

25 Java Coding Questions based on Set, Map etc

- [1. Find First Non-Repeating Character in a String using HashMap](#)
 - [2. Find Duplicate Elements in an Array using HashSet](#)
 - [3. Find Intersection of Two Arrays using Set](#)
 - [4. Find Word Frequency in a Sentence using HashMap](#)
 - [5. Check if Two Strings are Anagrams using HashMap](#)
 - [6. Find the First Repeating Character in a String using HashSet](#)
 - [7. Remove Duplicates from a List using HashSet](#)
 - [8. Find Two Numbers that Add Up to Target \(Two Sum\) using HashMap](#)
 - [9. Check if Array Contains Duplicate using HashSet](#)
 - [10. Group Anagrams from an Array of Strings using HashMap](#)
 - [11. Find Occurrence of Each Word in a String using HashMap](#)
 - [12. Count Occurrences of Each Character in a String](#)
 - [13. Remove Duplicates from an Array using HashSet](#)
 - [14. Find the First Non-Repeating Character in a String](#)
 - [15. Find Intersection of Two Arrays](#)
 - [16. Convert CamelCase to Snake_Case](#)
 - [17. Find Two Numbers that Sum to a Target \(Two Sum Problem\)](#)
 - [18. Check if Two Strings are Anagrams](#)
 - [19. Check if a String is a Palindrome](#)
 - [20. Remove Consecutive Duplicate Words from a Sentence](#)
 - [21. Find the First Repeating Character in a String](#)
 - [22. Find the Missing Number in an Array](#)
 - [23. Find the Longest Word in a Sentence](#)
 - [24. Find Duplicate Elements in an Array](#)
 - [25. Find the Most Frequent Element in an Array](#)
-

1. Find First Non-Repeating Character in a String using HashMap

```
import java.util.LinkedHashMap;
import java.util.Map;

public class FirstUniqueChar {
    public static char firstUniqueChar(String str) {
        Map<Character, Integer> map = new LinkedHashMap<>();
        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            if (entry.getValue() == 1) return entry.getKey();
        }
        return '_';
    }

    public static void main(String[] args) {
        System.out.println(firstUniqueChar("swiss")); // Output: 'w'
    }
}
```

2. Find Duplicate Elements in an Array using HashSet

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

public class FindDuplicates {
    public static void findDuplicates(int[] arr) {
        Set<Integer> seen = new HashSet<>();
        for (int num : arr) {
            if (!seen.add(num)) {
                System.out.println("Duplicate: " + num);
            }
        }
    }

    public static void main(String[] args) {
        findDuplicates(new int[]{1, 2, 3, 4, 2, 5, 6, 3});
    }
}
```

3. Find Intersection of Two Arrays using Set

```
1  import java.util.Arrays;
2  import java.util.HashSet;
3  import java.util.Set;
4
5  public class ArrayIntersection {
6      public static int[] intersection(int[] arr1, int[] arr2) {
7          Set<Integer> set1 = new HashSet<>();
8          Set<Integer> result = new HashSet<>();
9          for (int num : arr1) set1.add(num);
10         for (int num : arr2) {
11             if (set1.contains(num)) result.add(num);
12         }
13         return result.stream().mapToInt(Integer::intValue).toArray();
14     }
15
16     public static void main(String[] args) {
17         System.out.println(Arrays.toString(intersection(new int[]{1, 2, 3, 4}, new int[]{3, 4, 5, 6})));
18     }
19 }
```

4. Find Word Frequency in a Sentence using HashMap

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

public class WordFrequency {
    public static void countWords(String sentence) {
        Map<String, Integer> map = new HashMap<>();
        String[] words = sentence.toLowerCase().split("\\s+");
        for (String word : words) {
            map.put(word, map.getOrDefault(word, 0) + 1);
        }
        System.out.println(map);
    }

    public static void main(String[] args) {
        countWords("This is a test and this is only a test");
    }
}
```

5. Check if Two Strings are Anagrams using HashMap

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

public class AnagramCheck {
    public static boolean isAnagram(String s1, String s2) {
        if (s1.length() != s2.length()) return false;
        Map<Character, Integer> map = new HashMap<>();
        for (char ch : s1.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for (char ch : s2.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) - 1);
        }
        return map.values().stream().allMatch(count -> count == 0);
    }

    public static void main(String[] args) {
        System.out.println(isAnagram("listen", "silent")); // true
        System.out.println(isAnagram("java", "python")); // false
    }
}
```

6. Find the First Repeating Character in a String using HashSet

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

public class FirstRepeatingChar {
    public static char firstRepeatingChar(String str) {
        Set<Character> set = new HashSet<>();
        for (char ch : str.toCharArray()) {
            if (!set.add(ch)) return ch;
        }
        return '_';
    }

    public static void main(String[] args) {
        System.out.println(firstRepeatingChar("abcdefa")); // Output: 'a'
    }
}
```

7. Remove Duplicates from a List using HashSet

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

public class RemoveDuplicates {
    public static List<Integer> removeDuplicates(List<Integer> list) {
        Set<Integer> set = new HashSet<>(list);
        return new ArrayList<>(set);
    }

    public static void main(String[] args) {
        System.out.println(removeDuplicates(List.of(1, 2, 3, 2, 4, 1, 5)));
    }
}
```

8. Find Two Numbers that Add Up to Target (Two Sum) using HashMap

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

public class TwoSum {
    public static int[] findTwoSum(int[] arr, int target) {
        Map<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < arr.length; i++) {
            int complement = target - arr[i];
            if (map.containsKey(complement)) {
                return new int[]{map.get(complement), i};
            }
            map.put(arr[i], i);
        }
        return new int[]{-1, -1};
    }

    public static void main(String[] args) {
        int[] result = findTwoSum(new int[]{2, 7, 11, 15}, 9);
        System.out.println(result[0] + ", " + result[1]); // Output: 0, 1
    }
}
```

9. Check if Array Contains Duplicate using HashSet

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

public class ContainsDuplicate {
    public static boolean containsDuplicate(int[] arr) {
        Set<Integer> set = new HashSet<>();
        for (int num : arr) {
            if (!set.add(num)) return true;
        }
        return false;
    }

    public static void main(String[] args) {
        System.out.println(containsDuplicate(new int[]{1, 2, 3, 4})); // false
        System.out.println(containsDuplicate(new int[]{1, 2, 3, 3})); // true
    }
}
```


10. Group Anagrams from an Array of Strings using HashMap

```
import java.util.*;

public class GroupAnagrams {
    public static List<List<String>> groupAnagrams(String[] words) {
        Map<String, List<String>> map = new HashMap<>();
        for (String word : words) {
            char[] chars = word.toCharArray();
            Arrays.sort(chars);
            String sorted = new String(chars);
            map.computeIfAbsent(sorted, k -> new ArrayList<>()).add(word);
        }
        return new ArrayList<>(map.values());
    }

    public static void main(String[] args) {
        String[] words = {"cat", "dog", "tac", "god", "act"};
        System.out.println(groupAnagrams(words));
    }
}
```

11. Find Occurrence of Each Word in a String using HashMap

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

public class WordOccurrence {
    public static void countWordFrequency(String str) {
        String[] words = str.toLowerCase().split("\\s+");
        Map<String, Integer> map = new HashMap<>();
        for (String word : words) {
            map.put(word, map.getOrDefault(word, 0) + 1);
        }
        System.out.println(map);
    }

    public static void main(String[] args) {
        countWordFrequency("This is a test and this is only a test");
    }
}
```

📌 **Explanation:** Splits the string into words and counts occurrences using `HashMap`.

12. Count Occurrences of Each Character in a String

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

public class CharFrequency {
    public static void countCharFrequency(String str) {
        Map<Character, Integer> map = new HashMap<>();
        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        System.out.println(map);
    }

    public static void main(String[] args) {
        countCharFrequency("hello world");
    }
}
```

📌 **Explanation:** Iterates through each character and counts its frequency using a `HashMap`.

13. Remove Duplicates from an Array using HashSet

```
1  import java.util.Arrays;
2  import java.util.HashSet;
3  import java.util.Set;
4
5  public class RemoveDuplicates {
6      public static int[] removeDuplicates(int[] arr) {
7          Set<Integer> set = new HashSet<>();
8          for (int num : arr) set.add(num);
9          return set.stream().mapToInt(Integer::intValue).toArray();
10     }
11
12     public static void main(String[] args) {
13         System.out.println(Arrays.toString(removeDuplicates(new int[]{1, 2, 3, 2, 4, 1, 5})));
14     }
15 }
```


📌 **Explanation:** Uses a `HashSet` to remove duplicate elements and return unique values.

14. Find the First Non-Repeating Character in a String

```
import java.util.LinkedHashMap;
import java.util.Map;

public class FirstNonRepeatingChar {
    public static char firstUniqueChar(String str) {
        Map<Character, Integer> map = new LinkedHashMap<>();
        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            if (entry.getValue() == 1) return entry.getKey();
        }
        return '_';
    }


    public static void main(String[] args) {
        System.out.println(firstUniqueChar("swiss")); // Output: 'w'
    }
}
```

 **Explanation:** Uses `LinkedHashMap` to maintain insertion order and find the first unique character.

15. Find Intersection of Two Arrays

```
× Array Intersection

1  import java.util.Arrays;
2  import java.util.HashSet;
3  import java.util.Set;
4
5  ✓ public class ArrayIntersection {
6  ✓      public static int[] intersection(int[] arr1, int[] arr2) {
7          Set<Integer> set1 = new HashSet<>();
8          Set<Integer> result = new HashSet<>();
9          for (int num : arr1) set1.add(num);
10         ✓ for (int num : arr2) {
11             if (set1.contains(num)) result.add(num);
12         }
13         return result.stream().mapToInt(Integer::intValue).toArray();
14     }
15
16     ✓ public static void main(String[] args) {
17         System.out.println(Arrays.toString(intersection(new int[]{1, 2, 3, 4}, new int[]{3, 4, 5, 6})));
18     }
19 }
```

 **Explanation:** Uses `HashSet` to find common elements between two arrays.

16. Convert CamelCase to Snake_Case

```
public class CamelToSnake {  
    public static String convertToSnakeCase(String str) {  
        return str.replaceAll("([a-z])([A-Z])", "$1_$2").toLowerCase();  
    }  
  
    public static void main(String[] args) {  
        System.out.println(convertToSnakeCase("camelCaseExample"));  
    }  
}
```

 **Explanation:** Uses regex to insert `_` between lowercase and uppercase characters.

17. Find Two Numbers that Sum to a Target (Two Sum Problem)

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

public class TwoSum {
    public static int[] findTwoSum(int[] arr, int target) {
        Map<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < arr.length; i++) {
            int complement = target - arr[i];
            if (map.containsKey(complement)) {
                return new int[]{map.get(complement), i};
            }
            map.put(arr[i], i);
        }
        return new int[]{-1, -1};
    }

    public static void main(String[] args) {
        int[] result = findTwoSum(new int[]{2, 7, 11, 15}, 9);
        System.out.println(result[0] + ", " + result[1]); // Output: 0, 1
    }
}
```

 **Explanation:** Uses `HashMap` to check for complement values.

18. Check if Two Strings are Anagrams

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

public class AnagramCheck {
    public static boolean isAnagram(String s1, String s2) {
        if (s1.length() != s2.length()) return false;
        Map<Character, Integer> map = new HashMap<>();
        for (char ch : s1.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for (char ch : s2.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) - 1);
        }
        return map.values().stream().allMatch(count -> count == 0);
    }

    public static void main(String[] args) {
        System.out.println(isAnagram("listen", "silent")); // true
        System.out.println(isAnagram("java", "python")); // false
    }
}
```

 **Explanation:** Uses a frequency count approach with `HashMap`.

19. Check if a String is a Palindrome

```
public class PalindromeCheck {
    public static boolean isPalindrome(String str) {
        str = str.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
        int left = 0, right = str.length() - 1;
        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) return false;
            left++;
            right--;
        }
        return true;
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("A man, a plan, a canal: Panama")); // true
    }
}
```

📌 **Explanation:** Cleans the string and checks for palindrome.

20. Remove Consecutive Duplicate Words from a Sentence

```
import java.util.Arrays;
import java.util.LinkedHashSet;
import java.util.Set;

public class RemoveDuplicateWords {
    public static String removeDuplicateWords(String sentence) {
        Set<String> set = new LinkedHashSet<>(Arrays.asList(sentence.split("\\s+")));
        return String.join(" ", set);
    }

    public static void main(String[] args) {
        System.out.println(removeDuplicateWords("hello hello world world java java"));
    }
}
```


📌 **Explanation:** Uses `LinkedHashSet` to maintain order and remove duplicates.

21. Find the First Repeating Character in a String

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

public class FirstRepeatingChar {
    public static char findFirstRepeatingChar(String str) {
        Set<Character> seen = new HashSet<>();
        for (char ch : str.toCharArray()) {
            if (seen.contains(ch)) {
                return ch;
            }
            seen.add(ch);
        }
        return '_'; // No repeating character found
    }

    public static void main(String[] args) {
        System.out.println(findFirstRepeatingChar("abca")); // Output: 'a'
        System.out.println(findFirstRepeatingChar("abcdef")); // Output: '_'
    }
}
```


 **Explanation:** Uses `HashSet` to track characters and returns the first repeating one.

22. Find the Missing Number in an Array

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

public class MissingNumber {
    public static int findMissingNumber(int[] arr, int n) {
        Set<Integer> set = new HashSet<>();
        for (int num : arr) {
            set.add(num);
        }
        for (int i = 1; i <= n; i++) {
            if (!set.contains(i)) {
                return i;
            }
        }
        return -1; // No missing number found
    }

    public static void main(String[] args) {
        System.out.println(findMissingNumber(new int[]{1, 2, 3, 5}, 5)); // Output: 4
    }
}
```

 **Explanation:** Uses `HashSet` to check for the missing number in the sequence.

23. Find the Longest Word in a Sentence

```
1 ✓ public class LongestWord {
2 ✓     public static String findLongestWord(String sentence) {
3         String[] words = sentence.split("\\s+");
4         String longest = "";
5         for (String word : words) {
6 ✓             if (word.length() > longest.length()) {
7                 longest = word;
8             }
9         }
10        return longest;
11    }
12
13 ✓    public static void main(String[] args) {
14        System.out.println(findLongestWord("This is a Java programming interview")); // Output: "programming"
15    }
16 }
```

 **Explanation:** Splits the sentence and finds the longest word.


24. Find Duplicate Elements in an Array

```
1  import java.util.HashSet;
2  import java.util.Set;
3
4  public class DuplicateElements {
5      public static Set<Integer> findDuplicates(int[] arr) {
6          Set<Integer> seen = new HashSet<>();
7          Set<Integer> duplicates = new HashSet<>();
8          for (int num : arr) {
9              if (!seen.add(num)) {
10                 duplicates.add(num);
11             }
12         }
13         return duplicates;
14     }
15
16     public static void main(String[] args) {
17         System.out.println(findDuplicates(new int[]{1, 2, 3, 4, 2, 5, 3})); // Output: [2, 3]
18     }
19 }
20
```

 **Explanation:** Uses `HashSet` to track duplicates efficiently.

25. Find the Most Frequent Element in an Array

```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  public class MostFrequentElement {
5      public static int findMostFrequent(int[] arr) {
6          Map<Integer, Integer> map = new HashMap<>();
7          int maxCount = 0, mostFrequent = arr[0];
8          for (int num : arr) {
9              int count = map.getOrDefault(num, 0) + 1;
10             map.put(num, count);
11             if (count > maxCount) {
12                 maxCount = count;
13                 mostFrequent = num;
14             }
15         }
16         return mostFrequent;
17     }
18
19     public static void main(String[] args) {
20         System.out.println(findMostFrequent(new int[]{1, 3, 2, 3, 4, 3, 5, 3})); // Output: 3
21     }
22 }
23
```

 **Explanation:** Uses `HashMap` to count occurrences and find the most frequent element.

Happy Learning! 