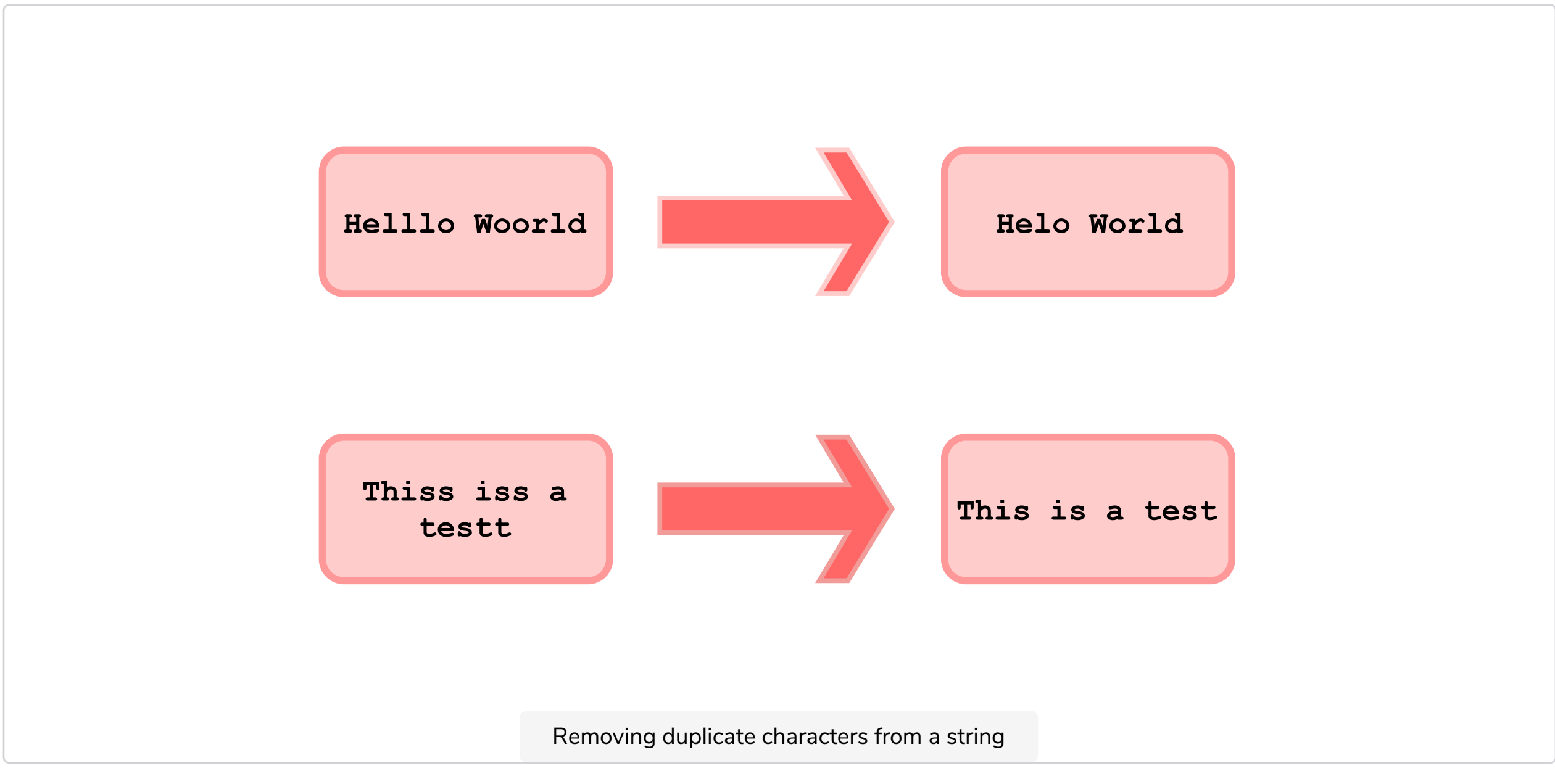
What does removing duplicates mean?

When given a string that has repeating adjacent characters, we only want to keep one of each character. To do this, we must eliminate the repeating characters. The illustration below shows this process.



Removing duplicate characters from a string

Implementing the Code

The code below shows how to do this with recursion. First, let's see the code, and -then we will go on to its explanation.

Try the code by changing the values of `text` to see how it works with other strings.

```
1  class RemoveDuplicatesClass {
2  
3      private static String remDuplicates(String text) {
4          if (text.length() == 1) {
5              return text;
6          }
7  
8          if (text.substring(0,1).equals(text.substring(1,2))) {
9              return remDuplicates(text.substring(1));
10         }
11         else {
12             return text.substring(0,1) + remDuplicates(text.substring(1));
13         }
14     }
15 
16     public static void main( String args[] ) {
17         String input1 = "Helloo";
18         String input2 = "Thiss iiss aa teesstt";
19 
20         System.out.println( "Original string: " + input1);
21 
22         String output = remDuplicates(input1);
23 
24         System.out.println("String after: " + output);
25     }
26 }
```

Run   Save   Reset

Understanding the Code

The recursive code can be broken down into two parts: the recursive method and the main where the method is called.

Driver Method

The recursive method is called within the driver function. Let's first look at what the method does, -from **lines 17 to 24**.

- The `main` method creates two **strings** called `input1` and `input2` which consist of multiple **adjacent repeating characters**.
- The method `remDuplicates` is called with `input2` as an argument.
- The string `output` is displayed to show the changes.

Recursive Method

Now let's examine the recursive method: `remDuplicates`. This is the code segment from **line 4 to line 13** in the snippet above.

Base Case

- The first *if condition* from **line 4 to line 6**; checks if the length of the string `text` is equal to 1. If it is found to be the length of 1 character, the method returns. This is our base case, for the method will terminate, and return to the main method, and no more recursive calls will be made.

Recursive Case

- The method takes in one argument: the string `text` that is to be reversed.
- The `if` condition; from **line 8 to line 10** checks if the character on the `0th` index is equal to the character on the `1st` index. If the condition is true, the recursive method `remDuplicates` is called again with the string that consists of only one character, i.e.- the character from the `1st` index till the end of the string.
- If the above condition evaluates to be false, the character at the `0th` index is to the recursive method call `remDuplicates`. It then takes the string from the `1st` index until the end of that string.

Understanding through a Stack