**Strings - Handling strings and common string functions**

**Functions - parameter passing, return**

**1. Count Vowels and Consonants**

**Problem:**

**Write a Java program to count the number of vowels and consonants in a given string.**

| **import java.util.\*;  public class VowelConsonantCount {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  int vowelCount = 0;  int consonantCount = 0;  String text = sc.next();  text = text.toLowerCase();  for (int i = 0; i < text.length(); i++) {  char ch = text.charAt(i);  if (Character.isLetter(ch)) {  if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')  vowelCount++;  else  consonantCount++;  }  }  System.out.println("Vowel count = " + vowelCount);  System.out.println("Consonant count = " + consonantCount);  }  } }** |
| --- |

**2. Reverse a String**

**Problem:**

**Write a Java program to reverse a given string without using any built-in reverse functions.**

| **import java.util.\*;  public class ReverseString {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  StringBuilder sb = new StringBuilder();  String text = sc.next();  for (int i = text.length() - 1; i >= 0; i--) {  sb.append(text.charAt(i));  }  System.out.println("Reversed String: " + sb.toString());  }  } }** |
| --- |

**3. Palindrome String Check**

**Problem:**

**Write a Java program to check if a given string is a palindrome (a string that reads the same forward and backward).**

| **import java.util.\*;  public class Palindrome {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  StringBuilder sb = new StringBuilder();  String text = sc.next();  text = text.toLowerCase();  for (int i = text.length() - 1; i >= 0; i--) {  sb.append(text.charAt(i));  }  String reverse = sb.toString();  if (text.equals(reverse))  System.out.println("Is a palindrome");  else  System.out.println("Not a palindrome");  }  } }** |
| --- |

**4. Remove Duplicates from a String**

**Problem:**

**Write a Java program to remove all duplicate characters from a given string and return the modified string.**

| **import java.util.\*;  public class RemoveDuplicates {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  boolean[] visited = new boolean[256];  String text = sc.next();  text = text.toLowerCase();  StringBuilder unique = new StringBuilder();  for (int i = 0; i < text.length(); i++) {  if (!visited[text.charAt(i)]) {  unique.append(text.charAt(i));  visited[text.charAt(i)] = true;  }  }  System.out.println("Without duplicates: " + unique.toString());  }  } }** |
| --- |

**5. Find the Longest Word in a Sentence**

**Problem:**

**Write a Java program that takes a sentence as input and returns the longest word in the sentence.**

| **import java.util.Scanner;  class LongestWord {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  ;  String input = sc.nextLine();  String[] list = input.split("\\s+");  String longest = "";  for (String word : list) {  if (word.length() > longest.length())  longest = word;  }  System.out.println("Longest word is: " + longest);  }  } }** |
| --- |

**6. Find Substring Occurrences**

**Problem:**

**Write a Java program to count how many times a given substring occurs in a string.**

| **import java.util.Scanner;  class CountOccurences {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in);) {  String main = sc.nextLine();  String sub = sc.nextLine();  int index = 0;  int count = 0;  while ((index = main.indexOf(sub, index)) != -1) {  count++;  index += sub.length();  }  System.out.println("Occurences = " + count);  }  } }** |
| --- |

**7. Toggle Case of Characters**

**Problem:**

**Write a Java program to toggle the case of each character in a given string. Convert uppercase letters to lowercase and vice versa.**

| **import java.util.Scanner;  public class ToggleCase {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter a string: ");  String input = sc.nextLine();   System.out.println("Toggled String: " + toggleCase(input));   sc.close();  }   public static String toggleCase(String str) {  StringBuilder result = new StringBuilder();  for (char ch : str.toCharArray()) {  result.append(Character.isUpperCase(ch) ? Character.toLowerCase(ch) : Character.toUpperCase(ch));  }  return result.toString();  } }** |
| --- |

**8. Compare Two Strings**

**Problem:**

**Write a Java program to compare two strings lexicographically (dictionary order) without using built-in compare methods.**

**Example Input:**

**String 1: "apple"**

**String 2: "banana"**

**Expected Output:**

**"apple" comes before "banana" in lexicographical order**

| **import java.util.Scanner;  public class CompareStrings {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  String str1 = scanner.nextLine();  String str2 = scanner.nextLine();   int len1 = str1.length();  int len2 = str2.length();  int minLen = Math.min(len1, len2);  boolean areEqual = true;   for (int i = 0; i < minLen; i++) {  char ch1 = str1.charAt(i);  char ch2 = str2.charAt(i);  if (ch1 < ch2) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  areEqual = false;  break;  } else if (ch1 > ch2) {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  areEqual = false;  break;  }  }   if (areEqual) {  if (len1 < len2) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  } else if (len1 > len2) {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  } else {  System.out.println("Both strings are equal");  }  }   scanner.close();  } }** |
| --- |

**9. Find the Most Frequent Character**

**Problem:**

**Write a Java program to find the most frequent character in a string. Example Input:**

**String: "success"**

**Expected Output:**

**Most Frequent Character: 's'**

| **import java.util.Scanner;  public class MostFrequentChar {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  System.out.print("Enter a string: ");  String input = sc.nextLine();   System.out.println("Most Frequent Character: '" + findMostFrequentChar(input) + "'");  }  }   public static char findMostFrequentChar(String str) {  int[] freq = new int[256];  char maxChar = str.charAt(0);  int maxCount = 0;   for (char ch : str.toCharArray()) {  freq[ch]++;  if (freq[ch] > maxCount) {  maxCount = freq[ch];  maxChar = ch;  }  }   return maxChar;  } }** |
| --- |

**10. Remove a Specific Character from a String**

**Problem:**

**Write a Java program to remove all occurrences of a specific character from a string. Example Input:**

**String: "Hello World"**

**Character to Remove: 'l'**

**Expected Output:**

**Modified String: "Heo Word"**

| **import java.util.Scanner;  public class RemoveCharacter {  public static void main(String[] args) {  try (Scanner sc = new Scanner(System.in)) {  System.out.print("Enter a string: ");  String input = sc.nextLine();  System.out.print("Enter character to remove: ");  char ch = sc.next().charAt(0);   System.out.println("Modified String: " + removeCharacter(input, ch));  }  }   public static String removeCharacter(String str, char ch) {  return str.replace(String.valueOf(ch), "");  } }** |
| --- |

**11. Write a Java program that accepts two strings from the user and checks if the two strings are anagrams of each other (i.e., whether they contain the same characters in any order).**

| **import java.util.Arrays; import java.util.Scanner;  public class AnagramChecker {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter first string: ");  String str1 = sc.next();  System.out.print("Enter second string: ");  String str2 = sc.next();   if (areAnagrams(str1, str2)) {  System.out.println("The strings are anagrams.");  } else {  System.out.println("The strings are not anagrams.");  }   sc.close();  }   public static boolean areAnagrams(String str1, String str2) {  if (str1.length() != str2.length())  return false;   char[] arr1 = str1.toCharArray();  char[] arr2 = str2.toCharArray();   Arrays.sort(arr1);  Arrays.sort(arr2);   return Arrays.equals(arr1, arr2);  } }** |
| --- |

**12. Write a replace method in Java that replaces a given word with another word in a sentence:**

| **import java.util.Scanner;  public class WordReplacer {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter a sentence: ");  sc.nextLine();  String sentence = sc.nextLine();  System.out.print("Enter word to replace: ");  String oldWord = sc.next();  System.out.print("Enter replacement word: ");  String newWord = sc.next();   System.out.println("Modified Sentence: " + replaceWord(sentence, oldWord, newWord));   sc.close();  }   public static String replaceWord(String sentence, String oldWord, String newWord) {  return sentence.replace(oldWord, newWord);  } }** |
| --- |