### **1. Write a Program to Remove Duplicates from a String**

java

import java.util.LinkedHashSet;

import java.util.Set;

public class RemoveDuplicates {

public static void main(String[] args) {

String str = "programming";

Set<Character> set = new LinkedHashSet<>();

for (char c : str.toCharArray()) {

set.add(c);

}

StringBuilder result = new StringBuilder();

for (char c : set) {

result.append(c);

}

System.out.println("String after removing duplicates: " + result.toString());

}

}

### **2. Write a Program to Print Duplicate Characters from a String**

java

import java.util.HashMap;

import java.util.Map;

public class PrintDuplicates {

public static void main(String[] args) {

String str = "programming";

Map<Character, Integer> map = new HashMap<>();

for (char c : str.toCharArray()) {

map.put(c, map.getOrDefault(c, 0) + 1);

}

System.out.println("Duplicate characters:");

for (Map.Entry<Character, Integer> entry : map.entrySet()) {

if (entry.getValue() > 1) {

System.out.println(entry.getKey());

}

}

}

}

### **3. Write a Program to Check if "2552" is a Palindrome or Not**

java

public class PalindromeCheck {

public static void main(String[] args) {

String str = "2552";

String reversed = new StringBuilder(str).reverse().toString();

if (str.equals(reversed)) {

System.out.println(str + " is a palindrome.");

} else {

System.out.println(str + " is not a palindrome.");

}

}

}

### **4. Write a Program to Count the Number of Consonants, Vowels, Special Characters in a String**

java

public class CountCharacters {

public static void main(String[] args) {

String str = "Hello, World!";

int vowels = 0, consonants = 0, specialChars = 0;

for (char c : str.toCharArray()) {

if (Character.isLetter(c)) {

if ("AEIOUaeiou".indexOf(c) != -1) {

vowels++;

} else {

consonants++;

}

} else if (!Character.isWhitespace(c)) {

specialChars++;

}

}

System.out.println("Vowels: " + vowels);

System.out.println("Consonants: " + consonants);

System.out.println("Special Characters: " + specialChars);

}

}

### **5. Write a Program to Implement Anagram Checking with Least Inbuilt Methods**

java

import java.util.Arrays;

public class AnagramCheck {

public static void main(String[] args) {

String str1 = "listen";

String str2 = "silent";

if (str1.length() != str2.length()) {

System.out.println("Strings are not anagrams.");

return;

}

char[] arr1 = str1.toCharArray();

char[] arr2 = str2.toCharArray();

Arrays.sort(arr1);

Arrays.sort(arr2);

if (Arrays.equals(arr1, arr2)) {

System.out.println("Strings are anagrams.");

} else {

System.out.println("Strings are not anagrams.");

}

}

}

### **6. Write a Program to Implement Pangram Checking with Least Inbuilt Methods**

java

public class PangramCheck {

public static void main(String[] args) {

String str = "The quick brown fox jumps over the lazy dog";

str = str.toLowerCase();

boolean[] alphabet = new boolean[26];

for (char c : str.toCharArray()) {

if (c >= 'a' && c <= 'z') {

alphabet[c - 'a'] = true;

}

}

boolean isPangram = true;

for (boolean b : alphabet) {

if (!b) {

isPangram = false;

break;

}

}

if (isPangram) {

System.out.println("The string is a pangram.");

} else {

System.out.println("The string is not a pangram.");

}

}

}

### **7. Write a Program to Find if a String Contains All Unique Characters**

java

import java.util.HashSet;

public class UniqueCharacters {

public static void main(String[] args) {

String str = "abcdefg";

HashSet<Character> set = new HashSet<>();

for (char c : str.toCharArray()) {

if (set.contains(c)) {

System.out.println("The string does not contain all unique characters.");

return;

}

set.add(c);

}

System.out.println("The string contains all unique characters.");

}

}

### **8. Write a Program to Find the Maximum Occurring Character in a String**

java

import java.util.HashMap;

import java.util.Map;

public class MaxOccurringCharacter {

public static void main(String[] args) {

String str = "programming";

Map<Character, Integer> map = new HashMap<>();

for (char c : str.toCharArray()) {

map.put(c, map.getOrDefault(c, 0) + 1);

}

char maxChar = str.charAt(0);

int maxCount = 1;

for (Map.Entry<Character, Integer> entry : map.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

maxChar = entry.getKey();

}

}

System.out.println("The maximum occurring character is: " + maxChar);

}

}