Java Collections Framework and Generics:

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.

```
package com.wipro.assignment02;
public class ReverseArray
{
	public static void main(String[] args) {
	// Declare and initialize an array with consecutive numbers
	  int[] numbers = new int[10];
	  for (int i = 0; i < numbers.length; i++) {
	     numbers[i] = i+1; // Initializing with consecutive numbers starting from 1
	  }

	  // Print the array in reverse order
	  System.out.println("Array in reverse order:");
	  for (int i = numbers.length - 1; i >= 0; i--) {
	      System.out.print(numbers[i] + " ");
	  }
    }
}
```

Output:

```
Array in reverse order:
10 9 8 7 6 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.

```
package com.wipro.assignment02;
import java.util.ArrayList;
import java.util.List;

public class RemoveEverySecondElement
{

	public static void main(String[] args)
	{

	  // Create and initialize a list with some example elements
	  List<Integer> numbers = new ArrayList<>();
	  for (int i = 1; i <= 10; i++) {
```

```
            numbers.add(i);
        }

        // Print the original list
        System.out.println("Original list: " + numbers);

        // Call the method to remove every second element
        removeEverySecondElement(numbers);

        // Print the resulting list
        System.out.println("List after removing every second element: " + numbers);
    }

    // Method to remove every second element from the list
    public static void removeEverySecondElement(List<Integer> list)
     {
        for (int i = 1; i < list.size(); i++) {
            list.remove(i);
        }
    }
}
```

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.

```
package com.wipro.assignment02;
import java.util.HashSet;
import java.util.Set;
public class UniqueWords
{

    public static void main(String[] args)
{
        // Initialize a string with some words
        String text = "aaa,bbb,ee,aa,aaa,ccc,ee,rrr,rrr,www,dd,cc.";

        // Call the method to count unique words
        int uniqueWordCount = countUniqueWords(text);

        // Print the number of unique words
        System.out.println("Number of unique words: " + uniqueWordCount);
    }

    // Method to count unique words
    public static int countUniqueWords(String text) {
        // Split the string into words based on spaces and punctuation
```

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

        // Use a set to store unique words
        Set<String> uniqueWords = new HashSet<>();

        // Add words to the set
        for (String word : words) {
            if (!word.isEmpty()) {
                uniqueWords.add(word.toLowerCase()); // Convert to lowercase to
ensure uniqueness
            }
        }

        // Print the unique words
        System.out.println("Unique words: " + uniqueWords);

        // Return the number of unique words
        return uniqueWords.size();
    }
}
```

Output:

```
Unique words: [aaa, ee, aa, dd, cc, ccc, bbb, rrr, www]
Number of unique words: 9
```

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.

```
package com.wipro.assignment02;
import java.util.HashMap;
import java.util.Map;
public class WordFrequency
{
    public static void main(String[] args)
    {
        // Initialize a string with some words
        String text = "This is a test. This test is only a test.";

        // Create an instance of the WordFrequency class
        WordFrequency wf = new WordFrequency();

        // Get the word frequency map
        Map<String, Integer> wordFrequencyMap = wf.getWordFrequency(text);

        // Print the word frequency map
        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }

    // Method to get the word frequency map
```

```
    public Map<String, Integer> getWordFrequency(String text)
    {
        String[] words = text.split("\\W+");

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

        for (String word : words) {
            if (!word.isEmpty()) {
                word = word.toLowerCase();
                wordFrequencyMap.put(word,wordFrequencyMap.getOrDefault(word,0) +1);
            }
        }
        return wordFrequencyMap;
    }
}
```

Output:

```
a: 2
test: 3
this: 2
only: 1
is: 2
```

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.

Employee Class:

```
package com.wipro.assignment02;
public 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 + "}";
    }
}
```

EmployeeComparator Class:

```
package com.wipro.assignment02;
import java.util.Comparator;
public class EmployeeComparator implements Comparator<Employee>
{
    @Override
    public int compare(Employee e1, Employee e2) {
        // Compare by salary
        int salaryComparison = Double.compare(e1.getSalary(), e2.getSalary());

        // If salaries are equal, compare by name
        if (salaryComparison == 0) {
            return e1.getName().compareTo(e2.getName());
        }

        return salaryComparison;
    }
}
```

Main Class:

```
package com.wipro.assignment02;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Main
{
    public static void main(String[] args)
 {
        // Create a list of Employee objects
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Vishal", 75000));
        employees.add(new Employee("Om", 50000));
        employees.add(new Employee("Chaitan", 75000));
        employees.add(new Employee("Parl", 50000));
        employees.add(new Employee("Sarthak", 90000));

        // Print the list before sorting
        System.out.println("Before sorting:");
        for (Employee employee : employees) {
```

```
            System.out.println(employee);
        }

        // Sort the list using EmployeeComparator
        Collections.sort(employees, new EmployeeComparator());

        // Print the list after sorting
        System.out.println("\nAfter sorting:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }
}
```

Output:

```
Before sorting:
Employee{name='Vishal', salary=75000.0}
Employee{name='Om', salary=50000.0}
Employee{name='Chaitan', salary=75000.0}
Employee{name='Parl', salary=50000.0}
Employee{name='Sarthak', salary=90000.0}

After sorting:
Employee{name='Om', salary=50000.0}
Employee{name='Parl', salary=50000.0}
Employee{name='Chaitan', salary=75000.0}
Employee{name='Vishal', salary=75000.0}
Employee{name='Sarthak', salary=90000.0}
```