String

1. length()
2. equal()
3. split()

- toLowerCase()
- toUpperCase()

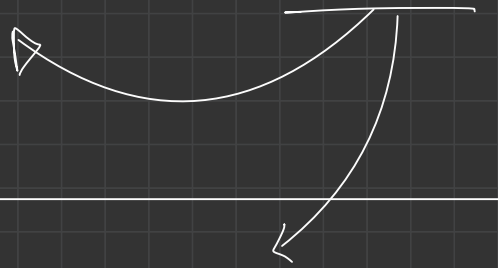
String

What? (Stream)

↳ collection of character

```
String str = "ABC";
String str = new String("ABC");
char[] arr = {'A', 'B', 'C'};
String str2 = new String(arr);
```

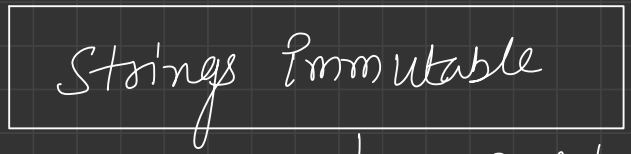
```
String str = "ABC";
char[] arr = str.toCharArray();
```



```
char[] arr = new char[str.length];
for (int i=0; i<str.length(); i++)
{
    arr[i] = str.charAt(i);
}
```

TC: O(N)
SC: O(N)

Are String mutable or immutable?



↳ StringBuilder / Memory diagram

garbage Collector . (Java)

(earlier) → use to collect useless data, and then delete it.

(now) → use to collect imp data, and clear everything else

CPP → doesn't have a garbage collector.

```
void fun(String str)
{
  syso(str);
}
main()
{
  String str = "Harshal";
  fun(str);
}
```

Substrings

String str = "abc" (0 1 2)

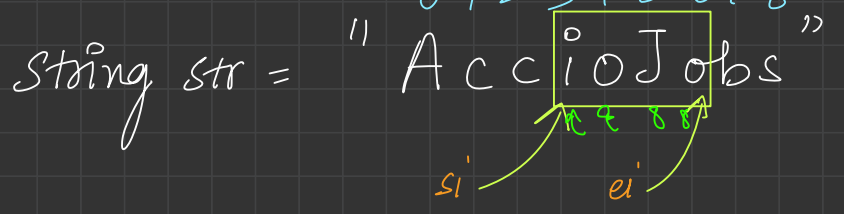
Ans
- a ✓
- ab ✓
- abc ✓
- b ✓
- bc ✓
- c ✓

give me a substring starting from index i and ending at j.

String = "Acci0J0bs" (0 1 2 3 4 5 6 7 8)

i = 3 j = 6

str2 = i0J0



str2 = "ioJo"

```
String str2 = "";
for(int i=si; i<=ei; i++)
    str2 += str.charAt(i);
```

inappropriate in strings

str2 = str.substring(si, ei+1);

(3, 7)

gets the substring of the string b/w

&& & ei.

Palindrome String

↳ same when reversed!

str = "abba"

reverse Str = "abba"

equal → palindrome

method reverse

```
String str2 = Str.reverse();
if (str.equals(str2) = true)
    print (palindrome)
else
    not palind
```

String str = "race, a car!"

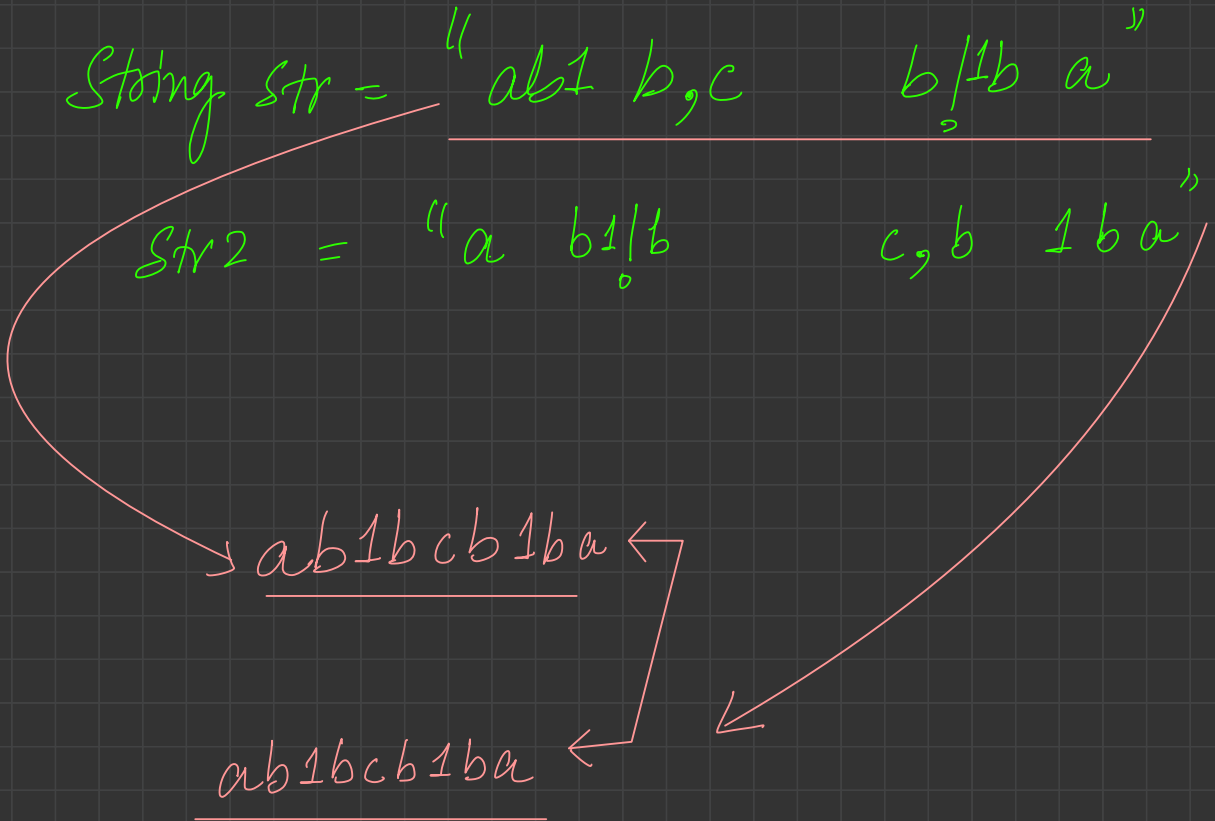
str2 = "racea,ecar"

→ Not Palindrome!

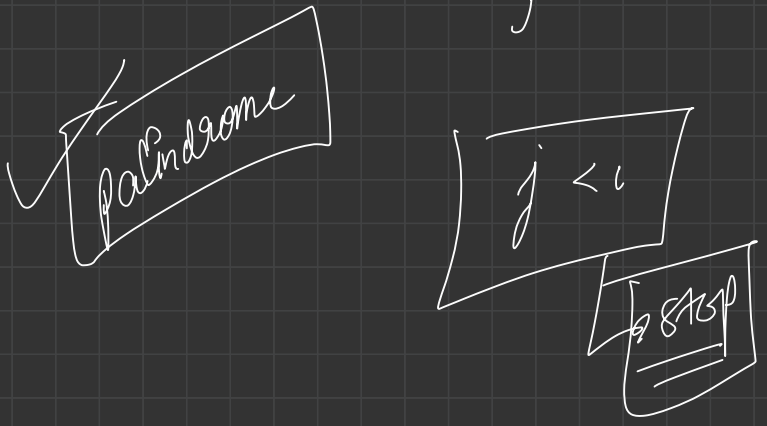
ignore → all special char & extra spaces

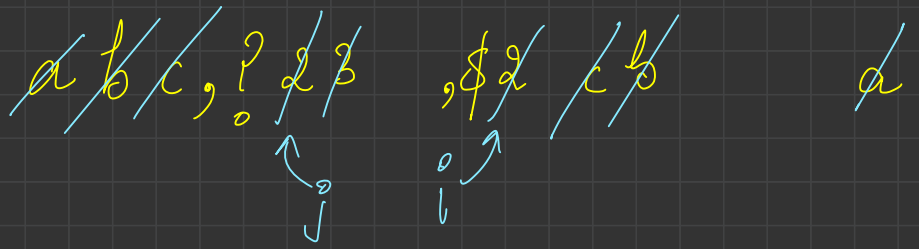
String str = "ab1 b,c b!1b a"

Str2 = "a b1!b c,b 1b a"



String str = "abcba"



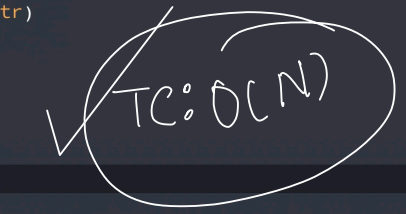


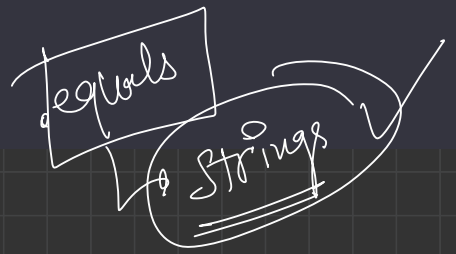
① inc i till you find next alpha Numeric

② dec j till you find prev alpha Numeric

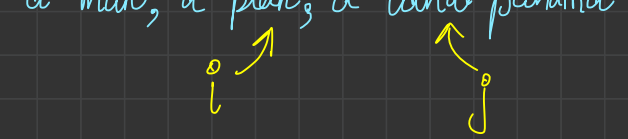
③ Hence palindrome!

```java
static int isPalindrome(String str)
{
    //write code here
    str = str.toLowerCase();

    int i = 0;
    int j = str.length() - 1;

    while (i < j) {
        // increase i till you find a position to be alpha numeric
        while (i < str.length() && isAlphaNumeric(str.charAt(i)) == false) {
            i++;
        }

        while (j >= 0 && isAlphaNumeric(str.charAt(j)) == false) {
            j--;
        }

        if (str.charAt(i) != str.charAt(j)) {
            return 0;
        } else {
            i++;
            j--;
        }
    }

    return 1;
}
```





a man, a plan, a canal Panama



distinct Palindromic Substrings

String = "aababaa"

{
a, aa, aaba, aabab, aababa, aababaa, aabababaa

a, ab, aba, abab, ababa, ababaaa

b, ba, bab, baba, babaa

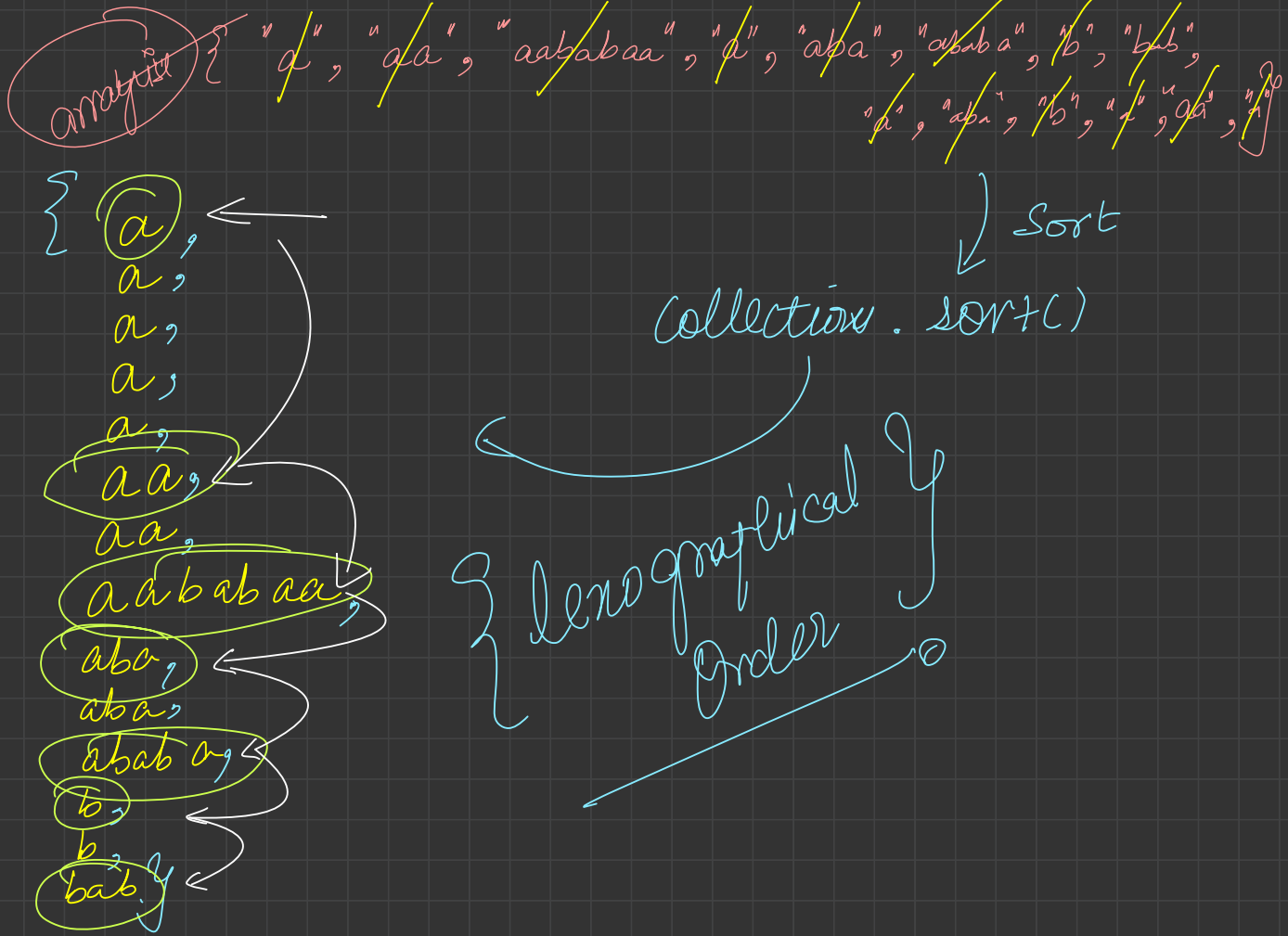
a, ab, aba, abaa

b, ba, baa

a, aa

a
}

output: { "a", "aa", "aababaa", "a", "aba", "ababa", "b", "bab", "a", "aba", "b", "a", "aa", "a" }


Array
{ "a", "aa", "aababaa", "a", "aba", "ababa", "b", "bab",
"a", "aba", "b", "a", "aa", "a" }
Sort
collection . sort()
{ lexographical Order }
{ a, a, a, a, a, aa, aa, aababaa, aba, abaa, abab a, b, b, bab, }

① generate all the substrings

② store all the palindromic substrings in a ArrayList.

③ lexographically sort all pali_ substr's.

using: Collections.sort();

④ filter all distinct values from it.

⑤ print them