**✅ Basic String Programs:**

1. **Reverse a String**
2. **Check if a String is a Palindrome**
3. **Count vowels and consonants in a String**
4. **Find the length of a String without using .length()**
5. **Compare two Strings without using .equals()**
6. **Remove all white spaces from a String**
7. **Count the number of words in a String**
8. **Find duplicate characters in a String**
9. **Check if two Strings are Anagrams**

### Toggle each character in a String (upper ⇄ lower)

### 1 .Reverse a String

### ****Using a**** for ****loop:****

public class ReverseString {

public static void main(String[] args) {

String original = "Java Developer";

String reversed = "";

for (int i = original.length() - 1; i >= 0; i--) {

reversed += original.charAt(i);

}

System.out.println("Original: " + original);

System.out.println("Reversed: " + reversed);

}

}

### ✅ ****2. Using**** StringBuilder ****(Recommended way):****

public class ReverseStringBuilder {

public static void main(String[] args) {

String original = "Java Developer";

StringBuilder sb = new StringBuilder(original);

String reversed = sb.reverse().toString();

System.out.println("Original: " + original);

System.out.println("Reversed: " + reversed);

}

}

2. **Check if a String is a Palindrome**

### ****1. Using a**** for ****loop (Manual check):****

public class PalindromeCheck {

public static void main(String[] args) {

String str = "madam";

String reversed = "";

// Reverse the string using a for loop

for (int i = str.length() - 1; i >= 0; i--) {

reversed += str.charAt(i);

}

// Compare original and reversed strings

if (str.equals(reversed)) {

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

} else {

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

}

}

}

### ****Second way to check the string :****

public class PalindromeCheck {

public static void main(String[] args) {

String original = "madam";

boolean isPalindrome = true;

for (int i = 0; i < original.length() / 2; i++) {

if (original.charAt(i) != original.charAt(original.length() - 1 - i)) {

isPalindrome = false;

break;

}

}

if (isPalindrome) {

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

} else {

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

}

}

}

### ✅ ****2. Using**** StringBuilder ****(One-liner comparison):****

java

CopyEdit

public class PalindromeUsingBuilder {

public static void main(String[] args) {

String original = "racecar";

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

if (original.equals(reversed)) {

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

} else {

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

}

}

}

**3. Count vowels and consonants in a String**

### ****Java Program: Count Vowels and Consonants****

public class VowelConsonantCounter {

public static void main(String[] args) {

String input = "Java Developer";

int vowels = 0, consonants = 0;

// Convert to lowercase for easier comparison

input = input.toLowerCase();

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

char ch = input.charAt(i);

if (ch >= 'a' && ch <= 'z') { // Check if it's an alphabet

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

vowels++;

} else {

consonants++;

}

}

}

System.out.println("Input String: " + input);

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

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

}

}

### 🧪 Sample Output:

For input "Java Developer":

vbnet

CopyEdit

Input String: java developer

Vowels: 6

Consonants: 8

4. **Find the length of a String without using .length()**

**Java Program: Find Length of a String Without .length()**

public class StringLengthWithoutLengthMethod {

public static void main(String[] args) {

String input = "Java Developer";

int length = 0;

// Iterate through the string and count the characters

for (int i = 0; ; i++) {

try {

input.charAt(i); // Try to access the character at index 'i'

length++; // If no exception, it means the index is valid, so increase length

} catch (StringIndexOutOfBoundsException e) {

break; // When the exception occurs, we've reached the end of the string

}

}

System.out.println("The length of the string is: " + length);

}

}

**🧪 Explanation:**

1. We **try** to access each character of the string using input.charAt(i).
2. As long as we don't get an exception, we increase the count (length++).
3. Once we reach the end of the string, a StringIndexOutOfBoundsException is thrown, and we **break** the loop.

**🧪 Sample Output:**

For input "Java Developer":

The length of the string is: 14

**5. Compare two Strings without using .equals()**

### ****Java Program: Compare Strings Without**** .equals()

public class StringCompareWithoutEquals {

public static void main(String[] args) {

String str1 = "hello";

String str2 = "hello";

boolean areEqual = true;

// Step 1: Check length

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

areEqual = false;

} else {

// Step 2: Compare each character

for (int i = 0; i < str1.length(); i++) {

if (str1.charAt(i) != str2.charAt(i)) {

areEqual = false;

break;

}

}

}

// Step 3: Output the result

if (areEqual) {

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

} else {

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

}

}

}

### 🧪 Sample Output:

If both strings are "hello":

sql

CopyEdit

Strings are equal.

If one is "hello" and the other is "hella":

Strings are not equal.

**6. Remove all white spaces from a String**

### ****Java Program: Remove All White Spaces****

public class RemoveWhiteSpaces {

public static void main(String[] args) {

String input = " Java Developer ";

// Using replaceAll() with regex to remove all whitespaces

String result = input.replaceAll("\\s", "");

System.out.println("Original String: \"" + input + "\"");

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

}

}

**7. Count the number of words in a String**

### ****Java Program: Count Words in a String****

public class WordCounter {

public static void main(String[] args) {

String input = "Java is a powerful language";

// Trim the string to remove leading/trailing spaces

// Split the string by one or more spaces using regex "\\s+"

String[] words = input.trim().split("\\s+");

System.out.println("Input String: \"" + input + "\"");

System.out.println("Number of words: " + words.length);

}

}

**8. Find duplicate characters in a String**

### ****Java Program: Find Duplicate Characters in a String****

import java.util.HashMap;

import java.util.Map;

public class DuplicateCharacters {

public static void main(String[] args) {

String input = "programming";

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

// Convert to lowercase for uniform comparison (optional)

input = input.toLowerCase();

// Count frequency of each character

for (char ch : input.toCharArray()) {

if (ch != ' ') { // ignore spaces if present

charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);

}

}

// Print characters that appear more than once

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

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

if (entry.getValue() > 1) {

System.out.println(entry.getKey() + " → " + entry.getValue() + " times");

}

}

}

}

**9.Check if two Strings are Anagrams**

### ****What is an Anagram?****

An **anagram** is a word or phrase formed by **rearranging the letters** of another word or phrase.

**Rules for Anagrams:**

1. **Same characters**
2. **Same number of each character**
3. **Just rearranged**
4. **Usually ignore case and spaces**

### ****Java Program: Check Anagram****

import java.util.Arrays;

public class AnagramChecker {

public static void main(String[] args) {

String str1 = "listen";

String str2 = "silent";

if (isAnagram(str1, str2)) {

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

} else {

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

}

}

static boolean isAnagram(String s1, String s2) {

// Remove spaces and convert to lowercase

s1 = s1.replaceAll("\\s", "").toLowerCase();

s2 = s2.replaceAll("\\s", "").toLowerCase();

// Check if lengths are same

if (s1.length() != s2.length()) {

return false;

}

// Convert to char array and sort

char[] arr1 = s1.toCharArray();

char[] arr2 = s2.toCharArray();

Arrays.sort(arr1);

Arrays.sort(arr2);

// Compare sorted arrays

return Arrays.equals(arr1, arr2);

}

}

### ****What does**** Arrays.sort(arr1) ****do?****

It **sorts the elements** of the array arr1 in **ascending order**.

Similarly:

Arrays.sort(arr2);

Sorts the second array arr2.

### 📦 Example:

String str1 = "listen";

char[] arr1 = str1.toCharArray(); // ['l','i','s','t','e','n']

Arrays.sort(arr1); // ['e','i','l','n','s','t']

Same happens with str2 = "silent" → becomes ['e','i','l','n','s','t']

### 🤔 Why do we sort in anagram checking?

Because **anagrams have the same letters**, just **in different order**.  
By sorting both strings, we bring them into a **standard format** to compare easily.

If sorted arrays are **equal**, then the strings are anagrams.

**11. Toggle each character in a String (upper ⇄ lower)**

### ****Java Program: Toggle Case****

public class ToggleCase {

public static void main(String[] args) {

String input = "Hello World!";

StringBuilder toggled = new StringBuilder();

for (char ch : input.toCharArray()) {

if (Character.isUpperCase(ch)) {

toggled.append(Character.toLowerCase(ch));

} else if (Character.isLowerCase(ch)) {

toggled.append(Character.toUpperCase(ch));

} else {

toggled.append(ch); // Keep non-letter characters as they are

}

}

System.out.println("Original: " + input);

System.out.println("Toggled : " + toggled);

}

}

### 🧪 Output:

yaml

CopyEdit

Original: Hello World!

Toggled : hELLO wORLD!

**Advanced / Logic-Based String Programs**:

1. **Reverse each word in a sentence**

### Java Program: Reverse Each Word (No StringBuilder)

java

CopyEdit

public class ReverseEachWordWithoutStringBuilder {

public static void main(String[] args) {

String sentence = "Java is fun";

String[] words = sentence.split(" ");

String result = "";

for (String word : words) {

String reversed = "";

for (int i = word.length() - 1; i >= 0; i--) {

reversed += word.charAt(i); // reversing the word manually

}

result += reversed + " ";

}

System.out.println("Original: " + sentence);

System.out.println("Reversed Words: " + result.trim());

}

}