Joining Strings

- `String.join("", …)` concatenates strings

```
String t = String.join("", "con", "ca", "ten", "ate");
// "concatenate"
```

- Or an array/list of strings
  - Useful for repeating a string

```
String s = "abc";
String[] arr = new String[3];
for (int i = 0; i < arr.length; i++) { arr[i] = s; }
String repeated = String.join("", arr); // "abcabcabc"
```

Substring

- substring(int startIndex, int endIndex)

```java
String card = "10C";
String power = card.substring(0, 2);
System.out.println(power); // 10
```

- substring(int startIndex)

```java
String text = "My name is John";
String extractWord = text.substring(11);
System.out.println(extractWord); // John
```

Searching (1)

- indexOf() - returns the first match index or -1

```
String fruits = "banana, apple, kiwi, banana, apple";
System.out.println(fruits.indexOf("banana"));    // 0
System.out.println(fruits.indexOf("orange"));    // -1
```

- lastIndexOf() - finds the last occurrence

```
String fruits = "banana, apple, kiwi, banana, apple";
System.out.println(fruits.lastIndexOf("banana")); // 21
System.out.println(fruits.lastIndexOf("orange")); // -1
```

Searching (2)

- `contains()` - checks whether one string contains another

```
String text = "I love fruits.";
System.out.println(text.contains("fruits"));
// true
System.out.println(text.contains("banana"));
// false
```

Splitting

- **Split** a string by a given **pattern**

```
String text = "Hello, john@softuni.bg, you have been
using john@softuni.bg in your registration";
String[] words = text.split(", ");
// words[]: "Hello", "john@softuni.bg","you have been…"
```

- **Split** by **multiple separators**

```
String text = "Hello, I am John.";
String[] words = text.split("[, .]+");
// "Hello", "I", "am", "John"
```

Replacing

- **replace(match, replacement)** - replaces **all** occurrences
  - The result is a new **string** (strings are **immutable**)

```
String text = "Hello, john@softuni.bg, you have been
using john@softuni.bg in your registration.";
String replacedText = text
      .replace("john@softuni.bg", "john@softuni.com");
System.out.println(replacedText);
// Hello, john@softuni.com, you have been using
john@softuni.com in your registration.
```

Solution: Text Filter (1)

```
String[] banWords = sc.nextLine.split(", ");
String text = sc.nextLine();
for (String banWord : banWords) {
  if (text.contains(banWord)) {
    String replacement = repeatStr("*",
     banWord.length());
    text = text.replace(banWord, replacement);
  }
}
System.out.println(text);
```

contains(…) checks if string contains another string

replace() a word with a sequence of asterisks of the same length

Check your solution here: https://judge.softuni.org/Contests/1669/

Solution: Text Filter (2)

```
private static String repeatStr(String str, int length) {
  String replacement = "";
  for (int i = 0; i < length; i++) {
    replacement += str;
  }
  return replacement;
}
```

Check your solution here: https://judge.softuni.org/Contests/1669/

Using StringBuilder Class

- Use the **StringBuilder** to build/modify strings

```
StringBuilder sb = new StringBuilder();
sb.append("Hello, ");
sb.append("John! ");
sb.append("I sent you an email.");
System.out.println(sb.toString());
// Hello, John! I sent you an email.
```

Concatenation vs. StringBuilder (1)

- **Concatenating** strings is a **slow** operation because each iteration **creates** a **new string**

```
System.out.println(new Date());
String text = "";
for (int i = 0; i < 1000000; i++)
  text += "a";
System.out.println(new Date());
```

```
Tue Jul 10 13:57:20 EEST 2018
Tue Jul 10 13:58:07 EEST 2018
```

Concatenation vs. StringBuilder (2)

- Using **StringBuilder**

```
System.out.println(new Date());
StringBuilder text = new
StringBuilder();
for (int i = 0; i < 1000000; i++)
  text.append("a");
System.out.println(new Date());
```

```
Tue Jul 10 14:51:31 EEST 2018
Tue Jul 10 14:51:31 EEST 2018
```

StringBuilder Methods (1)

- append() - appends the string representation of the argument

```
StringBuilder sb = new StringBuilder();
sb.append("Hello Peter, how are you?");
```

- length() - holds the length of the string in the buffer

```
sb.append("Hello Peter, how are you?");
System.out.println(sb.length()); // 25
```

- setLength(0) - removes all characters

StringBuilder Methods (2)

- `charAt(int index)` - returns char on index

```
StringBuilder sb = new StringBuilder();
sb.append("Hello Peter, how are you?");
System.out.println(sb.charAt(1)); // e
```

- `insert(int index, String str)` – inserts a string at the specified character position

```
sb.insert(11, " Ivanov");
System.out.println(sb);
// Hello Peter Ivanov, how are you?
```

StringBuilder Methods (3)

- `replace(int startIndex, int endIndex, String str)` - replaces the chars in a substring

```
sb.append("Hello Peter, how are you?");
sb.replace(6, 11, "George");
```

- `toString()` - converts the value of this instance to a String

```
String text = sb.toString();
System.out.println(text);
// Hello George, how are you?
```