**ASSIGNMENT Strinng (05-feb):**

**1.WAP(Write a program) to remove dublicates from a string(take any sting example with dublicate sting character).**

**Ans:** WRITING A PROGRAM TO REMOVE DUBLICATES FROM STRING

Program:

import java.util.LinkedHashSet;

public class RemoveDuplicates {

public static void main(String[] args) {

String inputString = "Hello, World!";

String result = removeDuplicates(inputString);

System.out.println("Original string: " + inputString);

System.out.println("String without duplicates: " + result);

}

public static String removeDuplicates(String input) {

// Using LinkedHashSet to maintain the order of characters

LinkedHashSet<Character> uniqueChars = new LinkedHashSet<>();

// Adding characters to the set

for (int i = 0; i < input.length(); i++) {

char currentChar = input.charAt(i);

uniqueChars.add(currentChar);

}

// Constructing the string without duplicates

StringBuilder resultBuilder = new StringBuilder();

for (char c : uniqueChars) {

resultBuilder.append(c);

}

return resultBuilder.toString();

}

}

Output: Original string: Hello, World!

String without duplicates: Helo, Wrd!

**2. Write a program to print dublicates character from the string.**

**Ans:** Here's an example program in Java that prints duplicate characters from a string:

Program:

import java.util.HashMap;

import java.util.Map;

public class PrintDuplicates {

public static void main(String[] args) {

String inputString = "Hello, World!";

printDuplicates(inputString);

}

public static void printDuplicates(String input) {

// Map to store character frequencies

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

// Counting character frequencies

for (int i = 0; i < input.length(); i++) {

char currentChar = input.charAt(i);

charFrequencies.put(currentChar, charFrequencies.getOrDefault(currentChar, 0) + 1);

}

// Printing duplicate characters

System.out.println("Duplicate characters in the string:");

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

if (entry.getValue() > 1) {

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

}

}

}

}

Output:

Duplicate characters in the string:

l

o

**3.write a program to check if “2552” is palidrome or not.**

**Ans:** public class PalindromeChecker {

public static boolean isPalindrome(String str) {

// Convert the string to lowercase for case-insensitive comparison

str = str.toLowerCase();

// Create a StringBuilder object to reverse the string

StringBuilder reversedStr = new StringBuilder(str);

reversedStr.reverse();

// Check if the original string is equal to its reverse

if (str.equals(reversedStr.toString())) {

return true;

} else {

return false;

}

}

public static void main(String[] args) {

String inputString = "2552";

if (isPalindrome(inputString)) {

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

} else {

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

}

}

}

**4.** **write a program to count the no of consonants wovels ,special characters in string .**

public class CharacterCounter {

public static void countCharacters(String str) {

// Convert the string to lowercase for case-insensitive comparison

str = str.toLowerCase();

int vowels = 0;

int consonants = 0;

int specialChars = 0;

for (int i = 0; i < str.length(); i++) {

char ch = str.charAt(i);

if (Character.isLetter(ch)) {

// Check if the character is a vowel

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

vowels++;

} else {

consonants++;

}

} else {

// Check if the character is a special character

specialChars++;

}

}

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

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

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

}

public static void main(String[] args) {

String inputString = "Hello! How are you?";

countCharacters(inputString);

}

}

**5.write a program to implement anagram checking inbuilt methods being used.in java**

import java.util.Arrays;

public class AnagramChecker {

public static boolean areAnagrams(String str1, String str2) {

// Convert strings to lowercase and remove spaces

str1 = str1.toLowerCase().replaceAll("\\s", "");

str2 = str2.toLowerCase().replaceAll("\\s", "");

// Check if the lengths of the strings are equal

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

return false;

}

// Convert strings to character arrays

char[] charArray1 = str1.toCharArray();

char[] charArray2 = str2.toCharArray();

// Sort the character arrays

Arrays.sort(charArray1);

Arrays.sort(charArray2);

// Compare the sorted arrays

return Arrays.equals(charArray1, charArray2);

}

public static void main(String[] args) {

String str1 = "listen";

String str2 = "silent";

if (areAnagrams(str1, str2)) {

System.out.println(str1 + " and " + str2 + " are anagrams.");

} else {

System.out.println(str1 + " and " + str2 + " are not anagrams.");

}

}

}

**6.write a program to implement pangram checking with least inbuilt method being used .in java**

**Ans:** public class PangramChecker {

public static boolean isPangram(String str) {

// Create a boolean array to track the occurrence of each letter

boolean[] letters = new boolean[26];

// Convert the string to lowercase

str = str.toLowerCase();

// Iterate through each character in the string

for (int i = 0; i < str.length(); i++) {

char c = str.charAt(i);

// Check if the character is an alphabet letter

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

// Mark the corresponding index as true in the boolean array

letters[c - 'a'] = true;

}

}

// Check if all the letters have occurred at least once

for (boolean letter : letters) {

if (!letter) {

return false;

}

}

return true;

}

public static void main(String[] args) {

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

if (isPangram(sentence)) {

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

} else {

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

}

}

}

**7.write a program to find if sting contains unique characters.**

public class UniqueCharactersChecker {

public static boolean hasUniqueCharacters(String str) {

// Create a boolean array to track the occurrence of each character

boolean[] characters = new boolean[128];

// Iterate through each character in the string

for (int i = 0; i < str.length(); i++) {

char c = str.charAt(i);

// Check if the character has already occurred

if (characters[c]) {

return false;

}

// Mark the corresponding index as true in the boolean array

characters[c] = true;

}

return true;

}

public static void main(String[] args) {

String input = "OpenAI";

if (hasUniqueCharacters(input)) {

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

} else {

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

}

}

}

**8.** **write a program to find the maximum occuring character in string.**

**Ans:** public class MaxOccurringCharacter {

public static char getMaxOccurringCharacter(String str) {

// Create an array to track the occurrence of each character

int[] count = new int[128];

// Iterate through each character in the string

for (int i = 0; i < str.length(); i++) {

char c = str.charAt(i);

count[c]++;

}

char maxChar = ' ';

int maxCount = 0;

// Find the character with the maximum occurrence

for (int i = 0; i < count.length; i++) {

if (count[i] > maxCount) {

maxCount = count[i];

maxChar = (char) i;

}

}

return maxChar;

}

public static void main(String[] args) {

String input = "Hello World";

char maxChar = getMaxOccurringCharacter(input);

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

}

}