Java training

`String`, `StringBuffer`, `StringBuilder`, `StringTokenizer`

© Bogdan Solga

Session overview

- The **`String`** class
- String operations
- `StringTokenizer`, `StringBuffer` and `StringBuilder`
- **Hands-on** - using strings

`String` class

- Representation of sequences of characters (words, sentences, …)
- **Eleven** constructors, + countless utilities for assembling & proc. Strings
- ***Immutable*** - their value cannot be changed
  - All the operations on a String variable will return a new String
- Final class → cannot be extended
- Contains various methods for manipulating strings
  - `concat()`
  - `split()`
  - `substring()`
  - …

`String` objects samples

```
String hello = “Hello, everyone!”;

char[] anArrayOfChars = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’};

String builtFromChars = new String(anArrayOfChars);

String subStringed = “some string”.substring(4); // returns “some”
```

String operations

- Getting the length: `int length = “holiday”.length(); // 7`
- Splitting by a char: `String[] splitted = “north,south”.split(“,”); // {“north”, “south”}`

- Sub-string: `String aPart = “something”.substring(4); // “some”`
- Concatenate: `String concat = “some”.concat(“where”); // “somewhere”`
  `String concat = “some” + “where”; // not recommended (TBD)`

- Character at: `char char = “some”.charAt(3); // ‘e’`
- Equals: `boolean equal = “some”.equals(“thing”); // false`
- Equals (case ignored): `boolean equal = “it”.equalsIgnoreCase(“IT”); // true`

- Index of: `int index = “some”.indexOf(‘o’); // 1`
- Last index of: `int lastIndex = “somewhere”.lastIndexOf(‘e’); // 8`

Object's 'toString()' method

- The `Object` class contains a '`toString()`' method
  - By default, it returns the full class name + '@' + the class hash code
  - It is invoked implicitly, even if the '`toString()`' method is not explicitly called
- Inherited in all the classes → can be overriden
- By overriding it, it can output the relevant information from that object
- Example:

```
Product product = new Product();
System.out.println(product);
// sample output: Product@745f
```

```
@Override
public String toString() {
    return this.id + " - " + this.name;
}

System.out.println(product);    // 23 - Samsung S7
```

StringBuilder, StringBuffer

- Helper classes used for building / joining strings
  - Main difference - `StringBuffer` is thread-safe, `StringBuilder` is not
- They contain methods for string processing / appending
- The resulted string - returned by a call to the '`toString()`' method

```
StringBuilder builder = new StringBuilder(“somewhere, ”);
builder.append(“over ”).append(“the ”).append(“rainbow”);
                                        // recommended String concatenation
System.out.println(builder.toString());
                                        // “somewhere, over the rainbow”
```

StringTokenizer

- Utility class used to tokenize a string → split it by some defined tokens
  - By default - it splits by ‘ ‘ (space)
- Example:

```
StringTokenizer tokenizer = new StringTokenizer(“I want a holiday”);
                                                    // splits by “ “

StringTokenizer tokenizer = new StringTokenizer(“Let’s,play”, “,”);
                                                    // splits by “,”

while (tokenizer.hasMoreElements()) {
    System.out.println(tokenizer.nextElement());
}
```

Q&A session

1. You ask, I answer
2. I ask, you answer

Hands-on

- Familiarize with the `String` class methods
  - Perform splitting, joining, size checks, ...
- Override the '`toString()`' method in several classes
  - Create a few `Product` related classes and implement toString in them
  - Create a Section class, which holds multiple Products, implement toString()
- Build new `Strings` using `StringBuilder` or StringBuffer
  - Familiarize yourself with the methods from the two classes
- Tokenize a `String` using the `StringTokenizer` class
  - Choose your split String