| Experiment No. 9 |
| --- |
| To implement various string functions in programming to manipulate and process textual data |
| Date of Performance:16/08/24 |
| Date of Submission:23/08/24 |

**Aim:-** **Aim:** To implement various String functions in programming to manipulate and process textual data

**Objective:-** Develop a Java application that reads user-inputted sentences and performs various string manipulation tasks. The application should count characters, reverse sentences, find substrings, convert to uppercase and lowercase, remove whitespace, replace characters, and check for palindromes. Utilize appropriate methods from the String, StringBuilder, and StringBuffer classes to ensure efficient and effective handling of these operations, displaying the results in a clear, user-friendly manner.

**Theory:-**

In Java, strings are objects that represent sequences of characters. The String class is part of the java.lang package and provides various methods for manipulating and handling strings. Unlike primitive data types, strings are immutable, meaning once created, their values cannot be changed.

Creating Strings

Strings can be created in multiple ways:

Using String Literal:

String str = "Hello, World!";

This method uses the string pool for efficient memory usage.

Using new Keyword:

String str = new String("Hello, World!");

This creates a new string object each time it is called.

String Methods

Length:

int length = str.length();

Concatenation:

String concatenated = str1 + str2;

String concatenated = str1.concat(str2);

Character Access:

char ch = str.charAt(0);

Substring:

String sub = str.substring(0, 5);

Equality:

boolean isEqual = str1.equals(str2);

boolean isEqualIgnoreCase = str1.equalsIgnoreCase(str2);

Comparison:

int comparison = str1.compareTo(str2);

Searching:

int index = str.indexOf('H');

int lastIndex = str.lastIndexOf('o');

Replacing:

String replaced = str.replace('l', 'p');

Trimming:

String trimmed = str.trim();

Converting Case:

String upper = str.toUpperCase();

String lower = str.toLowerCase();

Immutable Nature of Strings

When a string is modified, a new string object is created, and the reference is updated. This is because strings are immutable in Java. For example:

String str = "Hello";

str = str + " World"; // A new string "Hello World" is created

StringBuffer and StringBuilder

For mutable strings, Java provides StringBuffer and StringBuilder classes:

StringBuffer: Thread-safe and synchronized.

StringBuilder: Not thread-safe but faster.

Example:

StringBuffer sb = new StringBuffer("Hello");

sb.append(" World");

Practical Examples

Palindrome Check:

public boolean isPalindrome(String str) {

int n = str.length();

for (int i = 0; i < n / 2; i++) {

if (str.charAt(i) != str.charAt(n - i - 1)) {

return false;

}

}

return true;

}

String Reversal:

public String reverseString(String str) {

return new StringBuilder(str).reverse().toString();

}

Understanding and effectively using strings in Java is crucial for handling text data and performing various string operations efficiently.

**Code:-**

import java.util.Scanner;

public class ReviewAnalyzer {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a review: ");

String review = scanner.nextLine();

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

System.out.println("Word Count: " + words.length);

System.out.println("Review Length: " + review.length());

System.out.println("Uppercase Review: " + review.toUpperCase());

System.out.println("Lowercase Review: " + review.toLowerCase());

if (review.length() > 10) {

System.out.println("Substring (first 10 characters): " + review.substring(0, 10));

} else {

System.out.println("Substring: Review too short to extract 10 characters.");

}

String replacedReview = review.replace("bad", "good");

System.out.println("Replaced Review: " + replacedReview);

String trimmedReview = review.trim();

System.out.println("Trimmed Review: " + trimmedReview);

String referenceReview = "This product is great!";

if (review.equals(referenceReview)) {

System.out.println("Review matches the reference review.");

} else {

System.out.println("Review does not match the reference review.");

}

String comment = " - Thank you for your feedback!";

String concatenatedReview = review.concat(comment);

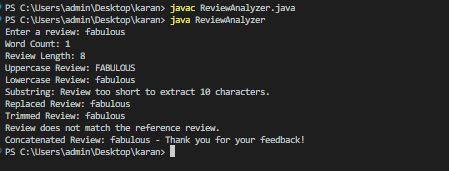
System.out.println("Concatenated Review: " + concatenatedReview);

scanner.close();

}

}

**Output:**

****

**Conclusion:-** The String class in Java is highly effective, providing immutable strings, extensive methods for manipulation, and seamless integration with other classes. Its functionality supports various operations, like searching, replacing, and formatting, making it essential for robust text processing in applications.