

Day:3

Task 1: Arrays- declaration, Initialization and usage;

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

```
public class ReverseArray {  
    public static void main(String[] args) {  
        // Declare and initialize an array of integers  
        int[] numbers = new int[10];  
  
        // Initialize the array with consecutive numbers  
        for (int i = 0; i < numbers.length; i++) {  
            numbers[i] = i + 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] + " ");  
        }  
    }  
}
```

1. Array Declaration and Initialization:

- An array of integers named **numbers** is declared and initialized with a size of 10.

2. Initializing with Consecutive Numbers:

- A **for** loop is used to iterate through each element of the array.
- The value of each element is set to its index plus 1, resulting in consecutive numbers starting from 1.

3. Printing in Reverse Order:

- Another **for** loop is used to iterate through the array in reverse order, starting from the last index (**numbers.length - 1**) down to 0.
- Each element of the array is printed, resulting in the array being displayed in reverse order.

Task2: List interface:

implement a method that takes a list as an arguments and remove every second element from the list,then print the resulting list,

Below are screenshot that declare array, and it remove every second elements;

The screenshot shows an IDE window titled "RemoveEverySec...". The code is in Java and defines a class "RemoveEverySecondElement" with a "main" method. The "main" method creates an "ArrayList" named "numbers" and adds elements 1 through 6. It then prints the original list, calls the "removeEverySecondElement" method, and prints the resulting list. The "removeEverySecondElement" method is a static method that takes a "List<Integer>" as input and removes every second element.

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class RemoveEverySecondElement {
5     public static void main(String[] args) {
6         // Create a list of integers
7         List<Integer> numbers = new ArrayList<>();
8         numbers.add(1);
9         numbers.add(2);
10        numbers.add(3);
11        numbers.add(4);
12        numbers.add(5);
13        numbers.add(6);
14
15        // Print the original list
16        System.out.println("Original List: " + numbers);
17
18        // Remove every second element from the list
19        removeEverySecondElement(numbers);
20
21        // Print the resulting list
22        System.out.println("Resulting List: " + numbers);
23    }
24
25    // Method to remove every second element from the list
26    public static void removeEverySecondElement(List<Integer> list) {
27        for (int i = list.size() - 1; i >= 0; i -= 2) {
28            list.remove(i);
29        }
30    }
31 }
```

Language: Java

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the output of the Java program. The "Original List" is [1, 2, 3, 4, 5, 6] and the "Resulting List" is [1, 3, 5]. The program finished with exit code 0.

```
Original List: [1, 2, 3, 4, 5, 6]
Resulting List: [1, 3, 5]

...Program finished with exit code 0
Press ENTER to exit console.
```

Activate Windows
Go to Settings to activate Windows.

Task 3: set interface:

Write a program that reads words from string variable into a Set and print out the number of unique words, demonstrating the unique property of sets:

Below is a Java 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 UniqueWordsCounter {
    public static void main(String[] args) {
        // Sample string variable containing words
        String text = "apple banana orange apple banana grape orange mango";

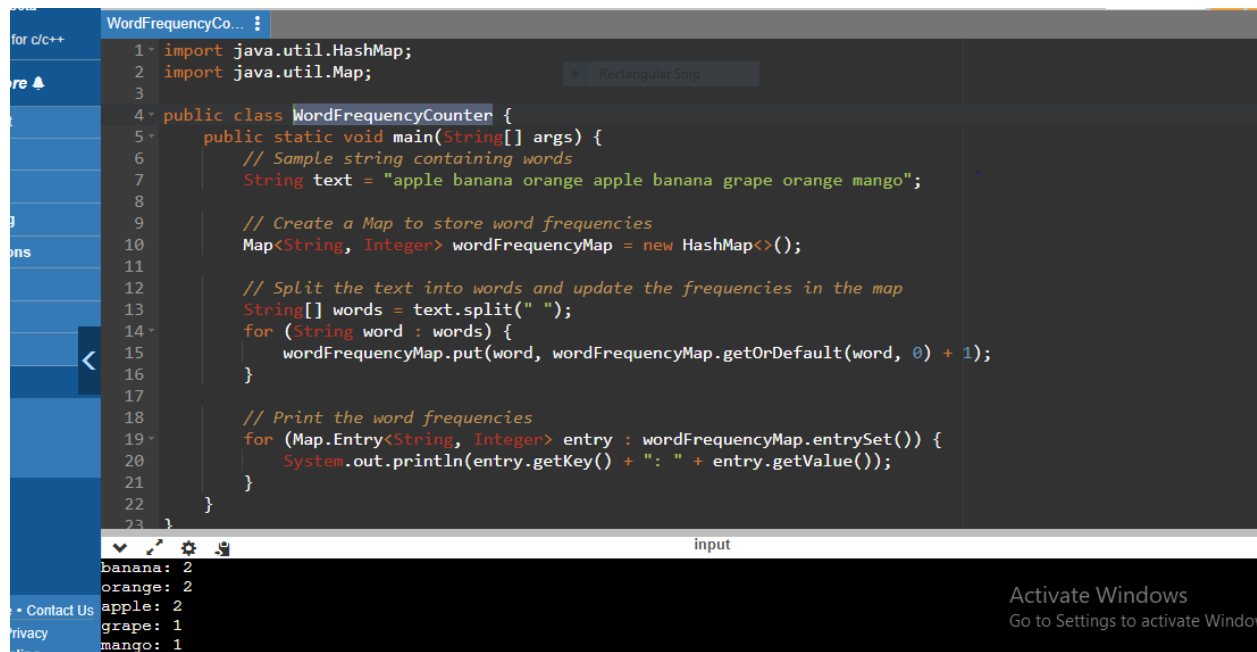
        // Create a Set to store unique words
        Set<String> uniqueWords = new HashSet<>();

        // Split the text into words and add them to the Set
        String[] words = text.split(" ");
        for (String word : words) {
            uniqueWords.add(word);
        }
    }
}
```

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

4 Map Interface:

Create a java class that uses a map to store the frequency of each word that appears in the given string;

A screenshot of a Java IDE window titled 'WordFrequencyCo...'. The code defines a class 'WordFrequencyCounter' with a 'main' method. The main method takes a string of words, splits it into an array, and uses a 'HashMap' to count the frequency of each word. The output is printed to the console. The console shows the following output: banana: 2, orange: 2, apple: 2, grape: 1, mango: 1. An 'Activate Windows' watermark is visible in the bottom right corner.

```
1 import java.util.HashMap;
2 import java.util.Map;
3
4 public class WordFrequencyCounter {
5     public static void main(String[] args) {
6         // Sample string containing words
7         String text = "apple banana orange apple banana grape orange mango";
8
9         // Create a Map to store word frequencies
10        Map<String, Integer> wordFrequencyMap = new HashMap<>();
11
12        // Split the text into words and update the frequencies in the map
13        String[] words = text.split(" ");
14        for (String word : words) {
15            wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1);
16        }
17
18        // Print the word frequencies
19        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet()) {
20            System.out.println(entry.getKey() + ": " + entry.getValue());
21        }
22    }
23 }
```

banana: 2
orange: 2
apple: 2
grape: 1
mango: 1

1. Sample String Variable:

- A sample string variable named text is initialized with words separated by spaces.

2. Map Creation:

- A HashMap named wordFrequencyMap is created to store word frequencies.
- The keys of the map will be words, and the values will be their corresponding frequencies.

3. Splitting and Updating Frequencies:

- The text string is split into individual words using the `split()` method, with space as the delimiter.
- A loop iterates over each word.
- For each word, its frequency in the map is updated:
 - If the word is already present in the map, its frequency is incremented by 1.
 - If the word is not present, it is added to the map with a frequency of 1.

4. Printing the Word Frequencies:

- Finally, the program prints the word frequencies stored in the map.