

Day 3: Manisha Assignment

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.

1. Array Declaration and Initialization:

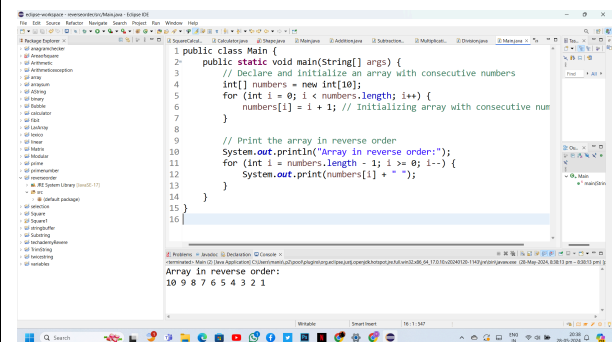
- We declare an array of integers named `numbers` with a size of 10.
- We use a `for` loop to initialize the array with consecutive numbers from 1 to 10.

2. Printing the Array in Reverse Order:

- We print a message to indicate that the array will be printed in reverse order.
- We use another `for` loop that starts from the end of the array and goes to the beginning, printing each element. This loop prints the array elements from the last to the first.

This program initializes an array with consecutive numbers and then prints these numbers in reverse order.

Here Output as follows 👍



The screenshot shows an IDE with a Java file named 'Main.java'. The code declares an array of integers 'numbers' of size 10, initializes it with values from 1 to 10 using a for loop, and then prints the array in reverse order using another for loop. The output window shows the text 'Array in reverse order:' followed by the numbers '10 9 8 7 6 5 4 3 2 1'.

```
1 public class Main {
2     public static void main(String[] args) {
3         // Declare and initialize an array with consecutive numbers
4         int[] numbers = new int[10];
5         for (int i = 0; i < numbers.length; i++) {
6             numbers[i] = i + 1; // Initializing array with consecutive num
7         }
8
9         // Print the array in reverse order
10        System.out.println("Array in reverse order:");
11        for (int i = numbers.length - 1; i >= 0; i--) {
12            System.out.print(numbers[i] + " ");
13        }
14    }
15 }
16
```

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.

1. Initializing the List:

- We create an `ArrayList` of integers named `numbers`.
- We use a `for` loop to add numbers 1 through 10 to the list.

2. Method to Remove Every Second Element:

- The method `removeEverySecondElement(List<Integer> list)` iterates through the list and removes every second element.
- We start the loop with `i = 1` (the second element) and increment `i` by 1 each time, effectively skipping the next element due to the removal.

3. Printing the Resulting List:

- After calling the method to remove every second element, we print the modified list to show the result.

Running the Program in Eclipse

1. Create the Project:

- Follow the steps mentioned above to create the project `RemoveEverySecondElement` in Eclipse.

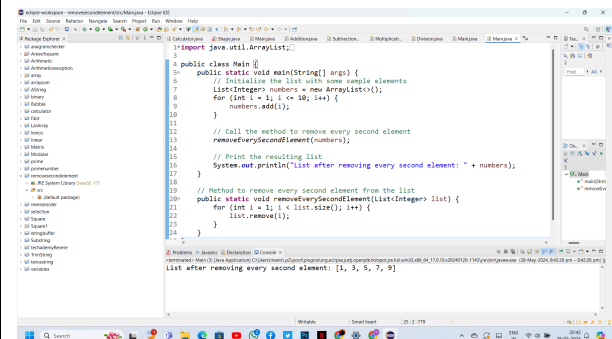
2. Add the Main Class:

- Follow the steps to create the `Main` class and copy the provided code into it.

3. Run the Main Class:

- Right-click on the `Main` class, select `Run As > Java Application`.

Here output as follows 👍



The screenshot shows the Eclipse IDE with a Java project named 'RemoveEverySecondElement'. The 'Main' class is open, displaying the following code:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        // Initialize the list with some sample elements
        List<Integer> numbers = new ArrayList<>();
        for (int i = 1; i <= 10; i++) {
            numbers.add(i);
        }

        // 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);
        }
    }
}
```

The output window at the bottom shows the result of running the program:

```
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.

1. Import Statements:

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

These are necessary to use the `HashSet` and `Set` classes.

2. Main Method:

- `public static void main(String[] args) {`
... `}`

This is the entry point of the program.

3. Input String:

- `String input = "This is a test. This test is only a test.";`

This is the string from which we want to extract unique words.

4. Split the Input String:

- `String[] words = input.toLowerCase().split("\\W+");`

The input string is converted to lower case to ensure case-insensitive comparison, and then it is split into words using a regular expression that matches non-word characters (`\\W+`).

5. Create a Set for Unique Words:

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

A `HashSet` is used to store unique words.

6. Add Words to the Set:

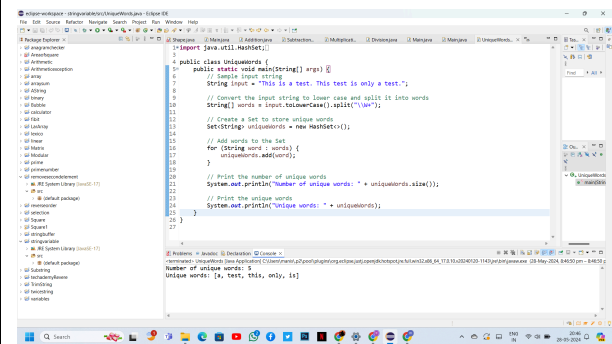
- `for (String word : words) {`
`uniqueWords.add(word);` `}`

This loop iterates over the array of words and adds each word to the `HashSet`.

7. Print the Number of Unique Words:

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

Here output as follows:



The screenshot shows a Java IDE with a code editor and a console window. The code in the editor is as follows:

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

public class UniqueWords {
    public static void main(String[] args) {
        // Sample input string
        String input = "This is a test. This test is only a test.";

        // Convert the input string to lower case and split it into words
        String[] words = input.toLowerCase().split("\\W+");

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

        // Add words to the Set
        for (String word : words) {
            uniqueWords.add(word);
        }

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

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

The console window shows the output of the program:

```
Number of unique words: 5
Unique words: [a, test, this, only, is]
```

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

1. Import Statements:

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

These imports are necessary to use the `HashMap` and `Map` classes.

## 2. Main Method:

- ```
- public static void main(String[] args) {
... }
```

This is the entry point of the program.

3. Input String:

- ```
- `String input = "This is a test. This test
is only a test.";
```

This is the string from which we want to calculate word frequencies.

#### 4. Create a Map for Word Frequencies:

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

A `HashMap` is used to store each word and its frequency count.

5. Split the Input String:

- ```
String[] words = input.toLowerCase().split("\\W+");
```

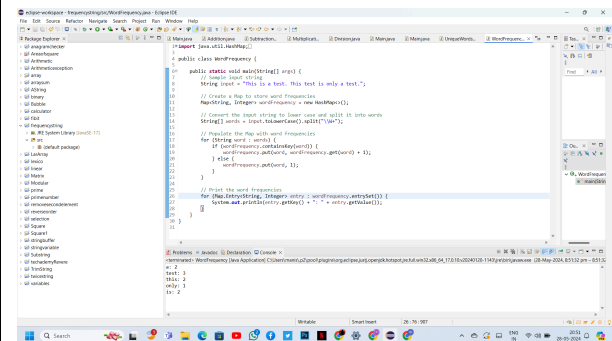
The input string is converted to lower case to ensure case-insensitive comparison, and then it is split into words using a regular expression that matches non-word characters (`'\\W+'`).

## 6. Populate the Map with Word Frequencies:

- The `for` loop iterates over the array of words. For each word:
  - If the word is already in the map (`if (wordFrequency.containsKey(word))`), increment its count by 1.
  - If the word is not in the map, add it with a count of 1.

## 7. Print the Word Frequency

Here output as Follows:



## 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.

### 1. Create a New Java Project:

- Open Eclipse.
- Go to `File` > `New` > `Java Project`.
- Enter the project name (e.g., `EmployeeSorterProject`) and click `Finish`.

### 2. Create a New Java Class:

- Right-click on the `src` folder of your project in the Project Explorer.
- Select `New` > `Class`.
- Enter the class name (e.g., `EmployeeSorter`) and click `Finish`.

### 3. Copy and Paste the Code:

- Copy the complete code block provided above.
- Paste it into the `EmployeeSorter` class file in Eclipse.

### 4. Run the Program:

- Right-click on the `EmployeeSorter` class in the Project Explorer.
- Select `Run As` > `Java Application`.
- The output will be displayed in the Console view at the bottom of the Eclipse window.

### Here Output as follows

```
44 public class EmployeeSorter {
45 public static void main(String[] args) {
46 // Create a list of employees
47 List<Employee> employees = new ArrayList<>();
48 employees.add(new Employee("John Doe", 50000));
49 employees.add(new Employee("Jane Smith", 60000));
50 employees.add(new Employee("Alice Johnson", 50000));
51 employees.add(new Employee("Bob Williams", 60000));
52
53 // Sort the list using the custom comparator
54 Collections.sort(employees, new EmployeeComparator());
55
56 // Print the sorted list
57 for (Employee employee : employees) {
58 System.out.println(employee);
59 }
60 }
61 }
62 }
```

```
5 // Import java.util.ArrayList;
6
7 class Employee {
8 private String name;
9 private double salary;
10
11 // Constructor
12 public Employee(String name, double salary) {
13 this.name = name;
14 this.salary = salary;
15 }
16
17 // Getters
18 public String getName() {
19 return name;
20 }
21
22 public double getSalary() {
23 return salary;
24 }
25 }
```

```
31 // toString method for easy printing
32 @Override
33 public String toString() {
34 return "Employee{name=" + name + ", salary=" + salary + "}";
35 }
36
37 class EmployeeComparator implements Comparator<Employee> {
38 @Override
39 public int compare(Employee e1, Employee e2) {
40 // First compare by salary
41 int salaryComparison = Double.compare(e1.getSalary(), e2.getSalary());
42 if (salaryComparison != 0) {
43 return salaryComparison;
44 }
45 // If salaries are the same, compare by name
46 return e1.getName().compareTo(e2.getName());
47 }
48 }
```

```
Employee{name="Alice Johnson", salary=50000.0}
Employee{name="John Doe", salary=50000.0}
Employee{name="Bob Williams", salary=60000.0}
Employee{name="Jane Smith", salary=60000.0}
```