Day 3

Task 1: Arrays - Declaration, Initialization, and Usage

Create a program that declares an array of integers, initializes it with consecutive numbers, and prints the array in reverse order.

```
import java.util.Scanner;


public class ReverseArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();


        int[] array = new int[size];
        for (int i = 0; i < size; i++) {
            array[i] = i + 1;
        }
        System.out.print("Original array: ");
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
        System.out.print("Reversed array: ");
        for (int i = array.length - 1; i >= 0; i--) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
```

```
    }
}
```

Here's what happens in this program:

- It asks the user for the size of the array.
- It initializes the array with consecutive numbers starting from 1.
- It prints the original array.
- It prints the array in reverse order.
- When you run this program, it will prompt you to enter the size of the array. After you input the size, it will output the original and reversed arrays.

Example output:

Enter the size of the array: 5

Original array: 1 2 3 4 5

Reversed array: 5 4 3 2 1

Task 2: List interface

Implement a method that takes a List as an argument and removes every second element from the list, then prints the resulting list.

Java code to remove every second element from a list and print the resulting list

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

public class RemoveSecondElement {

    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        for (int i = 1; i <= 10; i++) {
            list.add(i);
```

```
        }
        System.out.println("Original list: " + list);

        removeEverySecondElement(list);
        System.out.println("List after removing every second element: " + list);
    }

    public static void removeEverySecondElement(List<Integer> list) {
        for (int i = 1; i < list.size(); i++) {
            list.remove(i);
        }
    }
}
```

Explanation:

Method Definition:

- public static void removeEverySecondElement(List<Integer> list):
- This method is public and static, meaning it can be called without creating an instance of the class.
- It takes a List<Integer> as an argument, which is a list of integers.

For Loop Initialization:

- for (int i = 1; i < list.size(); i++):
- The loop starts with i = 1, which is the second element in the list (indexing starts at 0).
- The loop runs as long as i is less than the size of the list (i < list.size()).
- The loop variable i increments by 1 after each iteration (i++).

Removing Elements:

list.remove(i):

- On each iteration, the element at the current index i is removed from the list.
- When an element is removed, the size of the list decreases by 1.

- This reduction in size means that the indices of subsequent elements shift left by one position.

Output:

Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

List after removing every second element: [1, 3, 5, 7, 9]

Task 3: Set interface

Write a program that reads words from a String variable into a Set and prints out the number of unique words, demonstrating the unique property of sets.

```
import java.util.HashSet;
import java.util.Set;

public class UniqueWords {
    public static void main(String[] args) {
        String text = "Java is a programming language. Python is also a programming language.";

        String[] words = text.split("\\s+");

        Set<String> uniqueWords = new HashSet<>();

        for (String word : words) {
            uniqueWords.add(word);
        }

        System.out.println("Number of unique words: " + uniqueWords.size());

        System.out.println("Unique words: " + uniqueWords);
    }
}
```

Explanation:

- Import Statements: The HashSet and Set classes from the java.util package are imported to use sets in the program.
- String Variable: The text variable contains the string of words.
- Splitting the String: The split("\\s+") method is used to split the string into an array of words. The regular expression \\s+ matches one or more whitespace characters.
- Creating the Set: A HashSet is created to store the unique words.
- Adding Words to the Set: A for loop iterates over the array of words and adds each word to the set. Because sets automatically remove duplicates, only unique words will be stored.
- Printing the Results: The number of unique words is printed using uniqueWords.size(), and the unique words themselves are printed by simply printing the set.

Output:

Number of unique words: 9

Unique words: [a, language., Java, also, programming, is, Python, language, language;

Task 4: Map interface

Create a Java class that uses a Map to store the frequency of each word that appears in a given string.

```
import java.util.HashMap;
import java.util.Map;

public class WordFrequency {

    public static void main(String[] args) {
        String text = "Java is a programming language. Python is also a programming language.";

        String[] words = text.replaceAll("[^a-zA-Z ]", "").toLowerCase().split("\\s+");

        Map<String, Integer> wordFrequency = new HashMap<>();
```

```
        for (String word : words) {
            if (word.length() > 0) {
                wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);
            }
        }
        System.out.println("Word frequencies:");
        for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

String Variable: The text variable contains the string of words.

Removing Punctuation and Splitting:

- replaceAll("[^a-zA-Z]", "") removes all characters except letters and spaces from the string.
- .toLowerCase() converts the string to lowercase to ensure case insensitivity.
- .split("\\s+") splits the string into an array of words using one or more whitespace characters as the delimiter.
- Creating the Map: A HashMap named wordFrequency is created to store word frequencies. The key is the word itself, and the value is the count of occurrences.

Counting Frequencies:

- The for loop iterates through each word in the words array.
- wordFrequency.getOrDefault(word, 0) + 1 retrieves the current count of the word. If the word is not present, getOrDefault returns 0, and 1 is added to it to initialize the count.
- wordFrequency.put(word, ...) updates the count of the word in the map.

Output:

Word frequencies:

a: 2

language: 2

is: 2

programming: 2

java: 1

python: 1

also: 1

Task 5: Iterators and Comparators

Write a custom Comparator to sort a list of Employee objects by their salary and then by name if the salary is the same.

```
import java.util.*;

class  Employee {
   private String name;
   private double salary;

   public Employee(String name, double salary) {
      this.name = name;
      this.salary = salary;
   }

   public String getName() {
      return name;
   }

   public double getSalary() {
      return salary;
   }

   @Override
```

```
    public String toString() {
        return "Employee{" +
                "name='" + name + '\'' +
                ", salary=" + salary +
                '}';
    }
}

public class EmployeeSort {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employees = new ArrayList<>();

        System.out.println("Enter number of employees:");
        int numEmployees = scanner.nextInt();
        scanner.nextLine(); // consume newline

        for (int i = 0; i < numEmployees; i++) {
            System.out.println("Enter name for employee " + (i + 1) + ":");
            String name = scanner.nextLine();

            System.out.println("Enter salary for employee " + (i + 1) + ":");
            double salary = scanner.nextDouble();
            scanner.nextLine(); // consume newline

            employees.add(new Employee(name, salary));
        }
```

```
        Comparator<Employee> comparator = Comparator
                .comparingDouble(Employee::getSalary)
                .thenComparing(Employee::getName);

        Collections.sort(employees, comparator);
        System.out.println("\nSorted Employees:");
        employees.forEach(System.out::println);

        scanner.close();
    }
}
```

Employee Class: Represents an Employee with name and salary attributes.

EmployeeSort Class:

Main Method:

- Uses Scanner to take input from the user.
- Prompts the user to enter the number of employees and details (name and salary) for each employee.
- Creates instances of Employee and adds them to the employees list.
- Defines a custom Comparator using the Comparator.comparingDouble() method. This comparator first compares employees by their salary using Employee::getSalary, and then by their name using Employee::getName.
- Sorts the employees list using Collections.sort() method and the custom comparator.
- Prints the sorted list of employees.

Comparator Explanation:

- Comparator.comparingDouble(Employee::getSalary) creates a comparator that compares Employee objects by their salary in ascending order.
- .thenComparing(Employee::getName) is used to compare Employee objects by their name in ascending order if their salaries are the same.

Output:

Enter number of employees:

4

Enter name for employee 1:

John

Enter salary for employee 1:

60000

Enter name for employee 2:

Alice

Enter salary for employee 2:

75000

Enter name for employee 3:

Bob

Enter salary for employee 3:

50000

Enter name for employee 4:

Eve

Enter salary for employee 4:

60000

Sorted Employees:

Employee{name='Bob', salary=50000.0}

Employee{name='Eve', salary=60000.0}

Employee{name='John', salary=60000.0}

Employee{name='Alice', salary=75000.0}