

Government College of Engineering, Jalgaon
Department of Computer Engineering
Experiment No: 05

Name:
Subject:CO310U (Application programming Lab)
Class:T.Y. B.Tech
Date of Performance:

PRN:
Sem:V(Odd)
Academic Year:2024-25
Date of Completion:

Aim: Write a program for the following

- a. Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line
- b. Write a Java program that checks whether a given string is a palindrome or not.

Required Software: OpenJDK version "1.8.0_131"

OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-2ubuntu1.16.04.3-b11)

OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)

Java Compiler Version - JAVAC 1.8.0_131

Theory:

Java string

In Java, string is basically an object that represents a sequence of char values. An array of characters works the same as Java string.. For example:

1. `char[] ch={'j','a','v','a','t','p','o','i','n','t'};`
2. `String s=new String(ch);`

is same as:

1. `String s="javatpoint";`

Java String class provides a lot of methods to perform operations on strings such as `compare()`, `concat()`, `equals()`, `split()`, `length()`, `replace()`, `compareTo()`, `intern()`, `substring()` etc. The `java.lang.String` class implements *Serializable*, *Comparable* and *CharSequence* interfaces.

CharSequence Interface.

The `CharSequence` interface is used to represent the sequence of characters. `String`, `StringBuffer` and `StringBuilder` classes implement it. It means, we can create strings in java by using these three classes. The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, you can use `StringBuffer` and `StringBuilder` classes.

What is String in java

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The `java.lang.String` class is used to create a string object.

How to create a string object?

There are two ways to create String object:

1. By string literal
2. By new keyword

1) String Literal

Java String literal is created by using double quotes. For Example:

1. String s="welcome";

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

1. String s1="Welcome";
2. String s2="Welcome";//It doesn't create a new instance

In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool, that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance. String objects are stored in a special memory area known as the "string constant pool".

Why does Java use the concept of String literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

Java String class methods

No.	Method	Description
1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length

3	static String format(String format, Object... args)	returns a formatted string.
4	static String format(Locale l, String format, Object... args)	returns formatted string with given locale.
5	String substring(int beginIndex)	returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	returns a joined string.
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	returns a joined string.
10	boolean equals(Object another)	checks the equality of string with the given object.
11	boolean isEmpty()	checks if string is empty.

12	String concat(String str)	concatenates the specified string.
13	String replace(char old, char new)	replaces all occurrences of the specified char value.
14	String replace(CharSequence old, CharSequence new)	replaces all occurrences of the specified CharSequence.
15	static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
16	String[] split(String regex)	returns a split string matching regex.
17	String[] split(String regex, int limit)	returns a split string matching regex and limit.
18	String intern()	returns an interned string.
19	int indexOf(int ch)	returns the specified char value index.
20	int indexOf(int ch, int fromIndex)	returns the specified char value index starting with given index.
21	int indexOf(String substring)	returns the specified substring index.

22	<code>int indexOf(String substring, int fromIndex)</code>	returns the specified substring index starting with given index.
23	<code>String toLowerCase()</code>	returns a string in lowercase.
24	<code>String toLowerCase(Locale l)</code>	returns a string in lowercase using specified locale.
25	<code>String toUpperCase()</code>	returns a string in uppercase.
26	<code>String toUpperCase(Locale l)</code>	returns a string in uppercase using specified locale.
27	<code>String trim()</code>	removes beginning and ending spaces of this string.
28	<code>static String valueOf(int value)</code>	converts given type into string. It is an overloaded method.

Conclusion:

Name & sign of Teacher

Program:A

```

import java.io.*; // Importing io package

class p5 {          // Defining class

    public static void main(String[] args) {          // Defining main method

        try {      // Using try

            FileInputStream fis = new FileInputStream("Input.txt"); // Reading from file
            BufferedInputStream bis = new BufferedInputStream(fis); // Buffering the input

            int i; // Declaring variables
            int occur = 0;
            while ((i = bis.read()) != -1) { // Using while loop
                char a = (char) i; // Using type casting
                if (a == args[0].charAt(0)) { // Using if condition
                    occur++; // Incrementing variable
                }
            } // Closing while loop

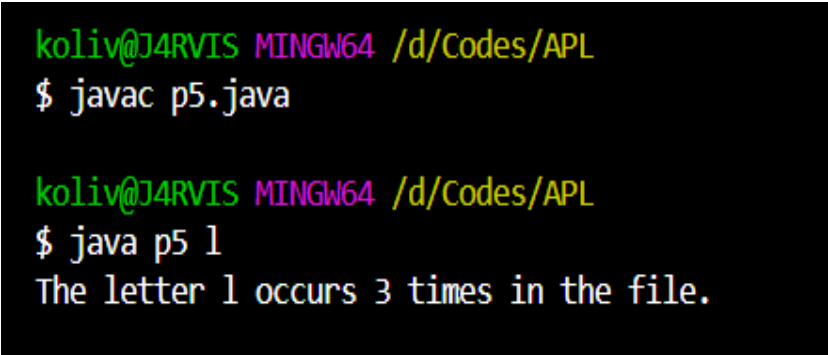
            System.out.println("The letter " + args[0].charAt(0) + " occurs " + occur + " times in the file.");

            bis.close(); // Closing the BufferedInputStream
            fis.close(); // Closing the FileInputStream

        } catch (Exception ex) { // Using catch
            System.out.println(ex.getMessage());
        } // Closing try-catch

    } // Closing main
}

```

Output:


```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p5.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p5 l
The letter l occurs 3 times in the file.

```

Program:B

```

import java.util.Scanner;

class p5 {
    public static void main(String[] args) {
        String str, rev = "";
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a string:");
        str = sc.nextLine();

        int length = str.length();
        for (int i = length - 1; i >= 0; i--) {
            rev += str.charAt(i); // Building the reversed string
        }

        if (str.equals(rev)) {
            System.out.println(str + " is a palindrome");
        } else {
            System.out.println(str + " is not a palindrome");
        }

        sc.close(); // Closing the Scanner
    }
}

```

Output:

```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p5.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p5
Enter a string:
NAYAN
NAYAN is a palindrome

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p5.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p5
Enter a string:
car
car is not a palindrome

```