1.Create a method which can perform a particular String operation based on the user’s choice. The method should accept the String object and the user’s choice and return the output of the operation.  Options are

A: Add the String to itself

B: Replace alternate positions with \*

C: Remove duplicate characters in the String

D: Change alternate characters to upper case

|  |  |
| --- | --- |
| Method Name | changeString |
| Method Description | Modify the string based on user choice |
| Argument | String string, char ch |
| Return Type | String |
| Logic | Perform the required operation based on the user choice and return the resulting string |

// A method to modify the string based on user choice

public static String changeString(String string, char ch) {

// A variable to store the result

String result = "";

// A switch case to perform the required operation based on the user choice

switch (ch) {

case 'A': // Add the string to itself

result = string + string;

break;

case 'B': // Replace alternate positions with \*

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

if (i % 2 == 0) { // If the position is even

result += "\*"; // Append \* to the result

} else { // If the position is odd

result += string.charAt(i); // Append the character at that position to the result

}

}

break;

case 'C': // Remove duplicate characters in the string

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

char c = string.charAt(i); // Get the character at that position

if (result.indexOf(c) == -1) { // If the character is not already present in the result

result += c; // Append it to the result

}

}

break;

case 'D': // Change alternate characters to upper case

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

char c = string.charAt(i); // Get the character at that position

if (i % 2 == 0) { // If the position is even

result += Character.toUpperCase(c); // Append the upper case of the character to the result

} else { // If the position is odd

result += c; // Append the character as it is to the result

}

}

break;

default: // Invalid choice

result = "Invalid choice";

}

// Return the result

return result;

}

2.What is the difference between STRING BUILDER AND STRING BUFFER.

[The difference between STRING BUILDER AND STRING BUFFER is that STRING BUILDER is **not thread-safe**, while STRING BUFFER is **thread-safe**1](https://www.geeksforgeeks.org/difference-between-stringbuffer-and-stringbuilder-in-java/)[2](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/). [This means that STRING BUILDER is faster than STRING BUFFER because it doesn’t have to perform synchronization to ensure thread safety2](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/)[3](https://www.w3docs.com/snippets/java/difference-between-stringbuilder-and-stringbuffer.html).

Some other differences are:

* STRING BUILDER was introduced in Java 5, while STRING BUFFER has been a part of the Java platform since the beginning[1](https://www.geeksforgeeks.org/difference-between-stringbuffer-and-stringbuilder-in-java/)[4](https://www.educba.com/stringbuffer-vs-stringbuilder/).
* [STRING BUILDER is implemented using an array of characters, while STRING BUFFER is implemented using a char [] array and a synchronized block3](https://www.w3docs.com/snippets/java/difference-between-stringbuilder-and-stringbuffer.html).
* [STRING BUILDER is preferred when the string can change and will only be accessed from a single thread, while STRING BUFFER is preferred when the string can change and will be accessed from multiple threads](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/)

3.Write a program called Bin2Dec to convert an input binary string into its equivalent decimal number. Your output shall look like:

Enter a Binary string: **1011**

The equivalent decimal number for binary "1011" is 11

 Enter a Binary string: **1234**

Error: Invalid Binary String "1234"

// A program called Bin2Dec to convert an input binary string into its equivalent decimal number

import java.util.Scanner; // To get user input

public class Bin2Dec {

// A method to check if a given string is a valid binary string

public static boolean isValidBinary(String s) {

// Loop through each character of the string

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

// Get the character at that position

char c = s.charAt(i);

// If the character is not 0 or 1, return false

if (c != '0' && c != '1') {

return false;

}

}

// If all characters are 0 or 1, return true

return true;

}

// A method to convert a binary string to a decimal number

public static int bin2Dec(String s) {

// Initialize a variable to store the result

int result = 0;

// Loop through each character of the string from right to left

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

// Get the character at that position

char c = s.charAt(i);

// Convert the character to a digit (0 or 1)

int d = Character.getNumericValue(c);

// Multiply the digit by the power of 2 based on its position and add it to the result

result += d \* Math.pow(2, s.length() - 1 - i);

}

// Return the result

return result;

}

public static void main(String[] args) {

// Create a scanner object to get user input

Scanner sc = new Scanner(System.in);

// Prompt the user to enter a binary string

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

// Read the input as a string

String input = sc.nextLine();

// Close the scanner

sc.close();

// Check if the input is a valid binary string

if (isValidBinary(input)) {

// Convert the input to a decimal number using the bin2Dec method

int output = bin2Dec(input);

// Display the output

System.out.println("The equivalent decimal number for binary \"" + input + "\" is " + output);

} else {

// Display an error message

System.out.println("Error: Invalid Binary String \"" + input + "\"");

}

}

}

4. You are asked to create an application for registering the details of jobseeker. The requirement is:

Username should always end with **\_job**and there should be atleast minimum of 8 characters to the left of **\_job**. Write a function to validate the same. Return true in case the validation is passed. In case of validation failure return false.

|  |  |
| --- | --- |
| Method Name | validateUserName |
| Method Description | Checks if the username is valid |
| Argument | String userName |
| Return Type | boolean |
| Logic | Checks if the username ends with \_job and  contains at least 8 characters to the left of \_job. If valid return true. Else return false. |

// A function to check if the username is valid

public static boolean validateUserName(String userName) {

// Check if the username ends with \_job

if (userName.endsWith("\_job")) {

// Get the length of the username

int len = userName.length();

// Check if there are at least 8 characters to the left of \_job

if (len >= 12) {

// Return true as the validation is passed

return true;

}

}

// Return false as the validation is failed

return false;

}