DATE:27.12.23 **JAVA- QUIZ-1**

1.Given the list of array return array in which each element is the product of other element except
element (try to do it without division operation)in java

```
import java.util.Scanner;
public class Demo{
    public int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];
        int prefixProduct = 1;
        for (int i = 0; i < n; i++) {
            result[i] = prefixProduct;
            prefixProduct *= nums[i];
        }


        int suffixProduct = 1;
        for (int i = n - 1; i >= 0; i--) {
            result[i] *= suffixProduct;
            suffixProduct *= nums[i];
        }

        return result;
    }

    public static void main(String[] args) {
        ProductExceptSelf solution = new ProductExceptSelf();
        int[] nums = {1, 2, 3, 4};
        int[] result = solution.productExceptSelf(nums);

        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}}
```

2.Given an array list return all possible permutations

Input: nums = [1,4,3]

Output: [[1,4,3],[1,3,4],[4,1,3],[4,3,1],[3,1,4],[3,4,1]]

```
import java.util.Scanner;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;
import java.util.Scanner;

public class Permutations {

    public List<List<Integer>> permute(int[] nums) {

        List<List<Integer>> result = new ArrayList<>();

        generatePermutations(nums, 0, result);

        return result;

    }

    private void generate(int[] nums, int index, List<List<Integer>> result) {

       if (index == nums.length - 1) {

            result.add(new ArrayList<>(Arrays.asList(Arrays.stream(nums).boxed().toArray(Integer[]::new))));

        }

      for (int i = index; i < nums.length; i++) {

            swap(nums, index, i);

            generatePermutations(nums, index + 1, result);

            swap(nums, index, i);       }

    }

    private void swap(int[] nums, int i, int j) {

        int temp = nums[i];

        nums[i] = nums[j];

        nums[j] = temp;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the elements of the array (space-separated): ");

        String[] input = scanner.nextLine().split(" ");

        int[] nums = Arrays.stream(input).mapToInt(Integer::parseInt).toArray();
        Permutations solution = new Permutations();

        List<List<Integer>> result = solution.permute(nums);

        System.out.println("All possible permutations: ");

        for (List<Integer> permutation : result) {

            System.out.println(permutation);

        }

    }
}
```

3.Return all the clubbed words
Input:words=["mat","mate","matbellmates","bell","bellmatesbell","butterribbon","butter","ribbon"]
Output: ["matbellmates","bellmatesbell","butterribbon"]

CODE:

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

public class Demo {
    public List<String> findClubbedWords(String[] words) {
        List<String> clubbedWords = new ArrayList<>();

        for (String word : words) {
            if (isClubbedWord(word, new ArrayList<>(Arrays.asList(words)))) {
                clubbedWords.add(word);
            }
        }

        return clubbedWords;
    }

    private boolean isClubbedWord(String word, List<String> wordList) {
        wordList.remove(word);

        for (int i = 1; i <= word.length(); i++) {
            String prefix = word.substring(0, i);
            String suffix = word.substring(i);

            if (wordList.contains(prefix) && (wordList.contains(suffix) || isClubbedWord(suffix, wordList))) {
                return true;
            }
        }

        wordList.add(word); // Backtrack
        return false;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the words (comma-separated): ");

        String[] input = scanner.nextLine().split(",\\s*");
        String[] words = Arrays.copyOf(input, input.length);

        ClubbedWords solution = new ClubbedWords();
        List<String> clubbedWords = solution.findClubbedWords(words);

        System.out.println("Clubbed words: " + clubbedWords);
    }
}
```