CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3rd

Subject Code:CSE201

Academic year: 2024-25

PART-II Strings

No.	Aim of the Practical

7.	AIM:
	
	Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho'
	
	front_times('Chocolate', 3) → 'ChoChoCho'
	
	front_times('Abc', 3) → 'AbcAbcAbc'
	
	PROGRAM CODE :
	
	import java.util.Scanner;
	
	public class p2s1 {
	
	public static void main(String[] args)
	{
	
	Scanner scanner = new Scanner(System.in);

```
        System.out.print("Enter a string:
");

        String str = scanner.next();



        System.out.print("Enter a non-
negative integer: ");

        int n = scanner.nextInt();



        System.out.println(frontTimes(str,
n));

    }



  public static  String frontTimes(String
str, int n) {

        String front = str.substring(0,
Math.min(3, str.length()));

         StringBuilder result = new
StringBuilder();



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

            result.append(front);

        }



        return result.toString();

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java p2s1
Enter a string: Saumya
Enter a non-negative integer: 3
SauSauSau
```

CONCLUSION:

From this Java code, we can conclude the following:

The method `frontTimes` takes two parameters: a `String` `str` and an `int` `n`.
The method returns a `String` that is `n` copies of the "front" of the input string `str`.
The "front" of the string is defined as the first 3 characters, or the entire string if it is shorter than 3 characters.
The `Math.min(3, str.length())` expression is used to ensure that we don't try to take a substring of length 3 from a string that is shorter than 3 characters.
The method uses a `for` loop to concatenate the "front" string `n` times to form the result string.
The method returns the resulting string.

8.

AIM:

Given an array of ints, return the number of 9's in the

array. array_count9([1, 2, 9]) → 1

array_count9([1, 9, 9]) → 2

array_count9([1, 9, 9, 3, 9]) → 3

PROGRAM:

```
import java.util.Scanner;

public class p2s2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
```

```
        int[] arr = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        System.out.println("Number of 9's in the array: " + arrayCount9(arr));
    }

    public static int arrayCount9(int[] arr) {
        int count = 0;
        for (int i : arr) {
            if (i == 9) {
                count++;
            }
        }
        return count;
    }
}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java p2s2
Enter the number of elements in the array: 5
Enter the elements of the array:
12
9
8
13
9
Number of 9's in the array: 2
```

CONCLUSION:

We initialize a variable `count` to 0, which will store the number of 9's in the array.

We use a `for` loop to iterate over the elements of the input array `nums`.
Inside the loop, we check if the current element `num` is equal to 9. If it is, we increment the `count` variable by 1.
After the loop finishes, we return the final value of `count`, which represents the number of 9's in the array.

Supplementary Experiment:

1. Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement.

Sample string : "The quick brown fox jumps over the lazy dog."

In the above string replace all the fox with cat.

PROGRAM:

```
public class p2s3 {

    public static void main(String[] args) {

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

        String regex = "fox";

        String replacement = "cat";



        String newString = sampleString.replace(regex, replacement);



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

```
        System.out.println("New string: " + newString);

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java sup1
Original string: The quick brown fox jumps over the lazy dog.
New string: The quick brown cat jumps over the lazy dog.
```

CONCLUSION:

From this practical, we can conclude the following:

9.

The `replaceAll()` method of the `String` class in Java can be used to replace each substring of a given string that matches a given regular expression with a given replacement.
The `replaceAll()` method takes two parameters: the regular expression to match, and the replacement string.
The regular expression can be a simple string, as in this example, or a more complex pattern using regular expression syntax.
The replacement string can be any string, including an empty string (`""`), which would effectively delete the matched substrings.
The `replaceAll()` method returns a new string with the replacements made, leaving the original string unchanged.
This method is case-sensitive, so if you want to replace substrings regardless of case, you can use the `(?i)` flag at the beginning of the regular expression to make it case-insensitive. For example: `String regex = "(?i)fox";`

AIM:

Given a string, return a string where for every char in the original, there are two chars.

double_char('The') → 'Tthhee

double_char('AAbb') → 'AAAAbbbb' ,double_char('Hi-There') → 'Hhii—Tthheerree'

PROGRAM:

```
import java.util.Scanner;
```

```
public class p2s3 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String str = scanner.nextLine();

        String newStr = "";

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

            newStr += str.substring(i, i + 1) + str.substring(i, i + 1);

        }

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

        System.out.println("New string: " + newStr);

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java p2s3
Enter a string: SAUMYA
Original string: SAUMYA
New string: SSAAUUMMYYAA
```

CONCLUSION:

The problem requires us to create a new string where each character in the original string is duplicated.

We can use a `for` loop to iterate over each character in the original string.
Inside the loop, we can use the `charAt()` method to get the current character, and then add it to the result string twice using the `+=` operator The resulting string will have each character duplicated, as required.
This problem is a good example of how to use a loop to iterate over a string and manipulate its characters.
The `charAt()` method is used to access individual characters in a string, and the `+=` operator is used to concatenate strings.

10.	The problem does not specify any restrictions on the input string, so the solution should work for any input string, including those with special characters, uppercase and lowercase letters, and digits.
	
	AIM:
	
	Perform following functionalities of the string:
	
	● Find Length of the String
	
	● Lowercase of the String
	
	● Uppercase of the String
	
	● Reverse String
	
	Sort the string
	
	PROGRAM:
	
	import java.util.*;
	
	import java.lang.*;
	
	class p2s4
	
	{
	
	public static void main(String []args)
	
	{
	
	Scanner s=new Scanner(System.in);
	
	System.out.println("Enter your string:");
	
	String x=s.nextLine();
	
	int n=x.length();
	
	System.out.println("Length of the string is:"+n);
	
	String y=x.toUpperCase();
	
	System.out.println("Upper case of the string is:"+y);

```
String z=x.toLowerCase();
System.out.println("Lower case of the string is:"+z);

String reverse = "";
for (int i = x.length() - 1; i >= 0; i--)
{
   reverse += x.charAt(i);
}
System.out.println("Reversed string: " + reverse);

 char[] ch = x.toCharArray();
       Arrays.sort(ch);
                  System.out.println("Sorted string:"+new String(ch));
}
}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java p2s4
Enter your string:
Saumya
Length of the string is:6
Upper case of the string is:SAUMYA
Lower case of the string is:saumya
Reversed string: aymuaS
Sorted string:Saamuy
```

CONCLUSION:

From this practical, we can conclude the following:

The `length()` method of the `String` class returns the length of the string.
The `toLowerCase()` and `toUpperCase()` methods of the `String` class return the lowercase and uppercase versions of the string, respectively.

11.

To reverse a string, we can use a loop to iterate over the characters of the string in reverse order and concatenate them to a new string.
To sort the characters of a string, we can convert the string to a character array, sort the array using the `Arrays.sort()` method, and then convert the sorted array back to a string.
The `String` class in Java provides various methods for manipulating strings, including `length()`, `toLowerCase()`, `toUpperCase()`, and others.
The `Arrays` class in Java provides various methods for manipulating arrays, including `sort()`.

AIM:

Perform following Functionalities of the string:

“CHARUSAT UNIVERSITY”

● Find length

● Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’

● Convert all character in lowercase

PROGRAM:

```
class p2s5
{
public static void main(String []args)
{
String s="CHARUSAT UNIVERSITY";
int n=s.length();
System.out.println("Length of the string is:"+n);

String x=s.replace('H','S');
System.out.println("After replacing string is:"+x);

String y=s.toLowerCase();
```

System.out.println("Lower case of the string is:"+y);

}

}

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java p2s5
Length of the string is:19
After replacing string is:CSARUSAT UNIVERSITY
Lower case of the string is:charusat university
```

CONCLUSION:

From this practical, we can conclude the following:

The `length()` method of the `String` class returns the length of the string.
The `replace()` method of the `String` class can be used to replace a character or a substring with another character or substring.
The `toLowerCase()` method of the `String` class can be used to convert all characters of a string to lowercase.
The `charAt()` method of the `String` class can be used to get the character at a specific index in a string.
We can use the `replace()` method to replace a character with another character, and the `toLowerCase()` method to convert the entire string to lowercase.

Supplementary Experiment:

Write a Java program to count and print all duplicates in the input string.

Sample Output:

The given string is: resource

The duplicate characters and counts are:

e appears 2 times

r appears 2 times

PROGRAM:

```
public class sup2 {

    public static void main(String[] args) {
```

```
        String str = "resource";
        System.out.println("The given string is: " + str);
        System.out.println("The duplicate characters and counts are:");
        countDuplicates(str);
    }

    public static void countDuplicates(String str) {
        char[] charArray = str.toCharArray();
        boolean[] printed = new boolean[256]; // assuming ASCII characters

        for (char c : charArray) {
            int count = 0;
            for (char d : charArray) {
                if (c == d) {
                    count++;
                }
            }
            if (count > 1 && !printed[c]) {
                System.out.println(c + " appears " + count + " times");
                printed[c] = true;
            }
        }
    }
}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 2>java sup2
The given string is: resource
The duplicate characters and counts are:
r appears 2 times
e appears 2 times
```

CONCLUSION:

From this practical, we can conclude the following:

We can use a count the occurrences of each character in a string.
We can iterate over the characters of the string using a `for` loop and a `toCharArray()` method.
We can use method to check if a character is already in the map, and the `put()` method to update the count of the character.
We can use method to iterate over the entries of the map and print the duplicate characters and their counts.
This practical demonstrates how to use to solve a problem that requires counting and storing data.