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;

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

public class RemoveEverySecondElement {
    public static void main(String[] args) {
        // Create a list of integers
        List<Integer> numbers = new ArrayList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);
        numbers.add(6);

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

        // Remove every second element from the list
        removeEverySecondElement(numbers);

        // Print the resulting list
        System.out.println("Resulting List: " + numbers);
    }

    // Method to remove every second element from the list
    public static void removeEverySecondElement(List<Integer> list) {
        for (int i = list.size() - 1; i >= 0; i -= 2) {
            list.remove(i);
        }
    }
}
```

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


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

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:

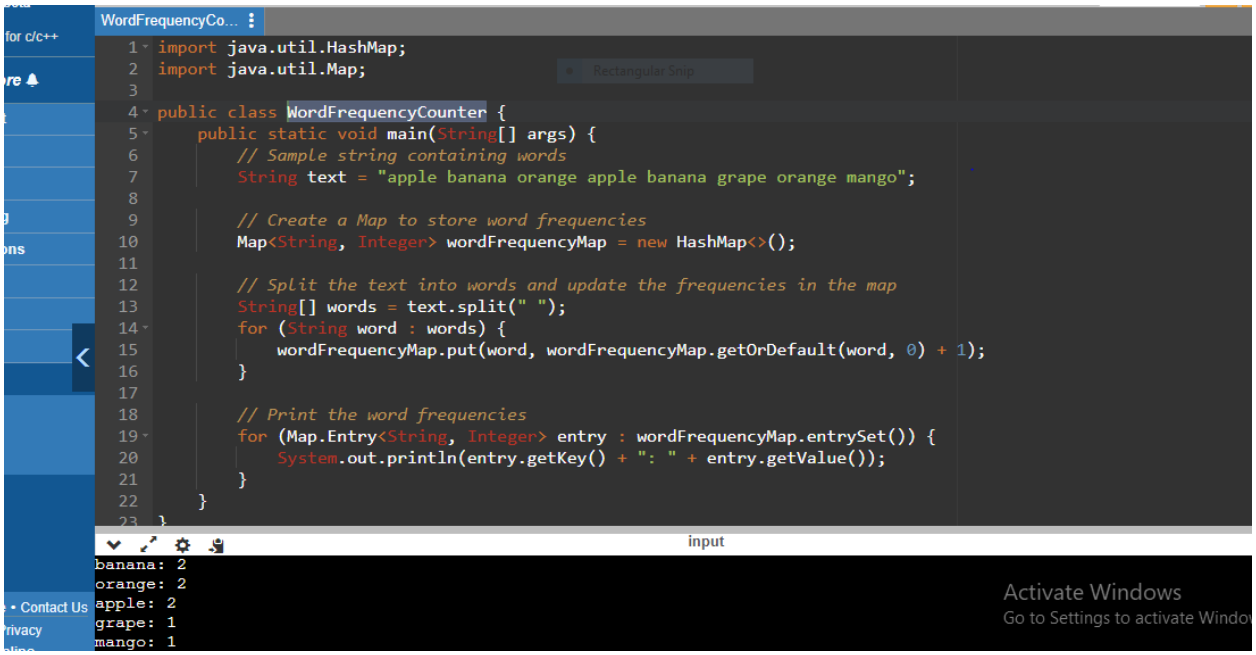
Below is a Java program that reads words from a string variable into a Set and prints out the number of unique words, demonstrating the unique property of sets:

```
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;



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

public class WordFrequencyCounter {
    public static void main(String[] args) {
        // Sample string containing words
        String text = "apple banana orange apple banana grape orange mango";

        // Create a Map to store word frequencies
        Map<String, Integer> wordFrequencyMap = new HashMap<>();

        // Split the text into words and update the frequencies in the map
        String[] words = text.split(" ");
        for (String word : words) {
            wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1);
        }

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

```
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.**