Experiment No 3

Name: Bhuvi Ghosh **Sap ID:** 60009210191 **Batch:** D22

Aim: -To Implement Array, String and Vector in Java. Theory: -

1.Array in Java

An array is a collection of similar types of data. For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

String [] array = new String [100];

Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed**.**

How to declare an array in Java?

In Java, here is how we can declare an array.

dataType [] arrayName;

dataType - it can be primitive data types like int, char, double, byte, etc. or Java objects

arrayName - it is an identifier

For example,

double[] data; Here, data is an array that can hold values of type double.

To define the number of elements that an array can hold, we have to allocate memory for the array in Java. For example,

```
// declare an array
double[] data;

// allocate memory
data = new double [10];
```

Here, the array can store 10 elements. We can also say that the size or length of the array is 10.

In Java, we can declare and allocate the memory of an array in one single statement. For example,

double [] data = new double [10];

How to Initialize Arrays in Java?

In Java, we can initialize arrays during declaration. For example,

```
//declare and initialize and array
int [] age = {12, 4, 5, 2, 5};
```

Here, we have created an array named age and initialized it with the values inside the curly brackets.

Here we have not provided the size of the array. In this case, the Java compiler automatically specifies the size by counting the number of elements in the array (i.e. 5).In the Java array,

each memory location is associated with a number. The number is known as an array index. We can also initialize arrays in Java, using the index number. For example,

```
// declare an array
int[] age = new int[5];
// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5;
```

How to Access Elements of an Array in Java?

We can access the element of an array using the index number. Here is the syntax for accessing elements of an array,

```
// access array elements
array[index]
```

Multidimensional Arrays

Arrays we have mentioned till now are called one-dimensional arrays. However, we can declare multidimensional arrays in Java. A multidimensional array is an array of arrays. That is, each element of a multidimensional array is an array itself. For example,

```
double [][] matrix = {{1.2, 4.3, 4.0},
    {4.1, -1.1}
};
```

Here, we have created a multidimensional array named matrix. It is a 2-dimensional array.

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

2-dimensional Array

Initialization of a 2-dimensional array in Java.

```
int [][] a = {
    {1, 2, 3},
    {4, 5, 6, 9},
```

```
    {7},
};
```

Copying Arrays Using arraycopy () method

In Java, the System class contains a method named arraycopy () to copy arrays. This method is a better approach to copy arrays than the above two. The arraycopy () method allows you to copy a specified portion of the source array to the destination array. For example,

arraycopy (Object src, int srcPos, Object dest, int destPos, int length)

Here,
src - source array you want to copy
srcPos - starting position (index) in the source array
dest - destination array where elements will be copied from the source
destPos - starting position (index) in the destination array
length - number of elements to copy

2. Strings in Java

Strings in Java are Objects that are backed internally by a char array. Since arrays are immutable (cannot grow), Strings are immutable as well. Whenever a change to a String is made, an entirely new String is created.

Syntax:

<String_Type> <string_variable> = "<sequence_of_string>";

Example:

String str = "Strings in java ";

Memory allotment of String

Whenever a String Object is created as a literal, the object will be created in String constant pool. This allows JVM to optimize the initialization of String literal.

For example:

String str = " Strings in java ";

The string can also be declared using new operator i.e. dynamically allocated. In case of String are dynamically allocated they are assigned a new memory location in heap. This string will not be added to String constant pool.

For example:

String str = new String ("Strings in java ");

StringBuffer is a peer class of String that provides much of the functionality of strings. The string represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

Syntax:

StringBuffer s = new StringBuffer ("**Strings in java**");

3. Wrapper classes in Java

The wrapper classes in Java are used to convert primitive types (int, char, float, etc) into corresponding objects. All primitive types have corresponding wrapper classes.. These wrapper classes come under the java.util package.

Why we need Wrapper Class

- Wrapper Class will convert primitive data types into objects. The objects are necessary if we wish to modify the arguments passed into the method (because primitive types are passed by value).
- Data structures in the Collection framework such as ArrayList and Vector store only the objects (reference types) and not the primitive types.
- The object is needed to support synchronization in multithreading.

Example

```
public class Main {
        public static void main(String[] args) {
          Integer myInt = 5;
          Double myDouble = 5.99;
          Character myChar = 'A';
          System.out.println(myInt);
          System.out.println(myDouble);
          System.out.println(myChar);
 }
```

4. Java Vector

The Vector class is an implementation of the List interface that allows us to create resizable-arrays similar to the ArrayList class.

Java Vector vs. ArrayList

In Java, both Array and vector implements the List interface and provides the same functionalities. However, there exist some differences between them.

The Vector class synchronizes each individual operation. This means whenever we want to perform some operation on vectors, the Vector class automatically applies a lock to that operation.

It is because when one thread is accessing a vector, and at the same time another thread tries to access it, an exception called `ConcurrentModificationException` is generated. Hence, this continuous use of lock for each operation makes vectors less efficient.

However, in array lists, methods are not synchronized. Instead, it uses the `Collections.synchronizedList()` method that synchronizes the list as a whole.

Creating a Vector

Here is how we can create vectors in Java.

`Vector<Type> vector = new Vector<>();`

Here, Type indicates the type of a linked list. For example,

// create Integer type linked list
Vector<Integer> vector= new Vector<>();

// create String type linked list
Vector<String> vector= new Vector<>();

5. Lab Assignments to complete in this session

i. WAP to find whether the entered 4 digit number is vampire or not. Combination of digits from this number forms 2-digit number. When they are multiplied by each other we get the original number. (1260=21*60, 1395=15*93, 1530=30*51

```
import java.util.Scanner;

public class VampireNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a 4-digit number: ");
        int number = scanner.nextInt();
        scanner.close();

        if (isVampireNumber(number)) {
            System.out.println(number + " is a vampire number.");
        } else {
            System.out.println(number + " is not a vampire number.");
        }
    }

    // Function to check if a number is a vampire number
    public static boolean isVampireNumber(int number) {
        if (number < 1000 || number > 9999) {
            return false;
        }

        String numStr = String.valueOf(number);

        for (int i = 10; i <= 99; i++) {
            for (int j = i; j <= 99; j++) {
                if (i * j == number) {
                    String ijStr = String.valueOf(i) + String.valueOf(j);
                    if (areAnagrams(numStr, ijStr)) {
                        return true;
                    }
                }
            }
        }

        return false;
    }

    // Function to check if two strings are anagrams
    public static boolean areAnagrams(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }
```

```
        for (int i = 0; i < str1.length(); i++) {
            if (str1.indexOf(str2.charAt(i)) == -1) {
                return false;
            }
        }

        return true;
    }
}
```

OUTPUT:

```
PS D:\Sem 5\Java\Java lab>javac VampireNumber.java
PS D:\Sem 5\Java\Java lab>java VampireNumber
Enter a 4-digit number: 1260
1260 is a vampire number.
PS D:\Sem 5\Java\Java lab>java VampireNumber
Enter a 4-digit number: 1111
1111 is not a vampire number.
```

ii. WAP to display the following using irregular arrays

1
2 3
4 5 6

```
public class IrregularArray {
    public static void main(String[] args) {
        int[][] array = {
            {1},
            {2, 3},
            {4, 5, 6}
        };

        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(array[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
PS D:\Sem 5\Java\Java lab>javac IrregularArray.java
PS D:\Sem 5\Java\Java lab>java IrregularArray
1
2 3
4 5 6
```

iii. WAP a java program for the following problem statement

You have been given an array of positive integers A1,A2,...,An with legnth N and you have to print an array of same legnth(N) where the values in the new array are the sum of every number in the array, except the number at that index.

Input:

The first line of input contains a single integer T denoting the number of test cases.
Each test cases contain two lines.First line contains N, the length of the array and second

line contains N space separated positive integers.

Output:

For each test case, output a single array of same length.

Constraints:

$1 \leq T \leq 100$

$1 \leq N \leq 105$

0 <= A[i] <= 109

Example:

Input

2

4

1 2 3 4

3

4 5 6

Output

9 8 7 6

11 10 9

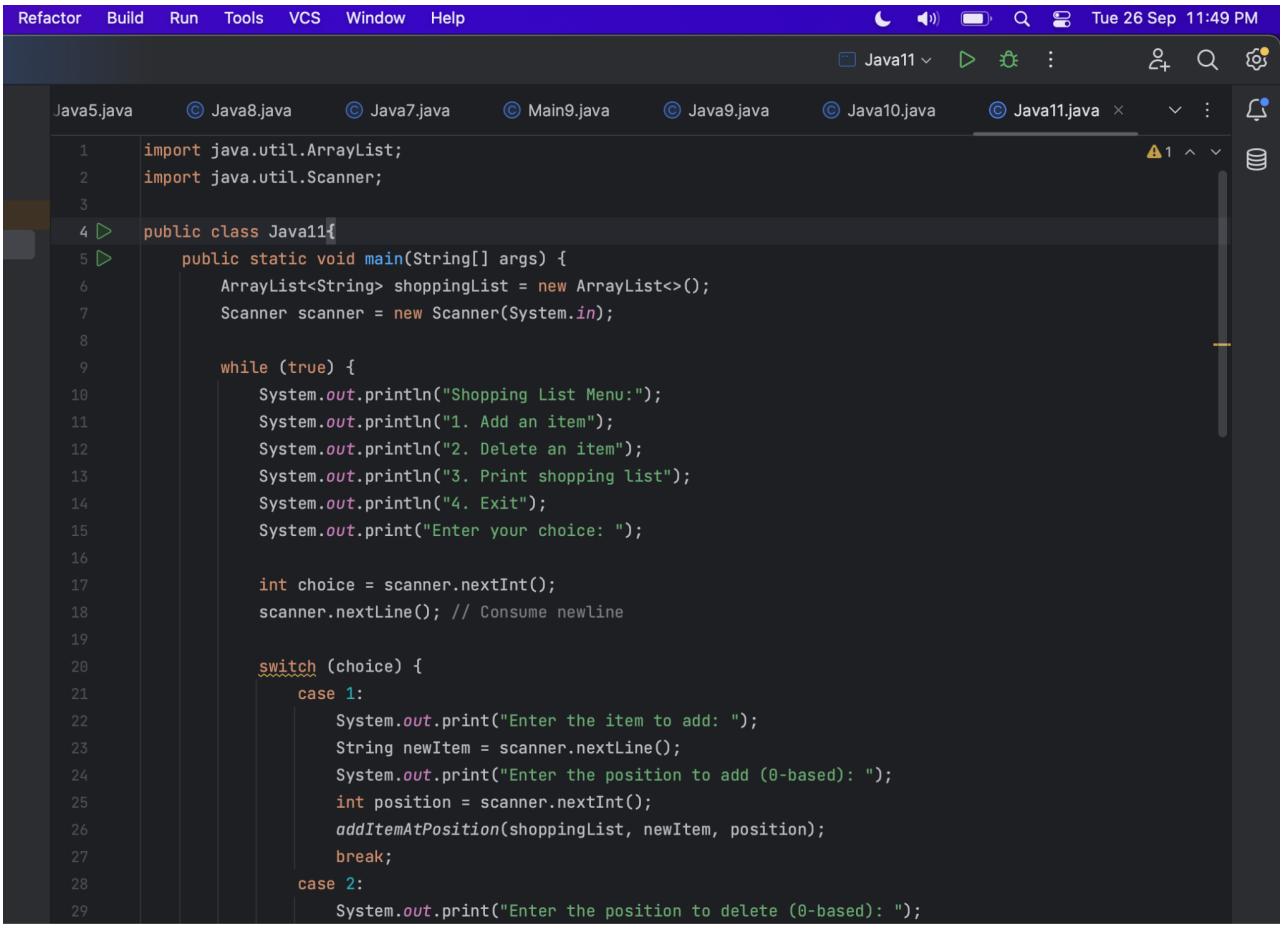
```
import java.util.Scanner;

public class SumArrayExceptIndex {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the no. of Times you want to run:");
        int t = scanner.nextInt();

        for (int i = 0; i < t; i++) {
            System.out.printf("Enter the no. of elements for the array no.
%d:",i+1);
            int n = scanner.nextInt();
            int[] arr = new int[n];
            for (int j = 0; j < n; j++) {
                arr[j] = scanner.nextInt();
            }

            int[] result = sumExceptIndex(arr);
            for (int k = 0; k < n; k++) {
                System.out.print(result[k] + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
    public static int[] sumExceptIndex(int[] arr) {
        int n = arr.length;
        int[] result = new int[n];

        // Calculate the total sum of the array
        int totalSum = 0;
        for (int i = 0; i < n; i++) {
```

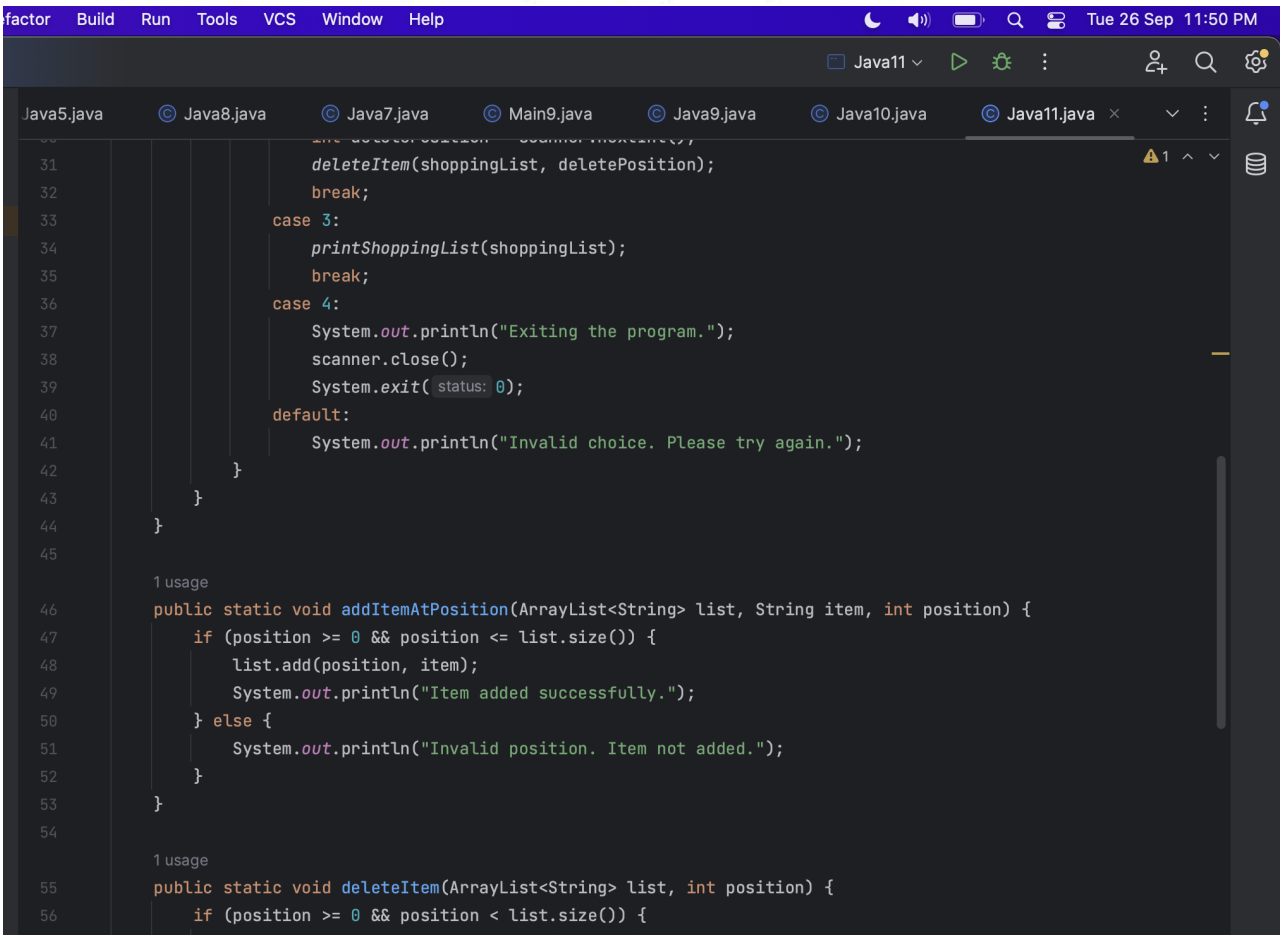
```
            totalSum += arr[i];
        }

        // Calculate the result array
        for (int i = 0; i < n; i++) {
            result[i] = totalSum - arr[i];
        }

        return result;
    }
}
```

OUTPUT:

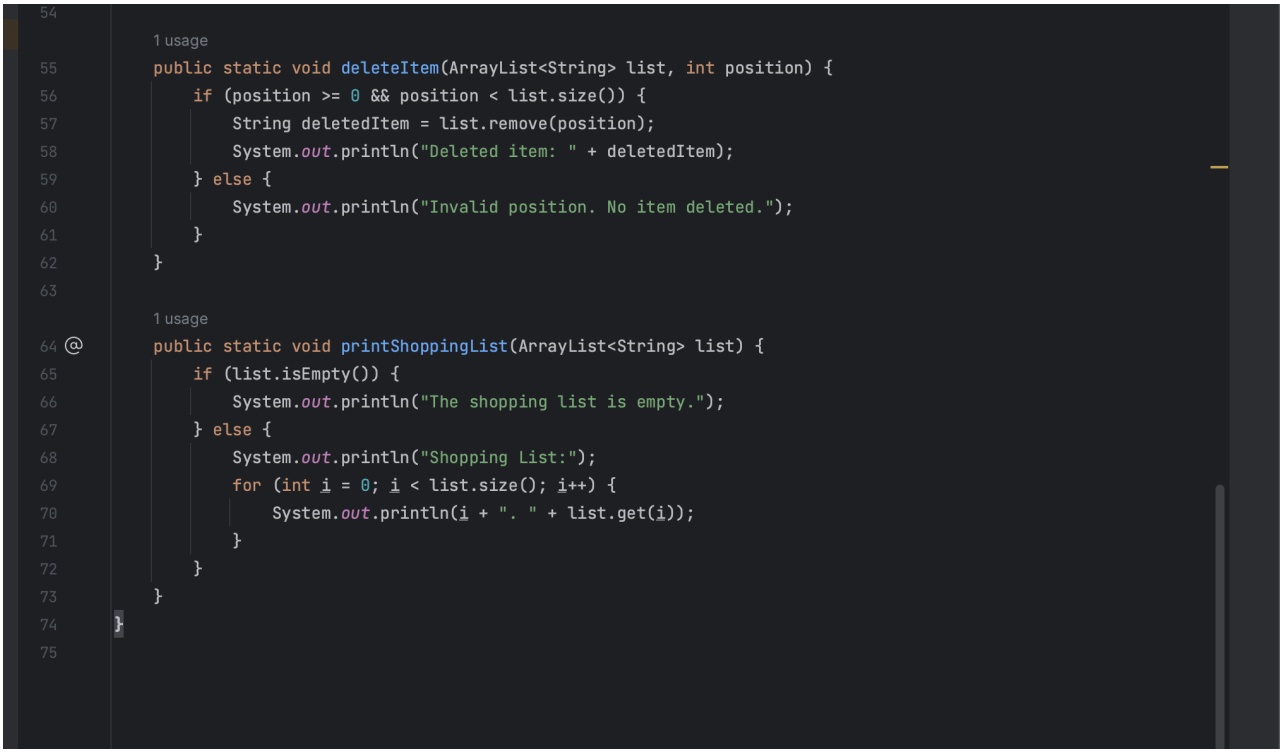
```
PS D:\Sem 5\Java\Java lab>javac SumArrayExceptIndex.java
PS D:\Sem 5\Java\Java lab>java SumArrayExceptIndex
Enter the no. of Times you want to run:2
Enter the no. of elements for the array no. 1:4
1 2 3 4
9 8 7 6
Enter the no. of elements for the array no. 2:3
4 5 6
11 10
```

iv. WAP that accepts a shopping list of items and performs the following operations: Add an item at a specified location, delete an item in the list, and print the contents of the vector





```
                    deleteItem(shoppingList, deletePosition);
                    break;
                case 3:
                    printShoppingList(shoppingList);
                    break;
                case 4:
                    System.out.println("Exiting the program.");
                    scanner.close();
                    System.exit( status: 0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }

    1 usage
    public static void addItemAtPosition(ArrayList<String> list, String item, int position) {
        if (position >= 0 && position <= list.size()) {
            list.add(position, item);
            System.out.println("Item added successfully.");
        } else {
            System.out.println("Invalid position. Item not added.");
        }
    }

    1 usage
    public static void deleteItem(ArrayList<String> list, int position) {
        if (position >= 0 && position < list.size()) {
```

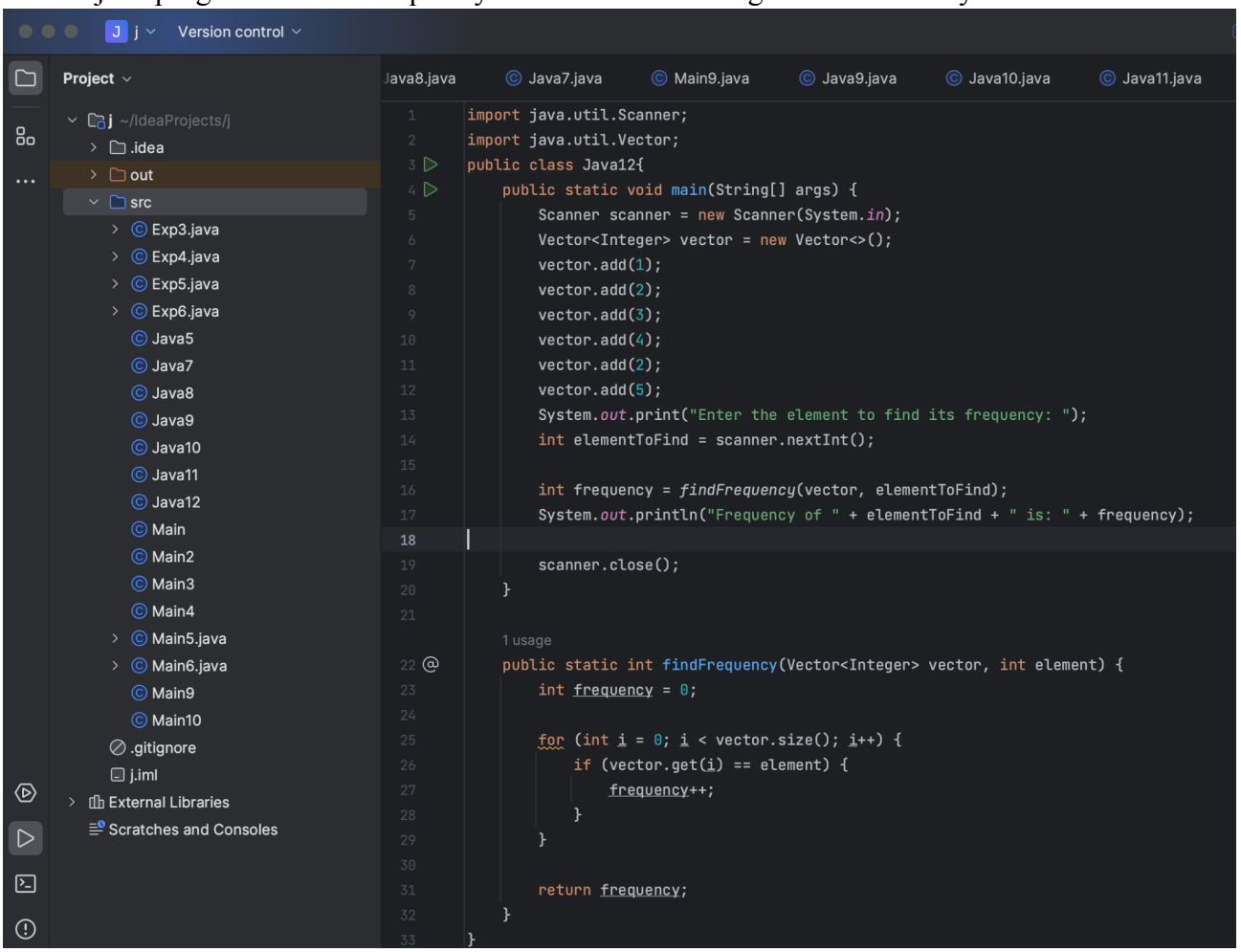


```
    1 usage
    public static void deleteItem(ArrayList<String> list, int position) {
        if (position >= 0 && position < list.size()) {
            String deletedItem = list.remove(position);
            System.out.println("Deleted item: " + deletedItem);
        } else {
            System.out.println("Invalid position. No item deleted.");
        }
    }

    1 usage
    public static void printShoppingList(ArrayList<String> list) {
        if (list.isEmpty()) {
            System.out.println("The shopping list is empty.");
        } else {
            System.out.println("Shopping List:");
            for (int i = 0; i < list.size(); i++) {
                System.out.println(i + ". " + list.get(i));
            }
        }
    }
}
```


Run
Java11
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/Intel
Shopping List Menu:
1. Add an item
2. Delete an item
3. Print shopping list
4. Exit
Enter your choice: 1
Enter the item to add: Bag
Enter the position to add (0-based): 2
Invalid position. Item not added.
Shopping List Menu:
1. Add an item
2. Delete an item
3. Print shopping list
4. Exit


Invalid position. Item not added.
Shopping List Menu:
1. Add an item
2. Delete an item
3. Print shopping list
4. Exit
Enter your choice: 1
Enter the item to add: Pen
Enter the position to add (0-based): 1
Invalid position. Item not added.
Shopping List Menu:
1. Add an item
2. Delete an item
3. Print shopping list
4. Exit
Enter your choice: 4
Exiting the program.
Process finished with exit code 0

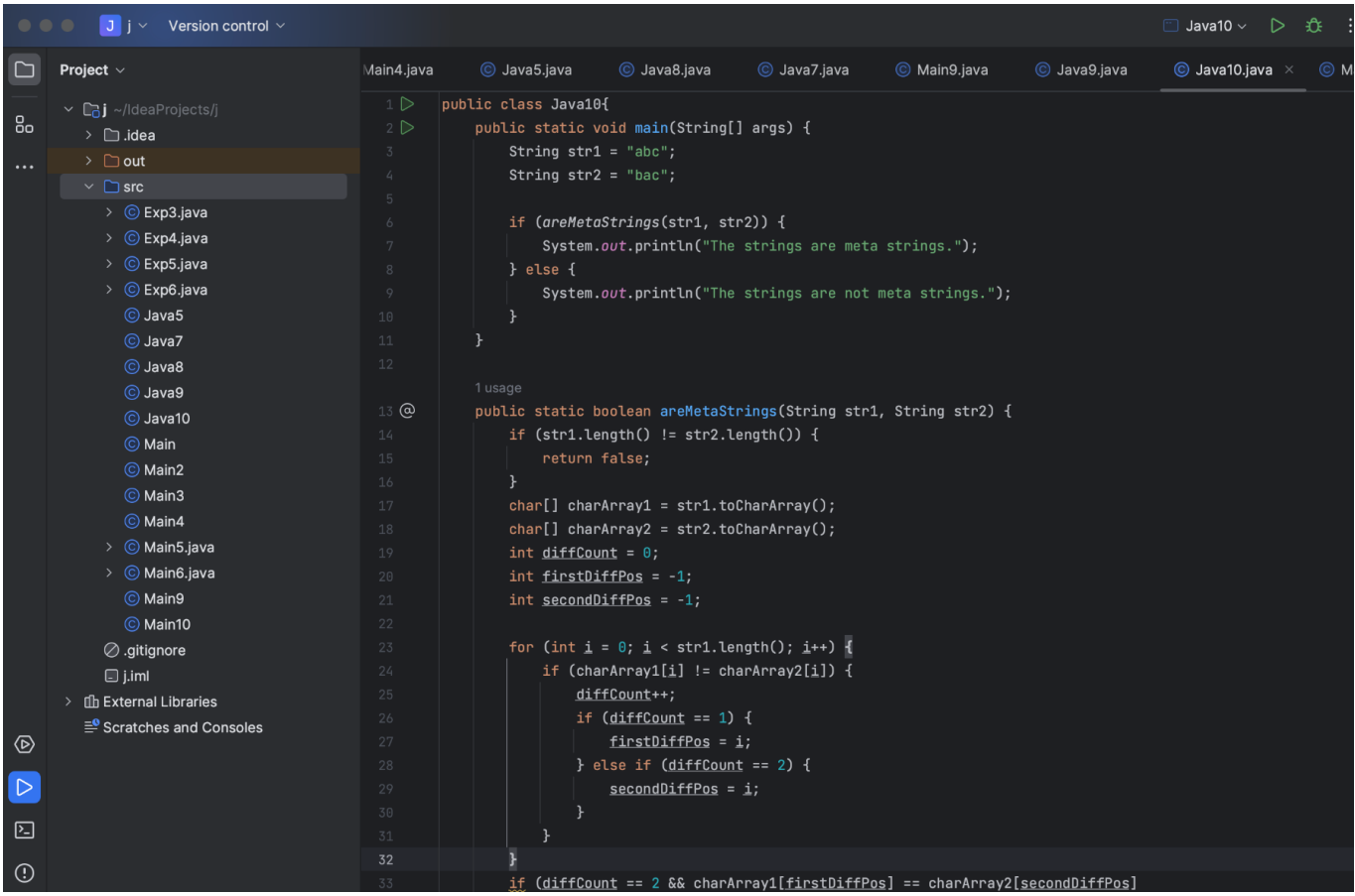
v. Write a java programs to find frequency of an element in the given Vector array.



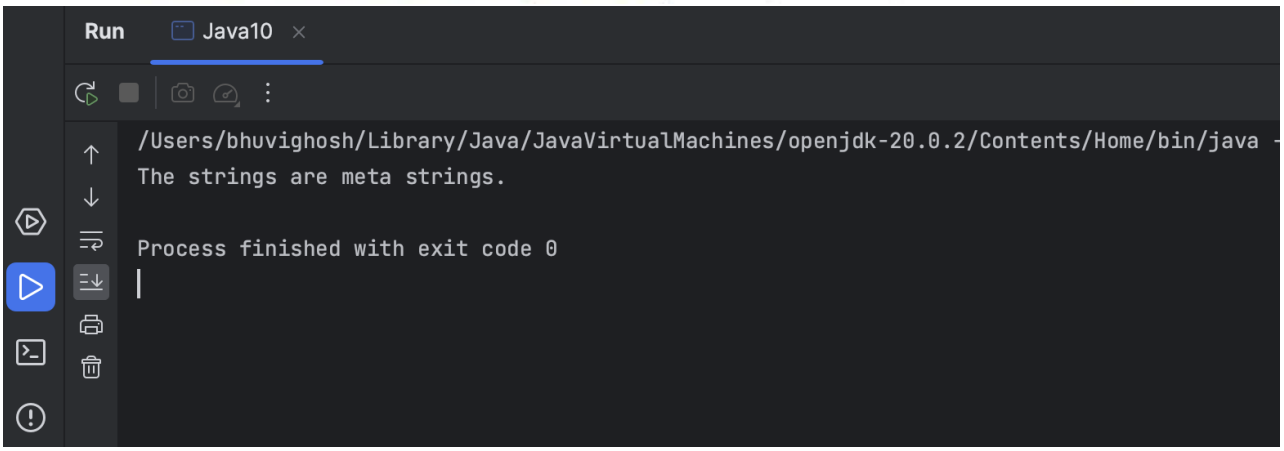
vi. WAP to check if 2 strings are Meta strings or not. Meta strings are the strings which can be made equal by exactly one swap in any of the strings. Equal string are not considered here as Meta strings.

Example: str1 = "geeks", str2 = "keegs"
By just swapping 'k' and 'g' in any of string, both will become same. Example: str1 = "Converse", str2 = "Conserve"
By just swapping 'v' and's' in any of string, both will become same. Algorithm (if reqd):

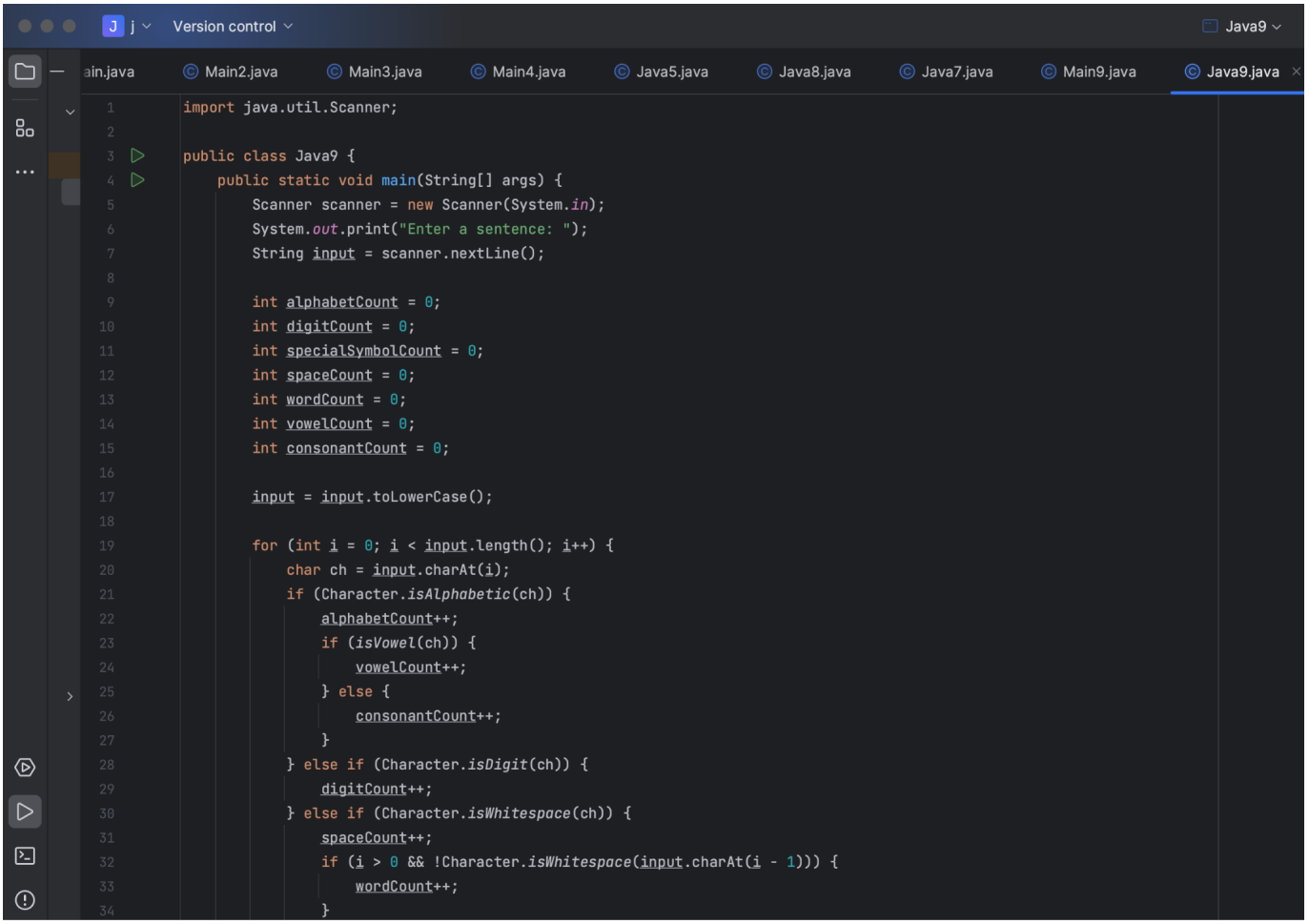
1. Check if both strings are of equal length or not, if not return false.
2. Otherwise, start comparing both strings and count number of unmatched characters and also store the index of unmatched characters.
3. If unmatched characters are more than 2 then return false.
4. Otherwise check if on swapping any of these two characters in any string would make the string equal or not.
5. If yes then return true. Otherwise return false.




```
        if (diffCount == 2 && charArray1[firstDiffPos] == charArray2[secondDiffPos]
                && charArray1[secondDiffPos] == charArray2[firstDiffPos]) {
            return true;
        }

        return false;
    }
}
```



vii. Write a java program to count number of alphabets, digits, special symbols, blank spaces and words from the given sentence. Also count number of vowels and consonants



```
39
40              if (!input.isEmpty()) {
41                  wordCount++;
42              }
43
44              System.out.println("Alphabets: " + alphabetCount);
45              System.out.println("Digits: " + digitCount);
46              System.out.println("Special Symbols: " + specialSymbolCount);
47              System.out.println("Blank Spaces: " + spaceCount);
48              System.out.println("Words: " + wordCount);
49              System.out.println("Vowels: " + vowelCount);
50              System.out.println("Consonants: " + consonantCount);
51
52              scanner.close();
53          }
54
            1 usage
55          private static boolean isVowel(char ch) {
56              return "aeiou".indexOf(ch) != -1;
57          }
58      }
```

Run Java9

```
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
Enter a sentence: Hi this is a sentence
Alphabets: 17
Digits: 0
Special Symbols: 0
Blank Spaces: 4
Words: 5
Vowels: 7
Consonants: 10

Process finished with exit code 0
```