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 |

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

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"

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. |

```
1. public class Sample_String{
  public static void main(String[] args){
```

```
//String Concatenation
String str1 = "Rock";
String str2 = "Star";
//Method 1 : Using concat
String str3 = str1.concat(str2);
System.out.println(str3);
//Method 2 : Using "+" operator
String str4 = str1 + str2;
System.out.println(str4);
}
}
```

```
public class Sample_String{
  public static void main(String[] args){  //Our sample string for this tutorial
  String str_Sample = "RockStar";
  //Length of a String
  System.out.println("Length of String: " + str_Sample.length());}}
```

```
public class Sample_String{
  public static void main(String[] args){//Character at position
String str_Sample = "RockStar";
System.out.println("Character at position 5: " + str_Sample.charAt(5));
//Index of a given character
System.out.println("Index of character 'S': " + str_Sample.indexOf('S'));}}
```

```
public class Sample_String{
  public static void main(String[] args){//Character at position
String str_Sample = "RockStar";
System.out.println("Character at position 5: " + str_Sample.charAt(5));}}
```

```
public class Sample_String{
  public static void main(String[] args){//Compare to a String
String str_Sample = "RockStar";
  System.out.println("Compare To 'ROCKSTAR': " + str_Sample.compareTo("rockstar"));
  //Compare to - Ignore case
  System.out.println("Compare To 'ROCKSTAR' - Case Ignored: " +
str_Sample.compareToIgnoreCase("ROCKSTAR"));}}
```

```
public class Sample_String{
  public static void main(String[] args){  //Check if ends with a particular sequence
String str_Sample = "RockStar";
  System.out.println("EndsWith character 'r': " + str_Sample.endsWith("r"));}}
```

```
public class Sample_String{
  public static void main(String[] args){//Convert to LowerCase
String str_Sample = "RockStar";
System.out.println("Convert to LowerCase: " + str_Sample.toLowerCase());
//Convert to UpperCase
System.out.println("Convert to UpperCase: " + str_Sample.toUpperCase());}}
```

2. Java provides three classes to represent a sequence of characters: String, StringBuffer, and StringBuilder. The String class is an immutable class whereas StringBuffer and StringBuilder classes are mutable. There are many differences between StringBuffer and StringBuilder. The StringBuilder class is introduced since JDK 1.5.

| | |
|---|---|
| StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |
| StringBuffer was introduced in Java 1.0 | StringBuilder was introduced in Java 1.5 |

Ex:

```
public class ConcatTest{
    public static void main(String[] args){
        long startTime = System.currentTimeMillis();
      StringBuffer sb = new StringBuffer("Java");
      for (int i=0; i<10000; i++){
        sb.append("Tpoint");
        }
      System.out.println("Time taken by StringBuffer: " + (System.currentTimeMillis() - startTime) +
"ms");
      startTime = System.currentTimeMillis();
      StringBuilder sb2 = new StringBuilder("Java");
      for (int i=0; i<10000; i++){
        sb2.append("Tpoint");
      }
      System.out.println("Time taken by StringBuilder: " + (System.currentTimeMillis() - startTime) +
 "ms");
    }
}
    }
```

3

```
package
javaexercises.keyboard;
```

```
import java.util.Scanner;


/**
 *
 * @author User
 */
public class Bin2Dec {

    public static void main(String[] args) {
        Bin2Dec aBin2Dec = new Bin2Dec();
        aBin2Dec.runTest("1011");
        aBin2Dec.runTest("0011");
        aBin2Dec.runTest("1010");
        aBin2Dec.runTest("1234");

//        Scanner in = new Scanner(System.in);
//        System.out.print("\nEnter a Binary string: ");
//        String bin = in.next();
//        aBin2Dec.runTest(bin);
    }

    /**
     * Test.
     *
     * @param binStr
     */
    private void runTest(String binStr)
    {
        if ( ! isBin(binStr)) {
            System.out.printf("Error: Invalid Binary String \"%1$s\"\n",
binStr);
            return;
        }
        System.out.printf("The equivalent decimal number for binary
\"%1$s\" is %2$d\n"
                        , binStr, convertBin2Dec(binStr));
    }

    /**
     * Check if string represent a binary value.
     *
     * @param binStr
     * @return
     */
```

```
    private static boolean isBin(String binStr)
    {
        for(int i = 0; i < binStr.length(); i++)
        {
            if (binStr.charAt(i) == '0') {
                continue;
            }
            if (binStr.charAt(i) == '1') {
                continue;
            }
            return false;
        }
        return true;
    }

    /**
     * Convert bin to decimal.
     *
     * @param  bin
     * @return int
     */
    private int convertBin2Dec(String binStr)
    {
        int number = 0;
        for(int i = 0; i < binStr.length(); i++)
        {
            if (binStr.charAt(i) == '0') {
                continue;
            }
            number += Math.pow(2, (binStr.length() - 1 - i));
        }
        return number;
    }

}
```

4.
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. |