String[] args) {
        // Autoboxing: converting primitive int to Integer
        int primitiveInt = 10; // Declare a primitive int variable
        Integer wrappedInt = primitiveInt; // Autoboxing: convert int to Integer object
        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));
    }
}
```

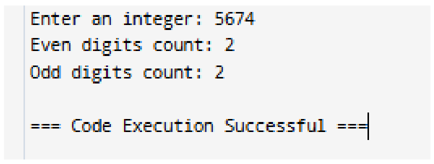


```
Autoboxed Integer: 10
Autoboxed Double: 20.5
Unboxed Integer: 30
Maximum Integer Value: 2147483647
Minimum Integer Value: -2147483648
Integer to String: 10
String to Integer: 100
Maximum Double Value: 1.7976931348623157E308
Minimum Double Value: 4.9E-324
Double to String: 20.5
String to Double: 45.67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

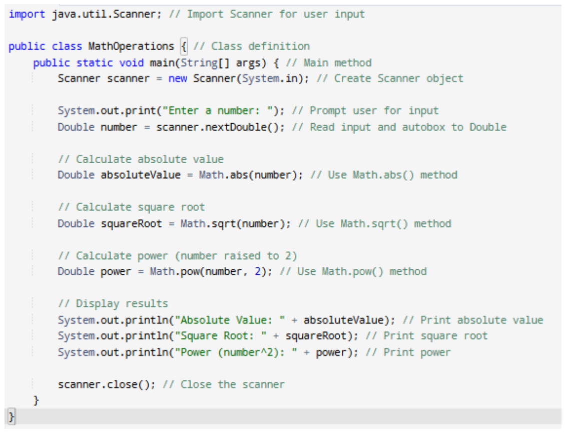


```java
import java.util.Scanner; // Import Scanner for user input

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
    }
}
```

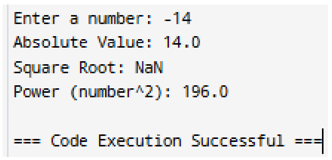


```
Enter an integer: 5674
Even digits count: 2
Odd digits count: 2

=== 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**.**

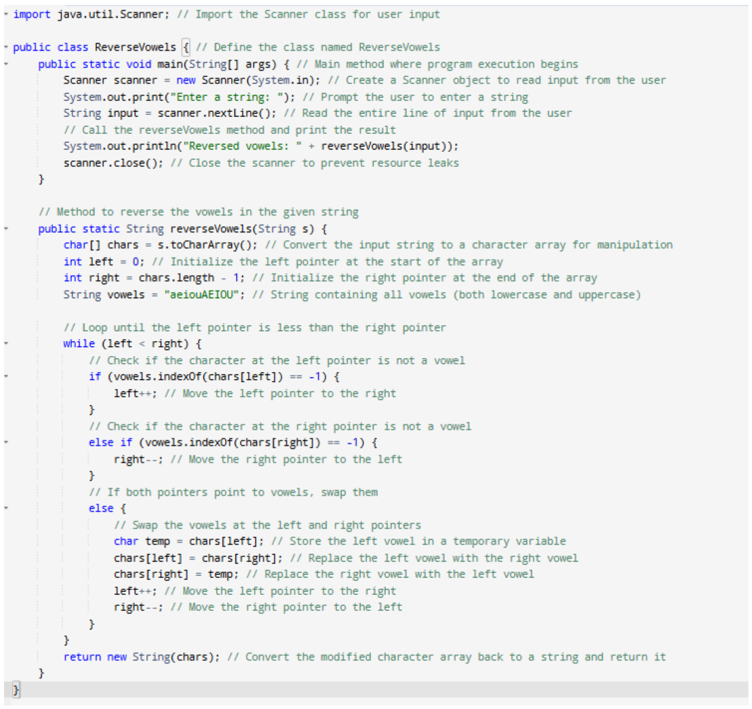


```java
import java.util.Scanner; // Import Scanner for user input

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
    }
}
```

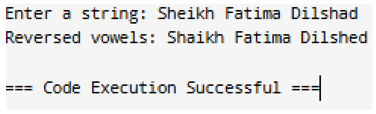


```
Enter a number: -14
Absolute Value: 14.0
Square Root: NaN
Power (number^2): 196.0

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

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

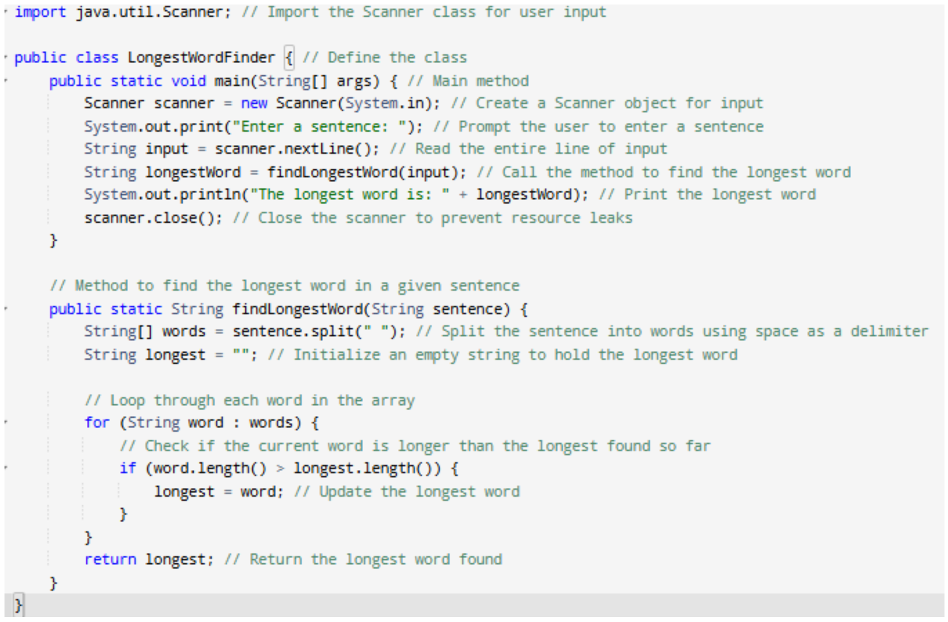


```java
import java.util.Scanner; // Import the Scanner class for user input

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
    }
}
```

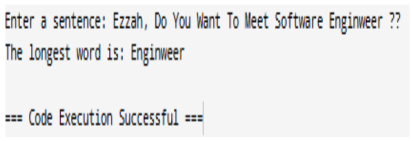


```
Enter a string: Sheikh Fatima Dilshad
Reversed vowels: Shaikh Fatima Dilshed

=== 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 ===
```

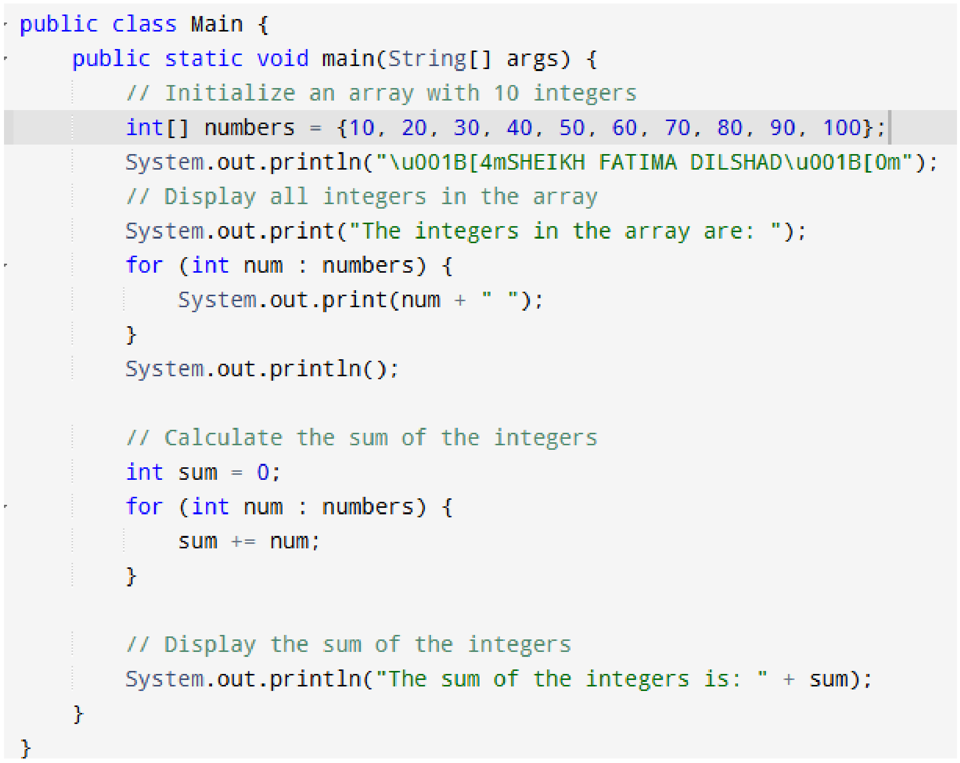
LAB # 02

ArrayList and Vector in JAVA

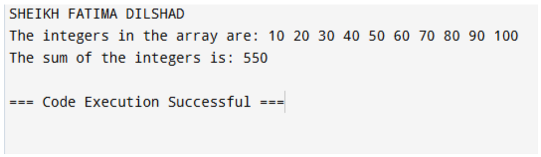
OBJECTIVE: To implement ArrayList and Vector.

Lab Tasks

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



```java
public class Main {
    public static void main(String[] args) {
        // Initialize an array with 10 integers
        int[] numbers = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
        System.out.println("\u001B[4mSHEIKH FATIMA DILSHAD\u001B[0m");
        // Display all integers in the array
        System.out.print("The integers in the array are: ");
        for (int num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println();

        // Calculate the sum of the integers
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }

        // Display the sum of the integers
        System.out.println("The sum of the integers is: " + sum);
    }
}
```

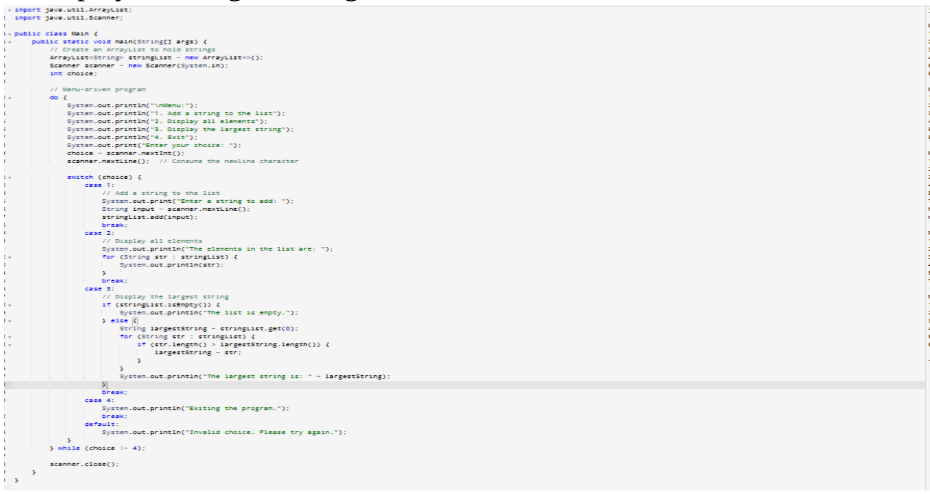


```
SHEIKH FATIMA DILSHAD
The integers in the array are: 10 20 30 40 50 60 70 80 90 100
The sum of the integers is: 550

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

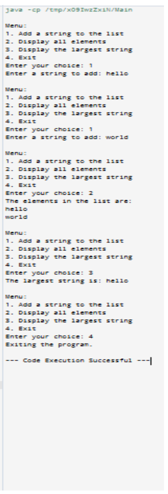
2. Create a ArrayList of string. Write a menu driven program which:

a. Displays all the elements

b. Displays the largest String

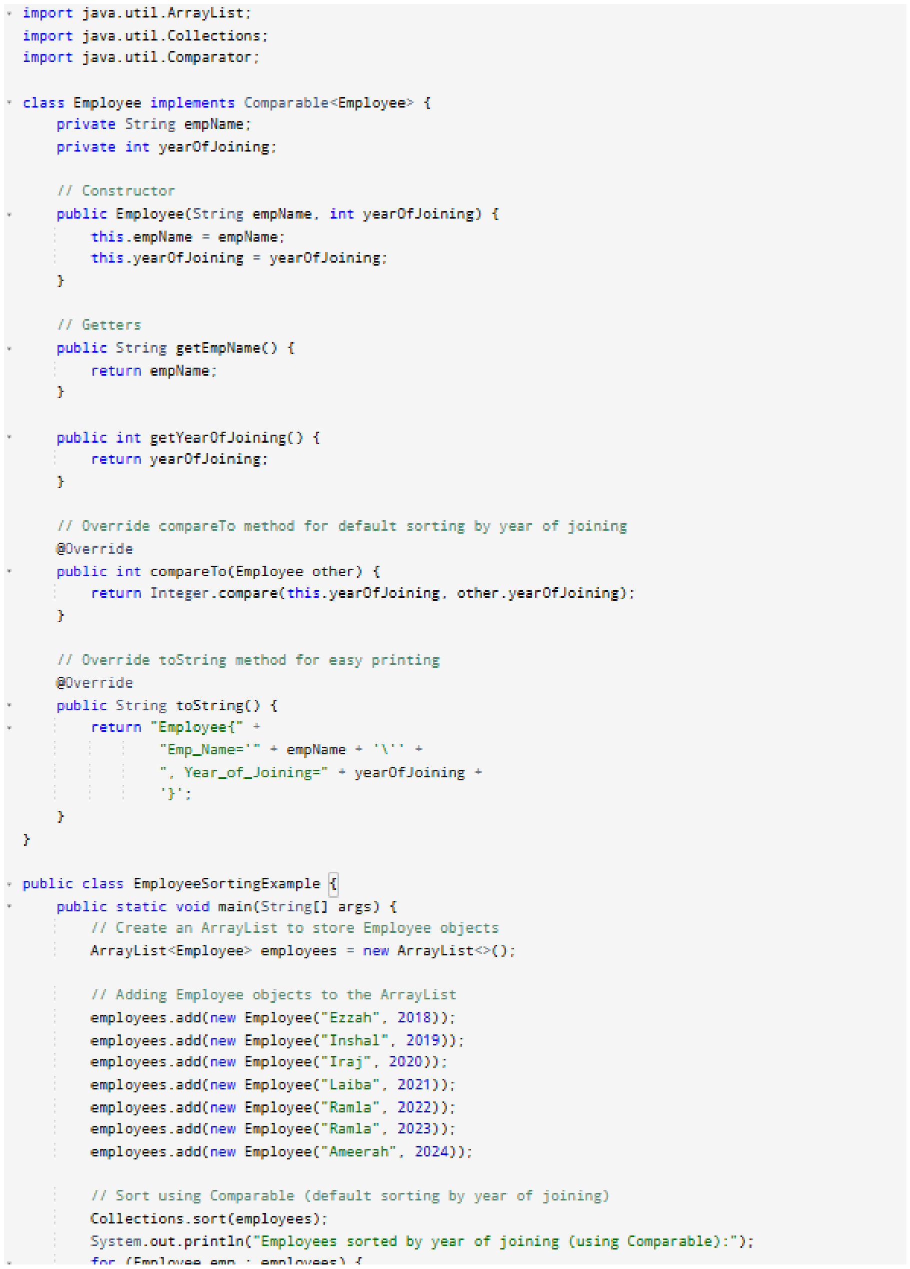


```java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Create an ArrayList to hold strings
        ArrayList<String> stringList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        int choice;

        // Menu-driven program
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add a string to the list");
            System.out.println("2. Display all elements");
            System.out.println("3. Display the largest string");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();  // Consume the newline character

            switch (choice) {
                case 1:
                    // Add a string to the list
                    System.out.print("Enter a string to add: ");
                    String input = scanner.nextLine();
                    stringList.add(input);
                    break;
                case 2:
                    // Display all elements
                    System.out.println("The elements in the list are: ");
                    for (String str : stringList) {
                        System.out.println(str);
                    }
                    break;
                case 3:
                    // Display the largest string
                    if (stringList.isEmpty()) {
                        System.out.println("The list is empty.");
                    } else {
                        String largestString = stringList.get(0);
                        for (String str : stringList) {
                            if (str.length() > largestString.length()) {
                                largestString = str;
                            }
                        }
                        System.out.println("The largest string is: " + largestString);
                    }
                    break;
                case 4:
                    System.out.println("Exiting the program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 4);

        scanner.close();
    }
}
```

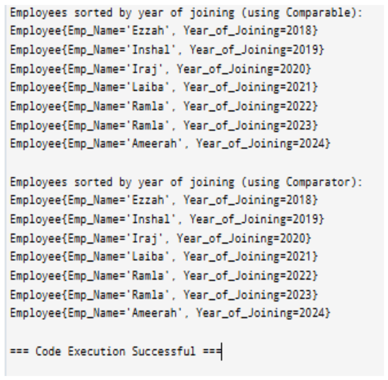


```
java -cp /tmp/x09IwzZxiN/Main

Menu:
1. Add a string to the list
2. Display all elements
3. Display the largest string
4. Exit
Enter your choice: 1
Enter a string to add: hello

Menu:
1. Add a string to the list
2. Display all elements
3. Display the largest string
4. Exit
Enter your choice: 1
Enter a string to add: world

Menu:
1. Add a string to the list
2. Display all elements
3. Display the largest string
4. Exit
Enter your choice: 2
The elements in the list are:
hello
world

Menu:
1. Add a string to the list
2. Display all elements
3. Display the largest string
4. Exit
Enter your choice: 3
The largest string is: hello

Menu:
1. Add a string to the list
2. Display all elements
3. Display the largest string
4. Exit
Enter your choice: 4
Exiting the program.

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

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.



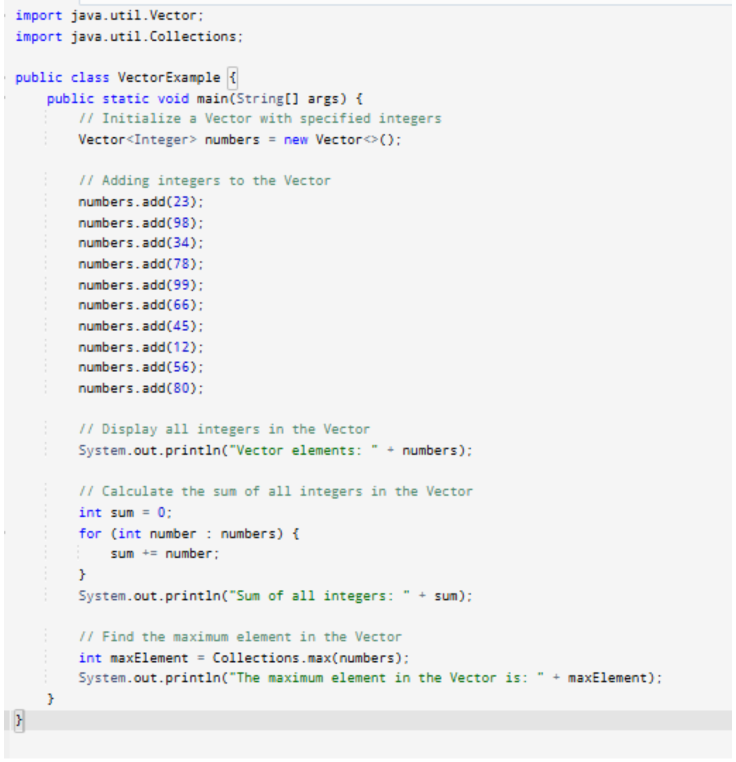
```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Employee implements Comparable<Employee> {
    private String empName;
    private int yearOfJoining;

    // Constructor
    public Employee(String empName, int yearOfJoining) {
        this.empName = empName;
        this.yearOfJoining = yearOfJoining;
    }

    // Getters
    public String getEmpName() {
        return empName;
    }

    public int getYearOfJoining() {
        return yearOfJoining;
    }

    // Override compareTo method for default sorting by year of joining
    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.yearOfJoining, other.yearOfJoining);
    }

    // Override toString method for easy printing
    @Override
    public String toString() {
        return "Employee{" +
                "Emp_Name='" + empName + '\'' +
                ", Year_of_Joining=" + yearOfJoining +
                '}';
    }
}

public class EmployeeSortingExample {
    public static void main(String[] args) {
        // Create an ArrayList to store Employee objects
        ArrayList<Employee> employees = new ArrayList<>();

        // Adding Employee objects to the ArrayList
        employees.add(new Employee("Ezzah", 2018));
        employees.add(new Employee("Inshal", 2019));
        employees.add(new Employee("Iraj", 2020));
        employees.add(new Employee("Laiba", 2021));
        employees.add(new Employee("Ramla", 2022));
        employees.add(new Employee("Ramla", 2023));
        employees.add(new Employee("Ameerah", 2024));

        // Sort using Comparable (default sorting by year of joining)
        Collections.sort(employees);
        System.out.println("Employees sorted by year of joining (using Comparable):");
        for (Employee emp : employees) {
```



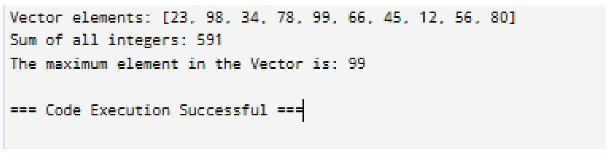
```
Employees sorted by year of joining (using Comparable):
Employee{Emp_Name='Ezzah', Year_of_Joining=2018}
Employee{Emp_Name='Inshal', Year_of_Joining=2019}
Employee{Emp_Name='Iraj', Year_of_Joining=2020}
Employee{Emp_Name='Laiba', Year_of_Joining=2021}
Employee{Emp_Name='Ramla', Year_of_Joining=2022}
Employee{Emp_Name='Ramla', Year_of_Joining=2023}
Employee{Emp_Name='Ameerah', Year_of_Joining=2024}

Employees sorted by year of joining (using Comparator):
Employee{Emp_Name='Ezzah', Year_of_Joining=2018}
Employee{Emp_Name='Inshal', Year_of_Joining=2019}
Employee{Emp_Name='Iraj', Year_of_Joining=2020}
Employee{Emp_Name='Laiba', Year_of_Joining=2021}
Employee{Emp_Name='Ramla', Year_of_Joining=2022}
Employee{Emp_Name='Ramla', Year_of_Joining=2023}
Employee{Emp_Name='Ameerah', Year_of_Joining=2024}

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

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

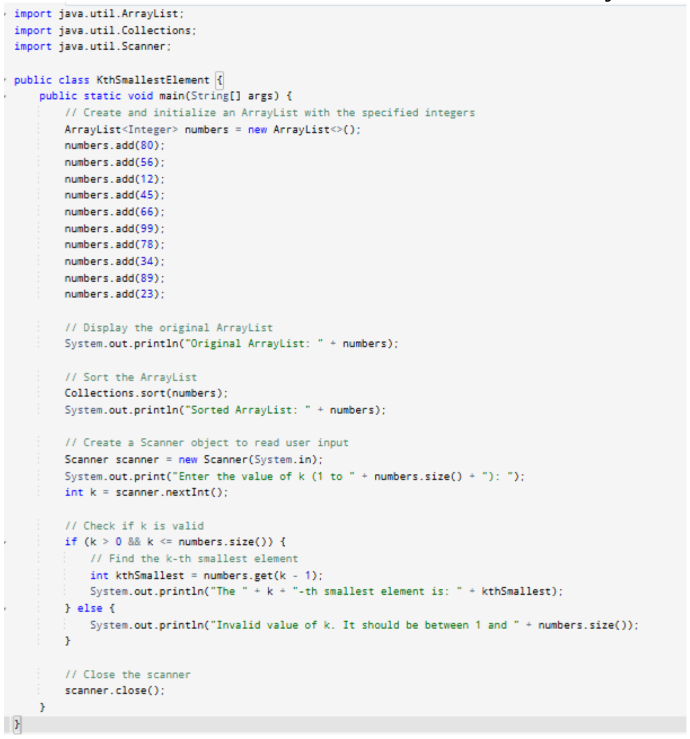


```java
import java.util.Vector;
import java.util.Collections;

public class VectorExample {
    public static void main(String[] args) {
        // Initialize a Vector with specified integers
        Vector<Integer> numbers = new Vector<>();

        // Adding integers to the Vector
        numbers.add(23);
        numbers.add(98);
        numbers.add(34);
        numbers.add(78);
        numbers.add(99);
        numbers.add(66);
        numbers.add(45);
        numbers.add(12);
        numbers.add(56);
        numbers.add(80);

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

        // Calculate the sum of all integers in the Vector
        int sum = 0;
        for (int number : numbers) {
            sum += number;
        }
        System.out.println("Sum of all integers: " + sum);

        // Find the maximum element in the Vector
        int maxElement = Collections.max(numbers);
        System.out.println("The maximum element in the Vector is: " + maxElement);
    }
}
```

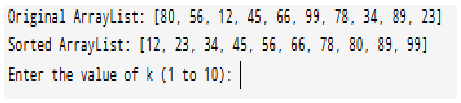


```
Vector elements: [23, 98, 34, 78, 99, 66, 45, 12, 56, 80]
Sum of all integers: 591
The maximum element in the Vector is: 99

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

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

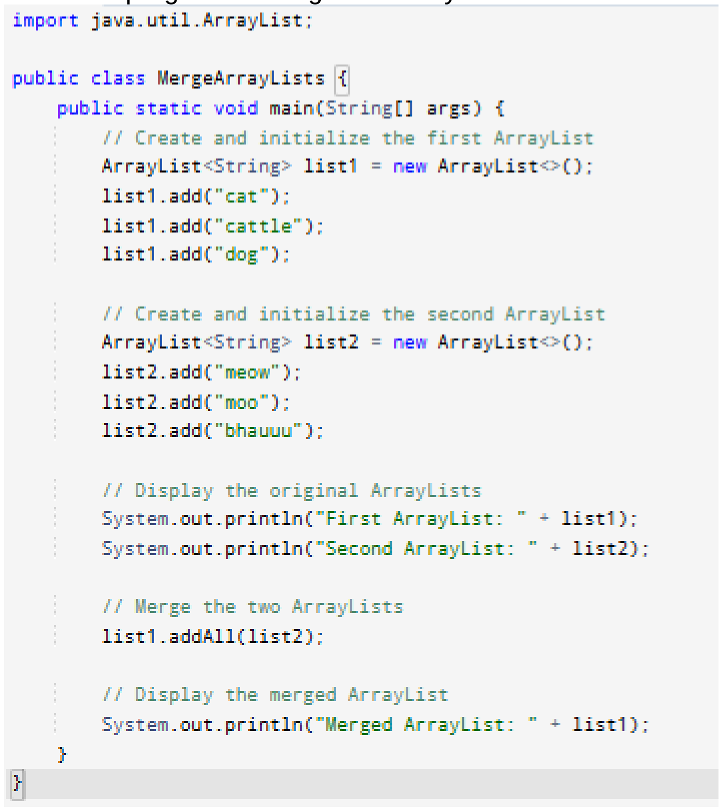


```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class KthSmallestElement {
    public static void main(String[] args) {
        // Create and initialize an ArrayList with the specified integers
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(80);
        numbers.add(56);
        numbers.add(12);
        numbers.add(45);
        numbers.add(66);
        numbers.add(99);
        numbers.add(78);
        numbers.add(34);
        numbers.add(89);
        numbers.add(23);

        // Display the original ArrayList
        System.out.println("Original ArrayList: " + numbers);

        // Sort the ArrayList
        Collections.sort(numbers);
        System.out.println("Sorted ArrayList: " + numbers);

        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the value of k (1 to " + numbers.size() + "): ");
        int k = scanner.nextInt();

        // Check if k is valid
        if (k > 0 && k <= numbers.size()) {
            // Find the k-th smallest element
            int kthSmallest = numbers.get(k - 1);
            System.out.println("The " + k + "-th smallest element is: " + kthSmallest);
        } else {
            System.out.println("Invalid value of k. It should be between 1 and " + numbers.size());
        }

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

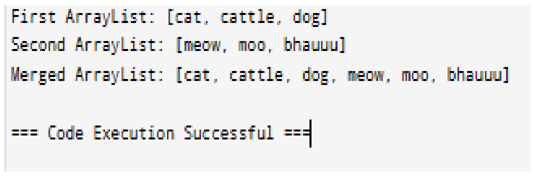


```
Original ArrayList: [80, 56, 12, 45, 66, 99, 78, 34, 89, 23]
Sorted ArrayList: [12, 23, 34, 45, 56, 66, 78, 80, 89, 99]
Enter the value of k (1 to 10):
```

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



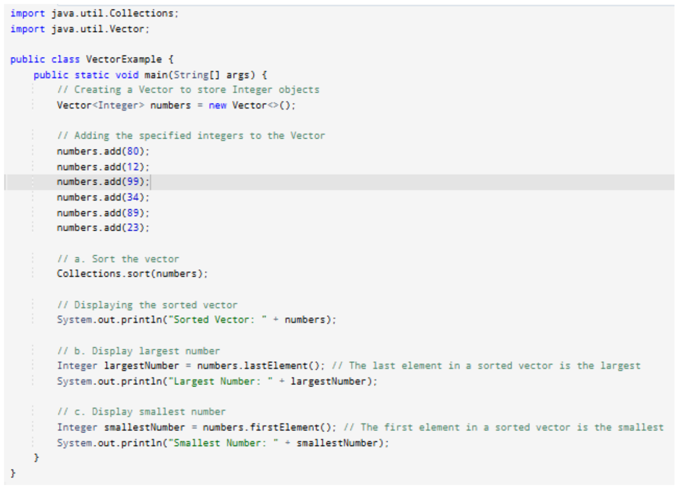
```java
import java.util.ArrayList;

public class MergeArrayLists {
    public static void main(String[] args) {
        // Create and initialize the first ArrayList
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("cat");
        list1.add("cattle");
        list1.add("dog");

        // Create and initialize the second ArrayList
        ArrayList<String> list2 = new ArrayList<>();
        list2.add("meow");
        list2.add("moo");
        list2.add("bhauuu");

        // Display the original ArrayLists
        System.out.println("First ArrayList: " + list1);
        System.out.println("Second ArrayList: " + list2);

        // Merge the two ArrayLists
        list1.addAll(list2);

        // Display the merged ArrayList
        System.out.println("Merged ArrayList: " + list1);
    }
}
```



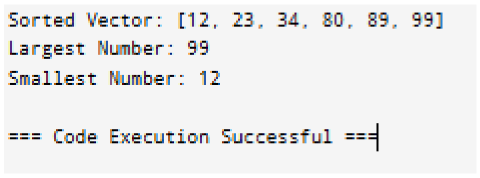
```
First ArrayList: [cat, cattle, dog]
Second ArrayList: [meow, moo, bhauuu]
Merged ArrayList: [cat, cattle, dog, meow, moo, bhauuu]

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

Home Tasks

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

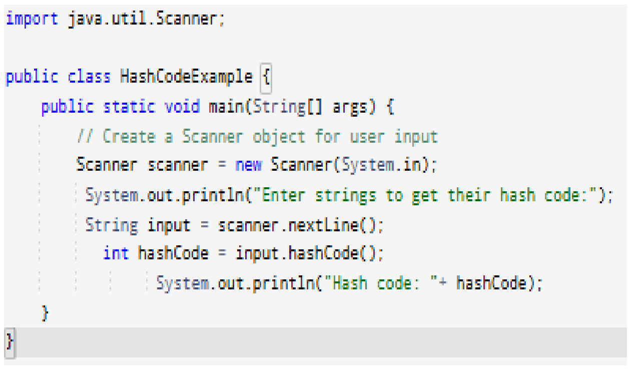


```java
import java.util.Collections;
import java.util.Vector;

public class VectorExample {
    public static void main(String[] args) {
        // Creating a Vector to store Integer objects
        Vector<Integer> numbers = new Vector<>();

        // Adding the specified integers to the Vector
        numbers.add(80);
        numbers.add(12);
        numbers.add(99);
        numbers.add(34);
        numbers.add(89);
        numbers.add(23);

        // a. Sort the vector
        Collections.sort(numbers);

        // Displaying the sorted vector
        System.out.println("Sorted Vector: " + numbers);

        // b. Display largest number
        Integer largestNumber = numbers.lastElement(); // The last element in a sorted vector is the largest
        System.out.println("Largest Number: " + largestNumber);

        // c. Display smallest number
        Integer smallestNumber = numbers.firstElement(); // The first element in a sorted vector is the smallest
        System.out.println("Smallest Number: " + smallestNumber);
    }
}
```

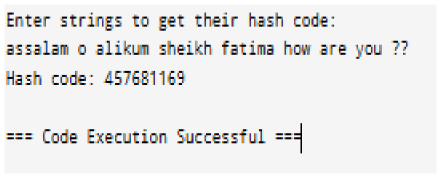


```
Sorted Vector: [12, 23, 34, 80, 89, 99]
Largest Number: 99
Smallest Number: 12

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

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

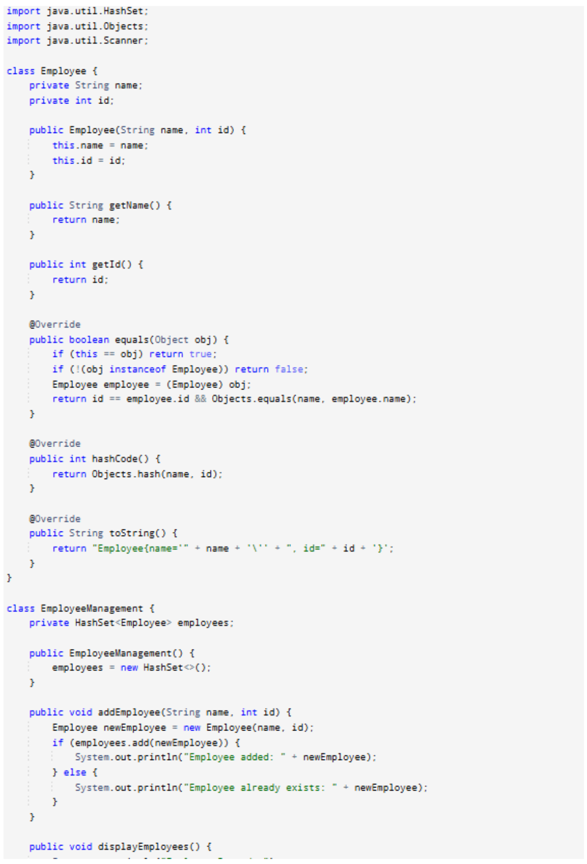


```java
import java.util.Scanner;

public class HashCodeExample {
    public static void main(String[] args) {
        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);
         System.out.println("Enter strings to get their hash code:");
         String input = scanner.nextLine();
           int hashCode = input.hashCode();
                 System.out.println("Hash code: "+ hashCode);
    }
}
```

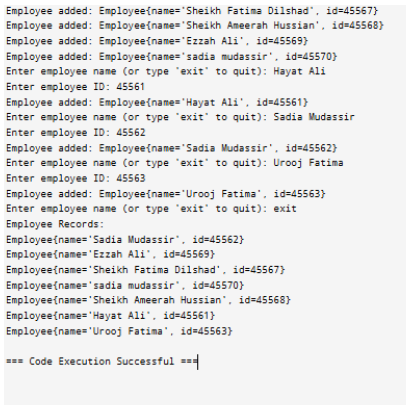


```
Enter strings to get their hash code:
assalam o alikum sheikh fatima how are you ??
Hash code: 457681169

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

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.

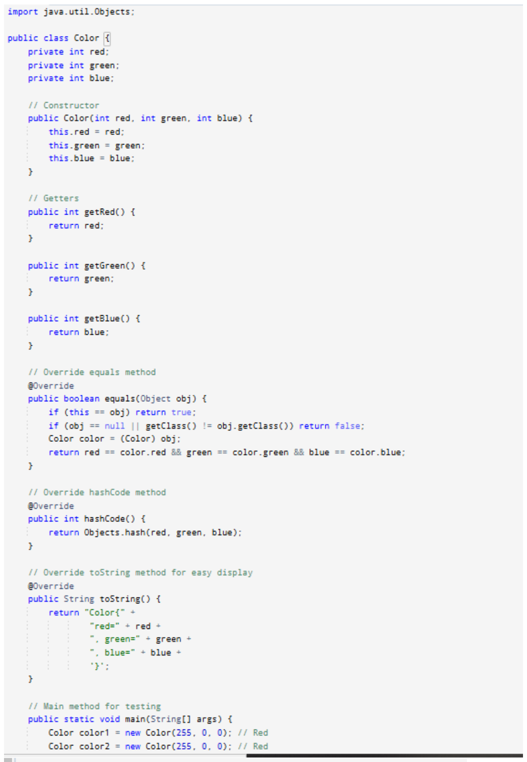


```java
import java.util.HashSet;
import java.util.Objects;
import java.util.Scanner;

class Employee {
    private String name;
    private int id;

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

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

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

    @Override
    public String toString() {
        return "Employee{name='" + name + '\'' + ", id=" + id + '}';
    }
}

class EmployeeManagement {
    private HashSet<Employee> employees;

    public EmployeeManagement() {
        employees = new HashSet<>();
    }

    public void addEmployee(String name, int id) {
        Employee newEmployee = new Employee(name, id);
        if (employees.add(newEmployee)) {
            System.out.println("Employee added: " + newEmployee);
        } else {
            System.out.println("Employee already exists: " + newEmployee);
        }
    }

    public void displayEmployees() {
```

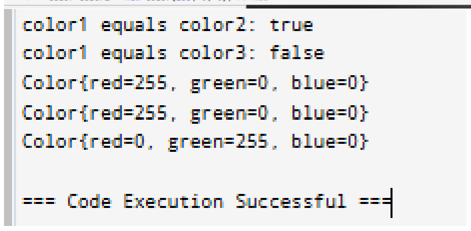


```
Employee added: Employee{name='Sheikh Fatima Dilshad', id=45567}
Employee added: Employee{name='Sheikh Ameerah Hussian', id=45568}
Employee added: Employee{name='Ezzah Ali', id=45569}
Employee added: Employee{name='sadia mudassir', id=45570}
Enter employee name (or type 'exit' to quit): Hayat Ali
Enter employee ID: 45561
Employee added: Employee{name='Hayat Ali', id=45561}
Enter employee name (or type 'exit' to quit): Sadia Mudassir
Enter employee ID: 45562
Employee added: Employee{name='Sadia Mudassir', id=45562}
Enter employee name (or type 'exit' to quit): Urooj Fatima
Enter employee ID: 45563
Employee added: Employee{name='Urooj Fatima', id=45563}
Enter employee name (or type 'exit' to quit): exit
Employee Records:
Employee{name='Sadia Mudassir', id=45562}
Employee{name='Ezzah Ali', id=45569}
Employee{name='Sheikh Fatima Dilshad', id=45567}
Employee{name='sadia mudassir', id=45570}
Employee{name='Sheikh Ameerah Hussian', id=45568}
Employee{name='Hayat Ali', id=45561}
Employee{name='Urooj Fatima', id=45563}

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

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



```java
import java.util.Objects;

public 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;
    }

    // Getters
    public int getRed() {
        return red;
    }

    public int getGreen() {
        return green;
    }

    public int getBlue() {
        return blue;
    }

    // Override equals method
    @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;
    }

    // Override hashCode method
    @Override
    public int hashCode() {
        return Objects.hash(red, green, blue);
    }

    // Override toString method for easy display
    @Override
    public String toString() {
        return "Color{" +
                "red=" + red +
                ", green=" + green +
                ", blue=" + blue +
                '}';
    }

    // Main method for testing
    public static void main(String[] args) {
        Color color1 = new Color(255, 0, 0); // Red
        Color color2 = new Color(255, 0, 0); // Red
```



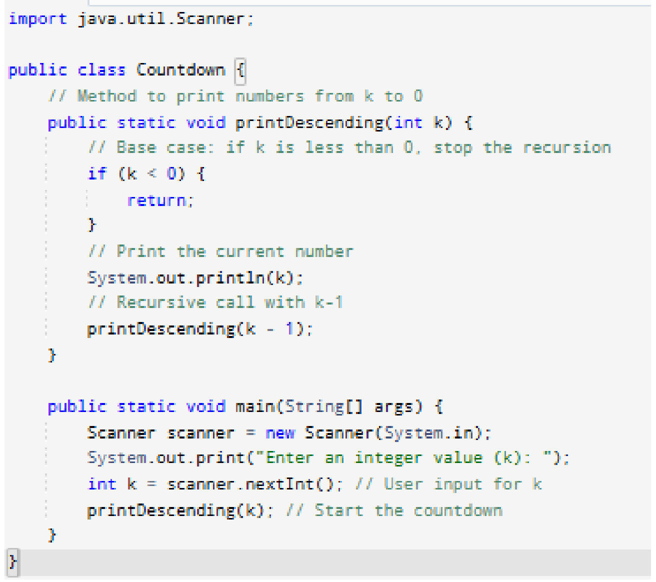
```
color1 equals color2: true
color1 equals color3: false
Color{red=255, green=0, blue=0}
Color{red=255, green=0, blue=0}
Color{red=0, green=255, blue=0}

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

LAB # 03
RECURSION

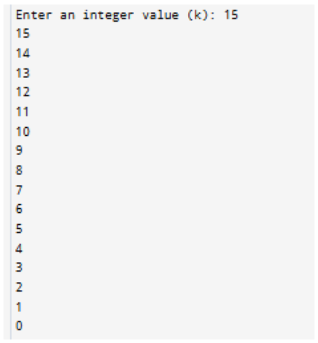
OBJECTIVE: To understand the complexities of the recursive functions and a way to reduce these complexities.

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.

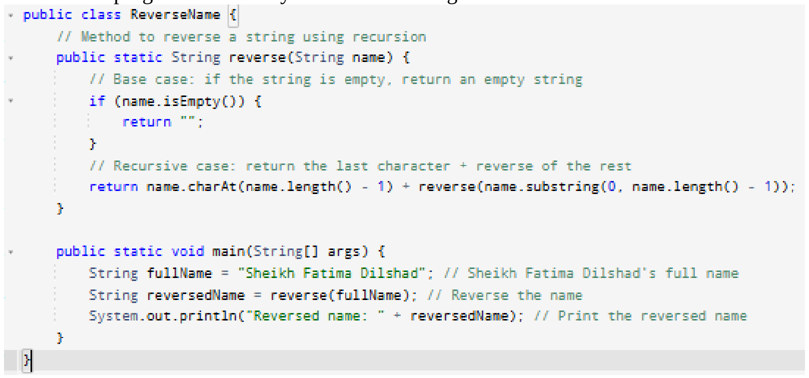


```
import java.util.Scanner;

public class Countdown {
    // Method to print numbers from k to 0
    public static void printDescending(int k) {
        // Base case: if k is less than 0, stop the recursion
        if (k < 0) {
            return;
        }
        // Print the current number
        System.out.println(k);
        // Recursive call with k-1
        printDescending(k - 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an integer value (k): ");
        int k = scanner.nextInt(); // User input for k
        printDescending(k); // Start the countdown
    }
}
```

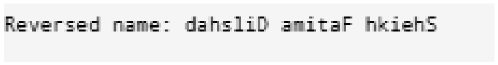


```
Enter an integer value (k): 15
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
```

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

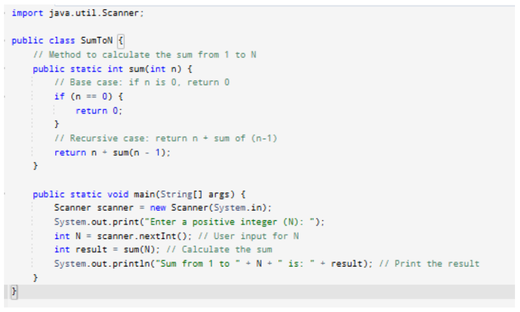


```
public class ReverseName {
    // Method to reverse a string using recursion
    public static String reverse(String name) {
        // Base case: if the string is empty, return an empty string
        if (name.isEmpty()) {
            return "";
        }
        // Recursive case: return the last character + reverse of the rest
        return name.charAt(name.length() - 1) + reverse(name.substring(0, name.length() - 1));
    }

    public static void main(String[] args) {
        String fullName = "Sheikh Fatima Dilshad"; // Sheikh Fatima Dilshad's full name
        String reversedName = reverse(fullName); // Reverse the name
        System.out.println("Reversed name: " + reversedName); // Print the reversed name
    }
}
```

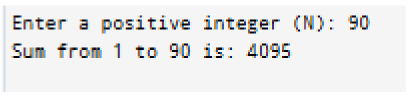


```
Reversed name: dahsliD amitaF hkiehS
```

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

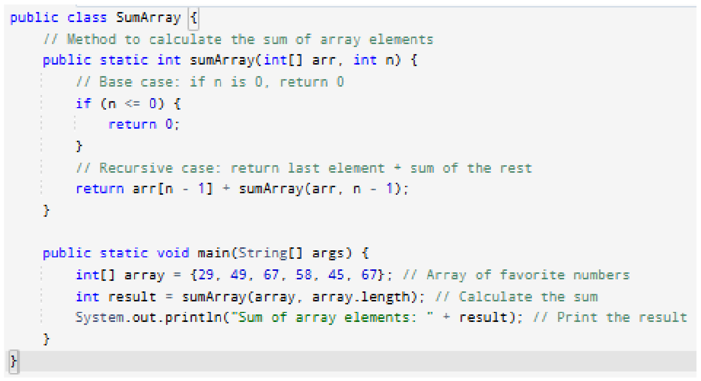


```java
import java.util.Scanner;

public class SumToN {
    // Method to calculate the sum from 1 to N
    public static int sum(int n) {
        // Base case: if n is 0, return 0
        if (n == 0) {
            return 0;
        }
        // Recursive case: return n + sum of (n-1)
        return n + sum(n - 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer (N): ");
        int N = scanner.nextInt(); // User input for N
        int result = sum(N); // Calculate the sum
        System.out.println("Sum from 1 to " + N + " is: " + result); // Print the result
    }
}
```

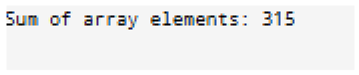


```
Enter a positive integer (N): 90
Sum from 1 to 90 is: 4095
```

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

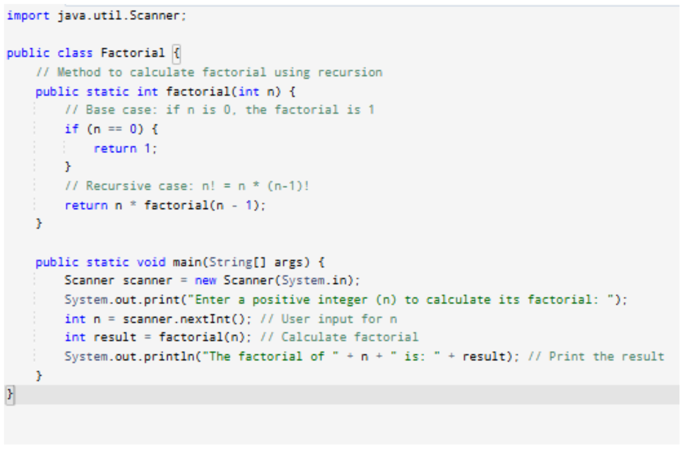


```java
public class SumArray {
    // Method to calculate the sum of array elements
    public static int sumArray(int[] arr, int n) {
        // Base case: if n is 0, return 0
        if (n <= 0) {
            return 0;
        }
        // Recursive case: return last element + sum of the rest
        return arr[n - 1] + sumArray(arr, n - 1);
    }

    public static void main(String[] args) {
        int[] array = {29, 49, 67, 58, 45, 67}; // Array of favorite numbers
        int result = sumArray(array, array.length); // Calculate the sum
        System.out.println("Sum of array elements: " + result); // Print the result
    }
}
```

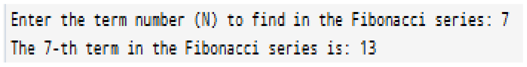


```
Sum of array elements: 315
```

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

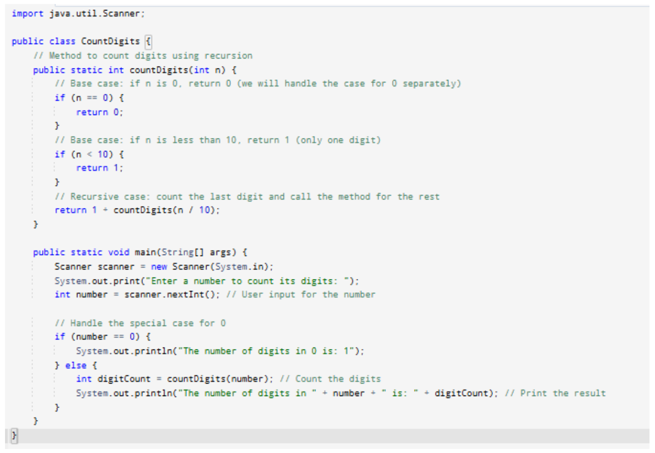


```java
import java.util.Scanner;

public class Factorial {
    // Method to calculate factorial using recursion
    public static int factorial(int n) {
        // Base case: if n is 0, the factorial is 1
        if (n == 0) {
            return 1;
        }
        // Recursive case: n! = n * (n-1)!
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer (n) to calculate its factorial: ");
        int n = scanner.nextInt(); // User input for n
        int result = factorial(n); // Calculate factorial
        System.out.println("The factorial of " + n + " is: " + result); // Print the result
    }
}
```

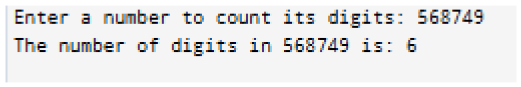


```
Enter the term number (N) to find in the Fibonacci series: 7
The 7-th term in the Fibonacci series is: 13
```

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



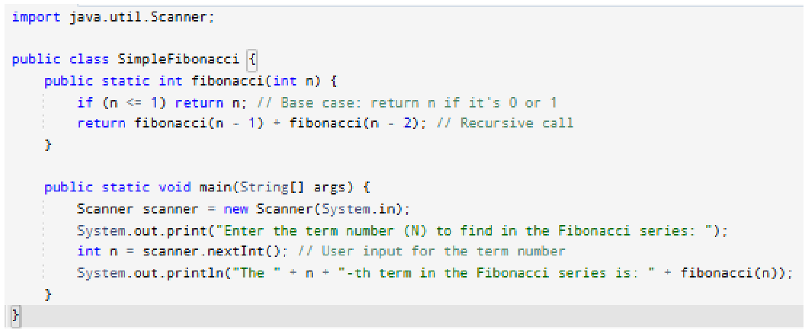
```java
import java.util.Scanner;

public class CountDigits {
    // Method to count digits using recursion
    public static int countDigits(int n) {
        // Base case: if n is 0, return 0 (we will handle the case for 0 separately)
        if (n == 0) {
            return 0;
        }
        // Base case: if n is less than 10, return 1 (only one digit)
        if (n < 10) {
            return 1;
        }
        // Recursive case: count the last digit and call the method for the rest
        return 1 + countDigits(n / 10);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to count its digits: ");
        int number = scanner.nextInt(); // User input for the number

        // Handle the special case for 0
        if (number == 0) {
            System.out.println("The number of digits in 0 is: 1");
        } else {
            int digitCount = countDigits(number); // Count the digits
            System.out.println("The number of digits in " + number + " is: " + digitCount); // Print the result
        }
    }
}
```



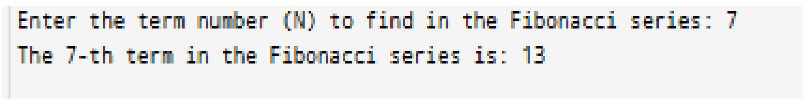
```
Enter a number to count its digits: 568749
The number of digits in 568749 is: 6
```

1. HOME TASK

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

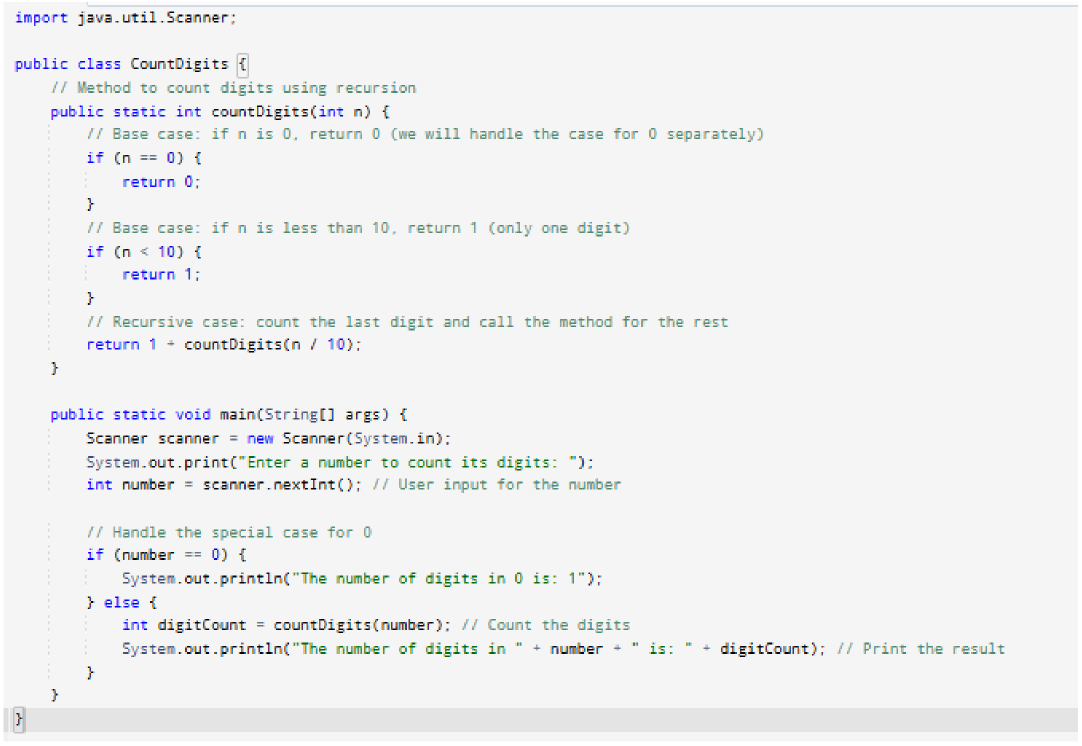


```java
import java.util.Scanner;

public class SimpleFibonacci {
    public static int fibonacci(int n) {
        if (n <= 1) return n; // Base case: return n if it's 0 or 1
        return fibonacci(n - 1) + fibonacci(n - 2); // Recursive call
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the term number (N) to find in the Fibonacci series: ");
        int n = scanner.nextInt(); // User input for the term number
        System.out.println("The " + n + "-th term in the Fibonacci series is: " + fibonacci(n));
    }
}
```

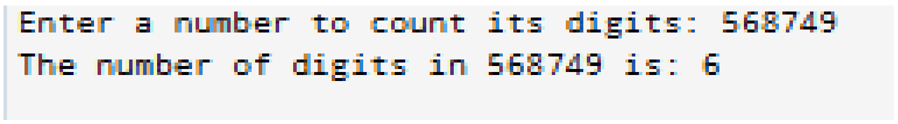


```
Enter the term number (N) to find in the Fibonacci series: 7
The 7-th term in the Fibonacci series is: 13
```

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

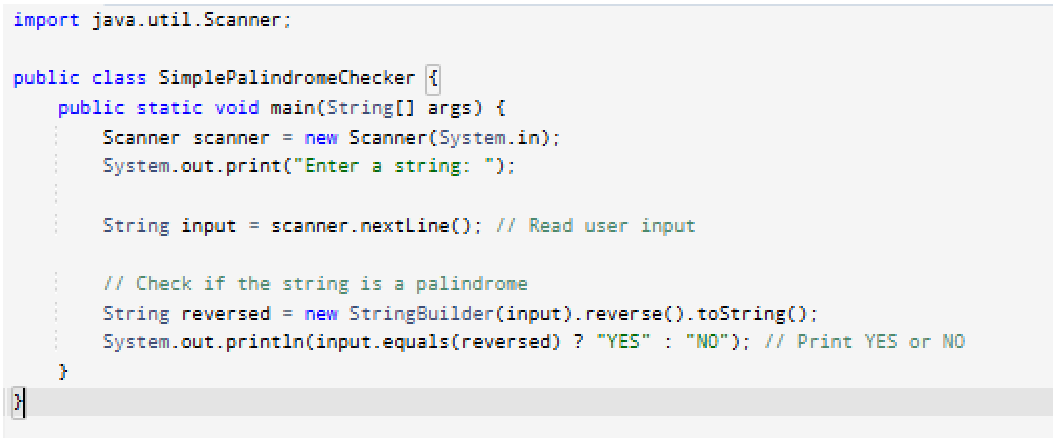


```java
import java.util.Scanner;

public class CountDigits {
    // Method to count digits using recursion
    public static int countDigits(int n) {
        // Base case: if n is 0, return 0 (we will handle the case for 0 separately)
        if (n == 0) {
            return 0;
        }
        // Base case: if n is less than 10, return 1 (only one digit)
        if (n < 10) {
            return 1;
        }
        // Recursive case: count the last digit and call the method for the rest
        return 1 + countDigits(n / 10);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to count its digits: ");
        int number = scanner.nextInt(); // User input for the number

        // Handle the special case for 0
        if (number == 0) {
            System.out.println("The number of digits in 0 is: 1");
        } else {
            int digitCount = countDigits(number); // Count the digits
            System.out.println("The number of digits in " + number + " is: " + digitCount); // Print the result
        }
    }
}
```

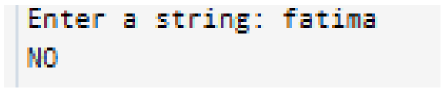


```
Enter a number to count its digits: 568749
The number of digits in 568749 is: 6
```

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".

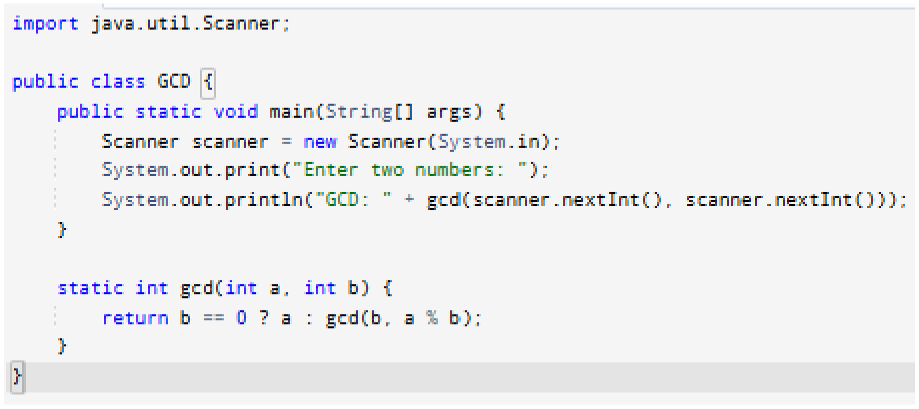


```java
import java.util.Scanner;

public class SimplePalindromeChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");

        String input = scanner.nextLine(); // Read user input

        // Check if the string is a palindrome
        String reversed = new StringBuilder(input).reverse().toString();
        System.out.println(input.equals(reversed) ? "YES" : "NO"); // Print YES or NO
    }
}
```

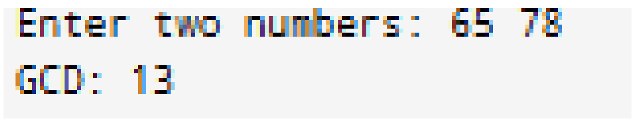


```
Enter a string: fatima
NO
```

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



```java
import java.util.Scanner;

public class GCD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        System.out.println("GCD: " + gcd(scanner.nextInt(), scanner.nextInt()));
    }

    static int gcd(int a, int b) {
        return b == 0 ? a : gcd(b, a % b);
    }
}
```



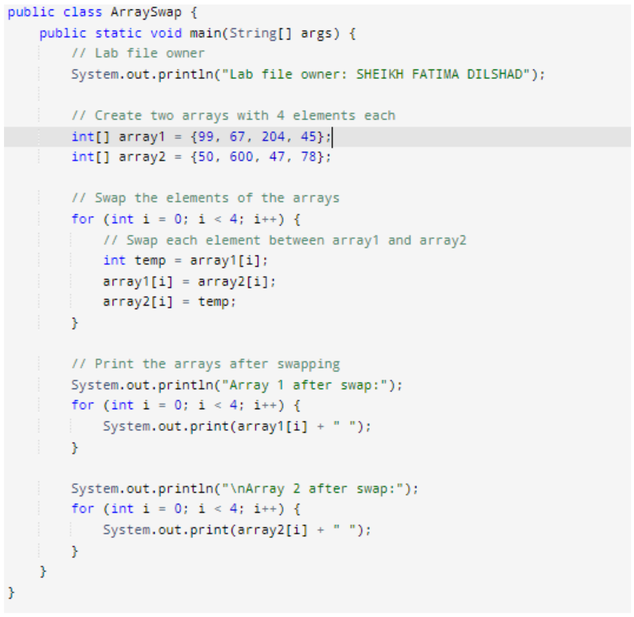
```
Enter two numbers: 65 78
GCD: 13
```

LAB # 04
ARRAYS IN JAVA

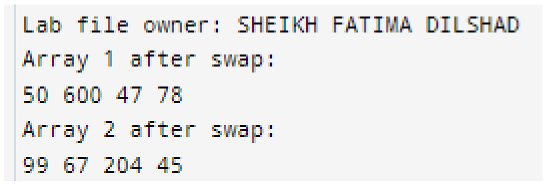
OBJECTIVE: To understand arrays and its memory allocation.

LAB TASKS

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

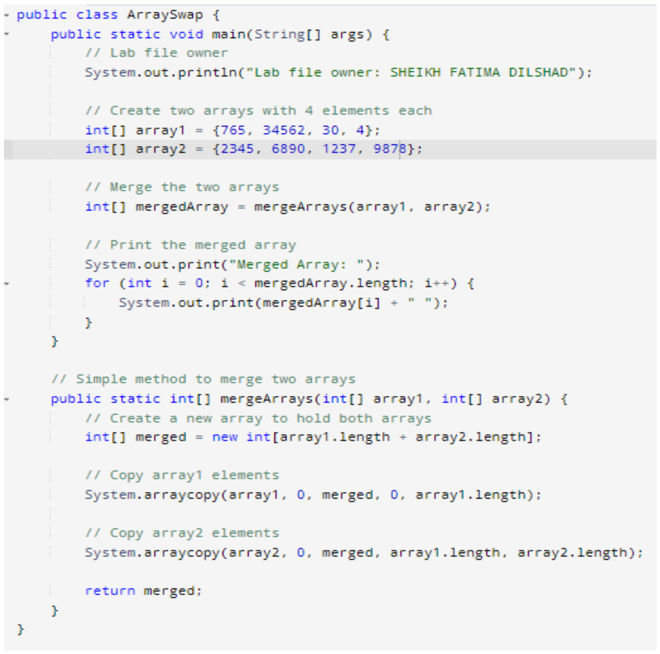


```java
public class ArraySwap {
    public static void main(String[] args) {
        // Lab file owner
        System.out.println("Lab file owner: SHEIKH FATIMA DILSHAD");

        // Create two arrays with 4 elements each
        int[] array1 = {99, 67, 204, 45};
        int[] array2 = {50, 600, 47, 78};

        // Swap the elements of the arrays
        for (int i = 0; i < 4; i++) {
            // Swap each element between array1 and array2
            int temp = array1[i];
            array1[i] = array2[i];
            array2[i] = temp;
        }

        // Print the arrays after swapping
        System.out.println("Array 1 after swap:");
        for (int i = 0; i < 4; i++) {
            System.out.print(array1[i] + " ");
        }

        System.out.println("\nArray 2 after swap:");
        for (int i = 0; i < 4; i++) {
            System.out.print(array2[i] + " ");
        }
    }
}
```

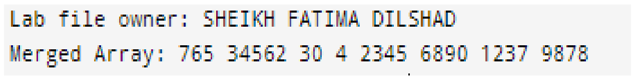


```
Lab file owner: SHEIKH FATIMA DILSHAD
Array 1 after swap:
50 600 47 78
Array 2 after swap:
99 67 204 45
```

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

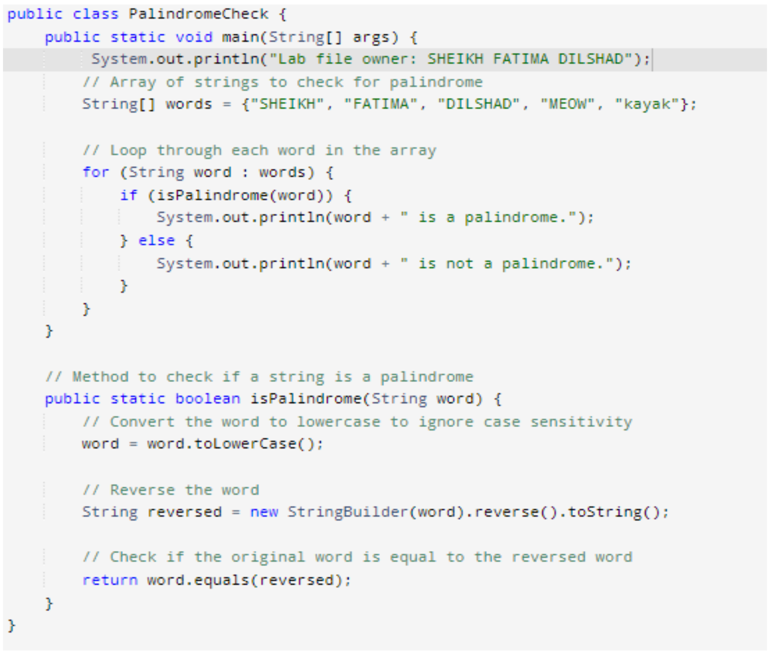


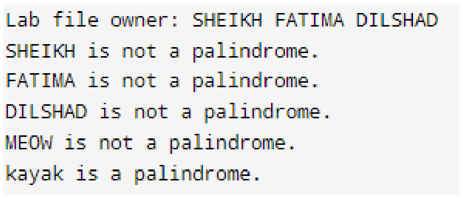
```java
public class ArraySwap {
    public static void main(String[] args) {
        // Lab file owner
        System.out.println("Lab file owner: SHEIKH FATIMA DILSHAD");

        // Create two arrays with 4 elements each
        int[] array1 = {765, 34562, 30, 4};
        int[] array2 = {2345, 6890, 1237, 9878};

        // Merge the two arrays
        int[] mergedArray = mergeArrays(array1, array2);

        // Print the merged array
        System.out.print("Merged Array: ");
        for (int i = 0; i < mergedArray.length; i++) {
            System.out.print(mergedArray[i] + " ");
        }
    }

    // Simple method to merge two arrays
    public static int[] mergeArrays(int[] array1, int[] array2) {
        // Create a new array to hold both arrays
        int[] merged = new int[array1.length + array2.length];

        // Copy array1 elements
        System.arraycopy(array1, 0, merged, 0, array1.length);

        // Copy array2 elements
        System.arraycopy(array2, 0, merged, array1.length, array2.length);

        return merged;
    }
}
```



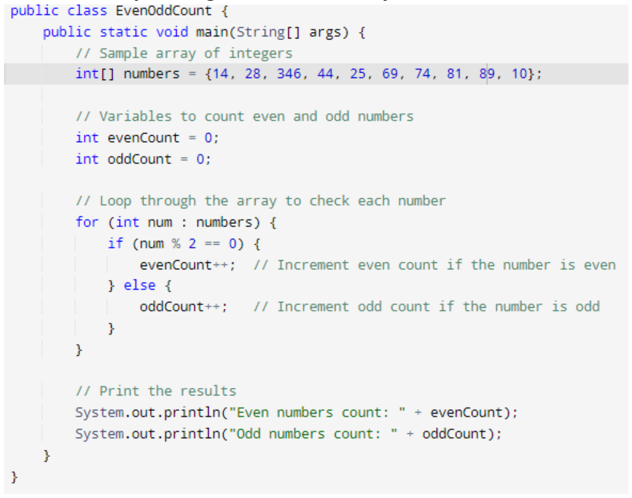
```
Lab file owner: SHEIKH FATIMA DILSHAD
Merged Array: 765 34562 30 4 2345 6890 1237 9878
```

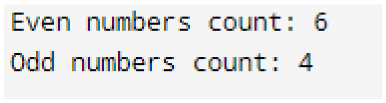
3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not.


```java
public class PalindromeCheck {
    public static void main(String[] args) {
         System.out.println("Lab file owner: SHEIKH FATIMA DILSHAD");
        // Array of strings to check for palindrome
        String[] words = {"SHEIKH", "FATIMA", "DILSHAD", "MEOW", "kayak"};

        // Loop through each word in the array
        for (String word : words) {
            if (isPalindrome(word)) {
                System.out.println(word + " is a palindrome.");
            } else {
                System.out.println(word + " is not a palindrome.");
            }
        }
    }

    // Method to check if a string is a palindrome
    public static boolean isPalindrome(String word) {
        // Convert the word to lowercase to ignore case sensitivity
        word = word.toLowerCase();

        // Reverse the word
        String reversed = new StringBuilder(word).reverse().toString();

        // Check if the original word is equal to the reversed word
        return word.equals(reversed);
    }
}
```

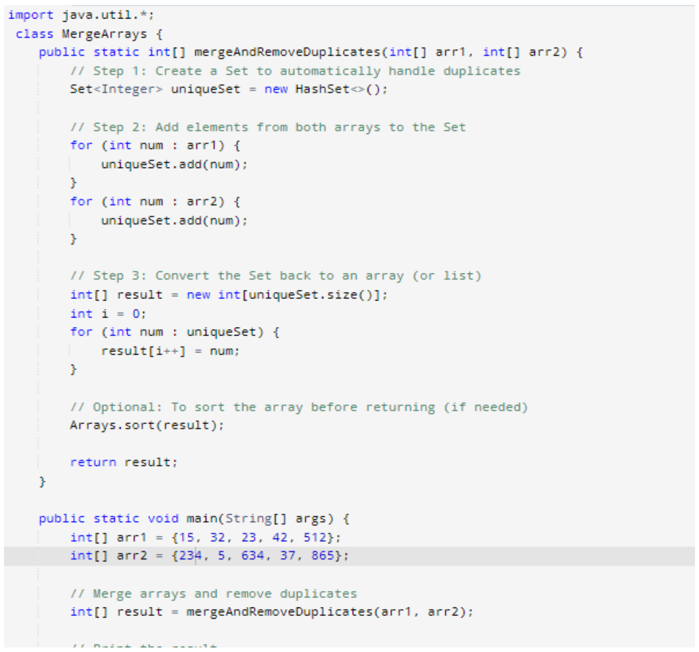

```
Lab file owner: SHEIKH FATIMA DILSHAD
SHEIKH is not a palindrome.
FATIMA is not a palindrome.
DILSHAD is not a palindrome.
MEOW is not a palindrome.
kayak is a palindrome.
```

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

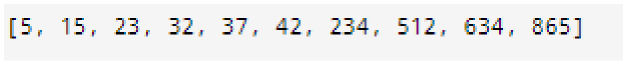

```java
public class EvenOddCount {
    public static void main(String[] args) {
        // Sample array of integers
        int[] numbers = {14, 28, 346, 44, 25, 69, 74, 81, 89, 10};

        // Variables to count even and odd numbers
        int evenCount = 0;
        int oddCount = 0;

        // Loop through the array to check each number
        for (int num : numbers) {
            if (num % 2 == 0) {
                evenCount++;  // Increment even count if the number is even
            } else {
                oddCount++;   // Increment odd count if the number is odd
            }
        }

        // Print the results
        System.out.println("Even numbers count: " + evenCount);
        System.out.println("Odd numbers count: " + oddCount);
    }
}
```


```
Even numbers count: 6
Odd numbers count: 4
```

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



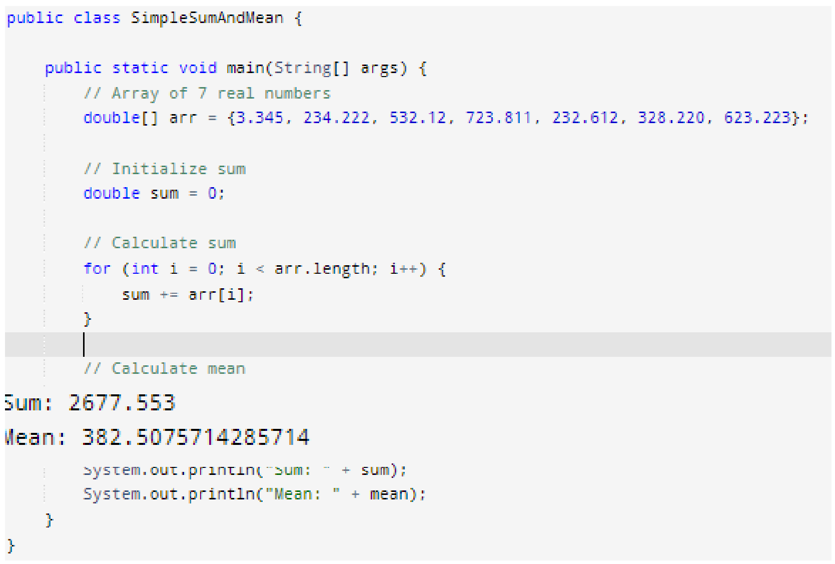
```java
import java.util.*;
 class MergeArrays {
    public static int[] mergeAndRemoveDuplicates(int[] arr1, int[] arr2) {
        // Step 1: Create a Set to automatically handle duplicates
        Set<Integer> uniqueSet = new HashSet<>();

        // Step 2: Add elements from both arrays to the Set
        for (int num : arr1) {
            uniqueSet.add(num);
        }
        for (int num : arr2) {
            uniqueSet.add(num);
        }

        // Step 3: Convert the Set back to an array (or list)
        int[] result = new int[uniqueSet.size()];
        int i = 0;
        for (int num : uniqueSet) {
            result[i++] = num;
        }

        // Optional: To sort the array before returning (if needed)
        Arrays.sort(result);

        return result;
    }

    public static void main(String[] args) {
        int[] arr1 = {15, 32, 23, 42, 512};
        int[] arr2 = {234, 5, 634, 37, 865};

        // Merge arrays and remove duplicates
        int[] result = mergeAndRemoveDuplicates(arr1, arr2);
```



```
[5, 15, 23, 32, 37, 42, 234, 512, 634, 865]
```

Home task:

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.

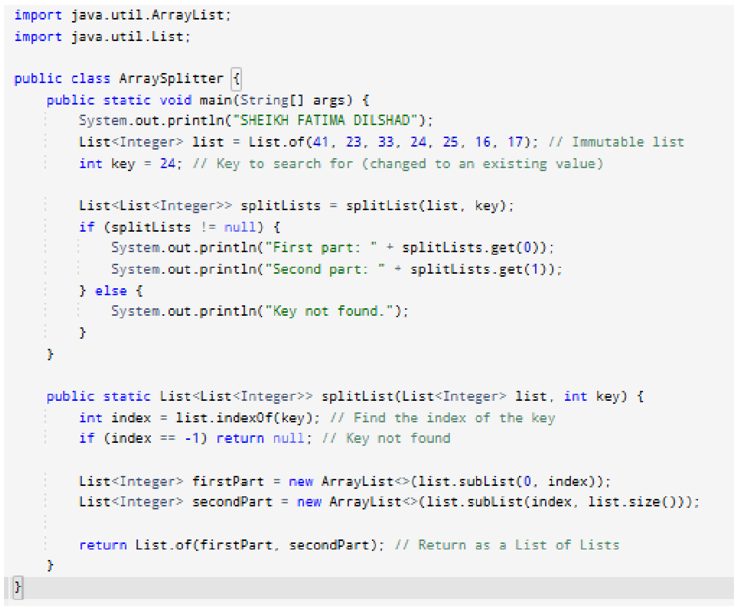


```java
public class SimpleSumAndMean {

    public static void main(String[] args) {
        // Array of 7 real numbers
        double[] arr = {3.345, 234.222, 532.12, 723.811, 232.612, 328.220, 623.223};

        // Initialize sum
        double sum = 0;

        // Calculate sum
        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];
        }

        // Calculate mean
```

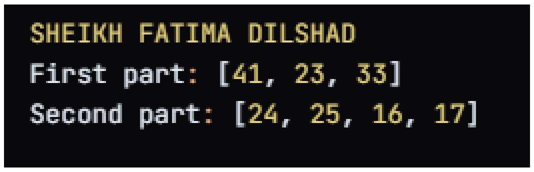
```
Sum: 2677.553
Mean: 382.5075714285714
```

```java
        System.out.println("Sum: " + sum);
        System.out.println("Mean: " + mean);
    }
}
```

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.

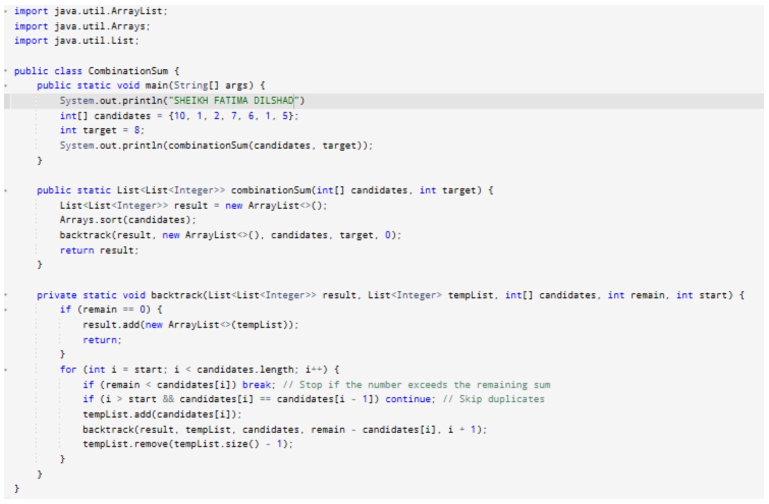


```java
import java.util.ArrayList;
import java.util.List;

public class ArraySplitter {
    public static void main(String[] args) {
        System.out.println("SHEIKH FATIMA DILSHAD");
        List<Integer> list = List.of(41, 23, 33, 24, 25, 16, 17); // Immutable list
        int key = 24; // Key to search for (changed to an existing value)

        List<List<Integer>> splitLists = splitList(list, key);
        if (splitLists != null) {
            System.out.println("First part: " + splitLists.get(0));
            System.out.println("Second part: " + splitLists.get(1));
        } else {
            System.out.println("Key not found.");
        }
    }

    public static List<List<Integer>> splitList(List<Integer> list, int key) {
        int index = list.indexOf(key); // Find the index of the key
        if (index == -1) return null; // Key not found

        List<Integer> firstPart = new ArrayList<>(list.subList(0, index));
        List<Integer> secondPart = new ArrayList<>(list.subList(index, list.size()));

        return List.of(firstPart, secondPart); // Return as a List of Lists
    }
}
```

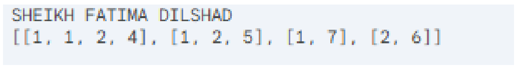


```
SHEIKH FATIMA DILSHAD
First part: [41, 23, 33]
Second part: [24, 25, 16, 17]
```

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

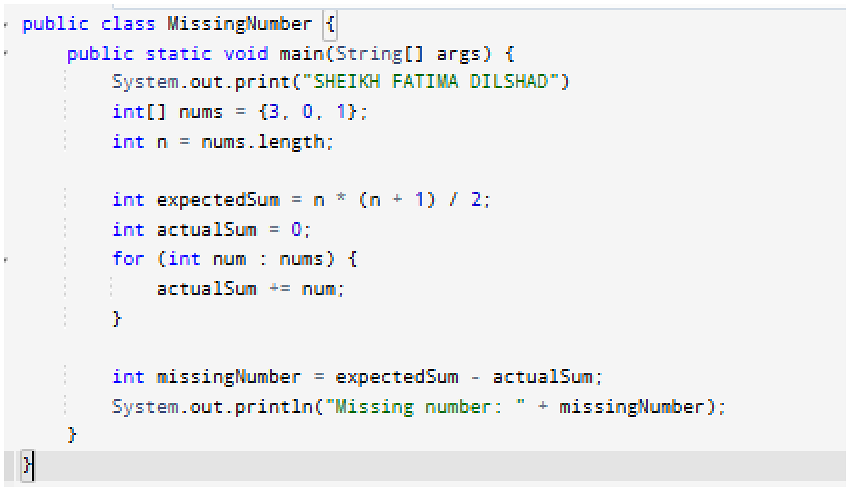


```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CombinationSum {
    public static void main(String[] args) {
        System.out.println("SHEIKH FATIMA DILSHAD")
        int[] candidates = {10, 1, 2, 7, 6, 1, 5};
        int target = 8;
        System.out.println(combinationSum(candidates, target));
    }

    public static List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(candidates);
        backtrack(result, new ArrayList<>(), candidates, target, 0);
        return result;
    }

    private static void backtrack(List<List<Integer>> result, List<Integer> tempList, int[] candidates, int remain, int start) {
        if (remain == 0) {
            result.add(new ArrayList<>(tempList));
            return;
        }
        for (int i = start; i < candidates.length; i++) {
            if (remain < candidates[i]) break; // Stop if the number exceeds the remaining sum
            if (i > start && candidates[i] == candidates[i - 1]) continue; // Skip duplicates
            tempList.add(candidates[i]);
            backtrack(result, tempList, candidates, remain - candidates[i], i + 1);
            tempList.remove(tempList.size() - 1);
        }
    }
}
```

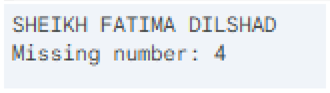


```
SHEIKH FATIMA DILSHAD
[[1, 1, 2, 4], [1, 2, 5], [1, 7], [2, 6]]
```

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.

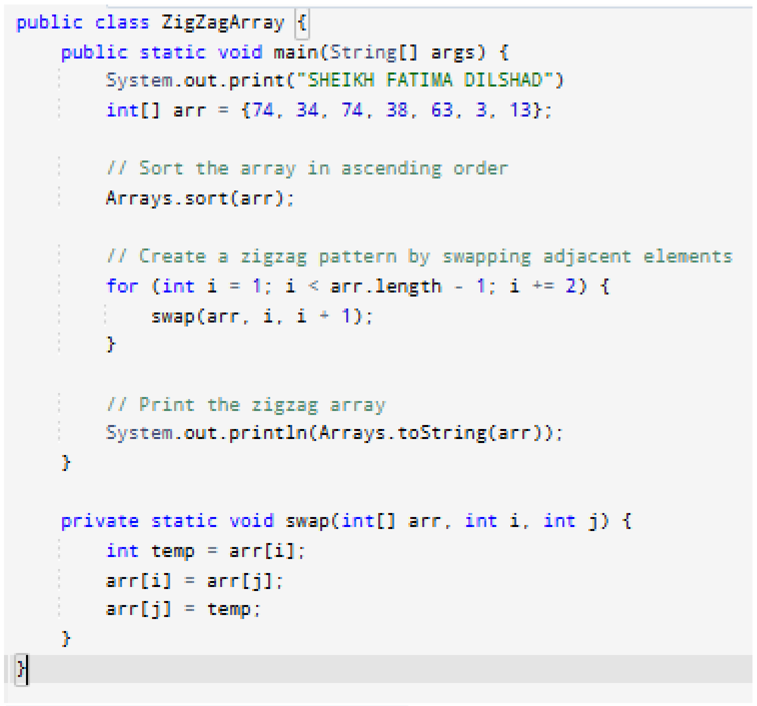


```
public class MissingNumber {
    public static void main(String[] args) {
        System.out.print("SHEIKH FATIMA DILSHAD")
        int[] nums = {3, 0, 1};
        int n = nums.length;

        int expectedSum = n * (n + 1) / 2;
        int actualSum = 0;
        for (int num : nums) {
            actualSum += num;
        }

        int missingNumber = expectedSum - actualSum;
        System.out.println("Missing number: " + missingNumber);
    }
}
```

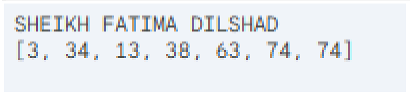


```
SHEIKH FATIMA DILSHAD
Missing number: 4
```

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.



```
public class ZigZagArray {
    public static void main(String[] args) {
        System.out.print("SHEIKH FATIMA DILSHAD")
        int[] arr = {74, 34, 74, 38, 63, 3, 13};

        // Sort the array in ascending order
        Arrays.sort(arr);

        // Create a zigzag pattern by swapping adjacent elements
        for (int i = 1; i < arr.length - 1; i += 2) {
            swap(arr, i, i + 1);
        }

        // Print the zigzag array
        System.out.println(Arrays.toString(arr));
    }

    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```



```
SHEIKH FATIMA DILSHAD
[3, 34, 13, 38, 63, 74, 74]
```