1. Remove Duplicates from a String

java
CopyEdit
```java
import java.util.LinkedHashSet;

public class RemoveDuplicates {
    public static void main(String[] args) {
        String str = "programming";
        System.out.println(removeDuplicates(str));
    }

    public static String removeDuplicates(String str) {
        LinkedHashSet<Character> set = new LinkedHashSet<>();
        for (char c : str.toCharArray()) {
            set.add(c);
        }
        StringBuilder sb = new StringBuilder();
        for (char c : set) {
            sb.append(c);
        }
        return sb.toString();
    }
}
```

Output:
```
progamin
```

---

2. Print Duplicate Characters from a String

java
CopyEdit
```java
import java.util.HashMap;

public class PrintDuplicates {
    public static void main(String[] args) {
        String str = "hello world";
        findDuplicates(str);
    }

    public static void findDuplicates(String str) {
        HashMap<Character, Integer> map = new HashMap<>();
        for (char c : str.toCharArray()) {
            if (c != ' ') {
                map.put(c, map.getOrDefault(c, 0) + 1);
            }
        }
        for (char c : map.keySet()) {
            if (map.get(c) > 1) {
                System.out.println(c + " : " + map.get(c));
            }
        }
    }
}
```

Output:

yaml
CopyEdit
```
l : 3
o : 2
```

---

3. Check if "2552" is a Palindrome

java
CopyEdit
```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "2552";
        System.out.println(isPalindrome(str));
    }

    public static boolean isPalindrome(String str) {
        int left = 0, right = str.length() - 1;
        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

Output:

```
true
```

---

4. Count Consonants, Vowels, Special Characters in a String

java
CopyEdit

```
public class CountCharacters {
    public static void main(String[] args) {
        String str = "Hello, World!";
        countCharacters(str);
    }

    public static void countCharacters(String str) {
        int vowels = 0, consonants = 0, specialChars = 0;
        str = str.toLowerCase();

        for (char c : str.toCharArray()) {
            if (c >= 'a' && c <= 'z') {
                if ("aeiou".indexOf(c) != -1) vowels++;
                else consonants++;
            } else if (c != ' ') {
                specialChars++;
            }
        }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
        System.out.println("Special Characters: " + specialChars);
    }
}
```

Output:

yaml
CopyEdit

```
Vowels: 3
Consonants: 7
Special Characters: 3
```

---

5. Anagram Checking (Least Inbuilt Methods)

java
CopyEdit

```
import java.util.Arrays;

public class AnagramCheck {
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";
        System.out.println(isAnagram(str1, str2));
    }

    public static boolean isAnagram(String str1, String str2) {
        if (str1.length() != str2.length()) return false;

        int[] count = new int[26];
        for (char c : str1.toCharArray()) count[c - 'a']++;
        for (char c : str2.toCharArray()) count[c - 'a']--;

        for (int i : count) if (i != 0) return false;
        return true;
    }
}
```

Output:

```
true
```

---

6. Pangram Checking (Least Inbuilt Methods)

java
CopyEdit
```java
public class PangramCheck {
    public static void main(String[] args) {
        String str = "The quick brown fox jumps over the lazy dog";
        System.out.println(isPangram(str));
    }

    public static boolean isPangram(String str) {
        boolean[] letters = new boolean[26];
        str = str.toLowerCase();

        for (char c : str.toCharArray()) {
            if (c >= 'a' && c <= 'z') {
                letters[c - 'a'] = true;
            }
        }

        for (boolean b : letters) if (!b) return false;
        return true;
    }
}
```

Output:
```
true
```

---

7. Find If String Contains All Unique Characters

java
CopyEdit
```java
import java.util.HashSet;

public class UniqueCharacters {
    public static void main(String[] args) {
        String str = "abcdef";
        System.out.println(hasUniqueCharacters(str));
    }

    public static boolean hasUniqueCharacters(String str) {
        HashSet<Character> set = new HashSet<>();
        for (char c : str.toCharArray()) {
            if (!set.add(c)) return false;
        }
        return true;
    }
}
```

Output:

```
true
```

---

8. Find the Maximum Occurring Character in a String

java
CopyEdit

```
import java.util.HashMap;

public class MaxOccurringChar {
    public static void main(String[] args) {
        String str = "test string";
        System.out.println("Max Occurring Character: " +
getMaxOccurringChar(str));
    }

    public static char getMaxOccurringChar(String str) {
        HashMap<Character, Integer> map = new HashMap<>();
        char maxChar = ' ';
        int maxCount = 0;

        for (char c : str.toCharArray()) {
            if (c != ' ') {
                map.put(c, map.getOrDefault(c, 0) + 1);
                if (map.get(c) > maxCount) {
                    maxCount = map.get(c);
                    maxChar = c;
                }
            }
        }
        return maxChar;
    }
}
```

Output:

```
Max Occurring Character: t
```