Java String and Text Processing

Lecturer: **Nataliia Stulova**
Teaching assistant: **Mohammadreza Hazirprasand**

Software Composition Group
University of Bern
14 October 2020

UNIVERSITÄT BERN

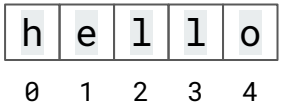
MASTER IN COMPUTER SCIENCE
OF THE UNIVERSITIES OF BERN, NEUCHÂTEL AND FRIBOURG

Software Skills Lab

Strings

java.lang.String

The `String` class represents character strings.



- Java strings are objects
- Java strings are **immutable** - once created, cannot be changed
- BUT you can query and search for substrings

```
String s = “hello”;
char c = s.charAt(1);          // c == ‘e’
int i = s.indexOf(l);          // i == ‘2’
String sub = s.substring(1,3); // sub == “el”
```

Special characters

`\'`	Single quotes. Only need to escape if a string is wrapped in Single quote as well.
`\t`	Tab
`\b`	Backspace
`\r`	Return
`\f`	Formfeed
`\n`	Newline

Pretty printing

- control how non-string values appear in a string: decimal point digits number, time and date, indentation
- useful when you work with templated text messages

```
String fs;
fs = String.format("The value of the float variable is " +
                   "%f, while the value of the integer " +
                   "variable is %d, and the string " +
                   "is %s", floatVar, intVar, stringVar);
System.out.println(fs);
```

more examples: https://www.javatpoint.com/java-string-format

Regular Expressions

Regular expressions

A regular expression (shortened as regex, regexp) is a sequence of characters that define a ***search pattern***. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

Use: in search engines, search and replace dialogs of word processors and text editors, in text processing utilities such as `sed` and `awk` and in lexical analysis.

java.util.regex API

There are many different flavors to choose from, such as `grep`, `Perl`, `Tcl`, `Python`, `PHP`, and `awk`. The regular expression syntax in the `java.util.regex` API is most similar to that found in `Perl`.

The java.util.regex package primarily consists of three classes:

- `Pattern`,
- `Matcher`, and
- `PatternSyntaxException`.

This lecture covers the basics of https://docs.oracle.com/javase/tutorial/essential/regex/intro.html

regex: character classes

`[abc]`	a, b, or c (simple class): `[bc]at` matches `bat` and `cat`
`[^abc]`	Any character except a, b, or c (negation): `[^r]at` matches `fat` but not `rat`
`[a-zA-Z]`	a through z, or A through Z, inclusive (range)
`[a-d[m-p]]`	a through d, or m through p: `[a-dm-p]` (union)
`[a-z&&[def]]`	d, e, or f (intersection)
`[a-z&&[^bc]]`	a through z, except for b and c: `[ad-z]` (subtraction)
`[a-z&&[^m-p]]`	a through z, and not m through p: `[a-lq-z]` (subtraction)

regex: predefined character classes

`.`	Any character (may or may not match line terminators)
`\d`	A digit: `[0-9]`
`\D`	A non-digit: `[^0-9]`
`\s`	A whitespace character: `[ \t\n\x0B\f\r]`
`\S`	A non-whitespace character: `[^\s]`
`\w`	A word character: `[a-zA-Z_0-9]`
`\W`	A non-word character: `[^\w]`

regex: quantifiers

`X?`	X, once or not at all
`X*`	X, zero or more times
`X+`	X, one or more times
`X{n}`	X, exactly n times
`X{n,}`	X, at least n times
`X{n,m}`	X, at least n but not more than m times

regex: boundary matches

`^`	The beginning of a line
`$`	The end of a line
`\b`	A word boundary
`\B`	A non-word boundary
`\A`	The beginning of the input
`\G`	The end of the previous match
`\Z`	The end of the input but for the final terminator, if any
`\z`	The end of the input

java.util.regex API use

```
String regex = "a{2}+";
Pattern pattern = Pattern.compile(regex);

String text = "aaaabbbba";
Matcher matcher = pattern.matcher(text);

while (matcher.find()) {
     System.out.println(
          String.format(
               "substring found at index [%d,%d]: %s",
               matcher.start(),
               matcher.end(),
               matcher.group()));
}
```

regular expression string

text to search the pattern in

java.util.regex API use

```
String regex = "a{2}+";
Pattern pattern = Pattern.compile(regex);

String text = "aaaabbbba";
Matcher matcher = pattern.matcher(text);

while (matcher.find()) {
     System.out.println(
          String.format(
               "substring found at index [%d,%d]: %s",
               matcher.start(),
               matcher.end(),
               matcher.group()));
}
```

regular expression string

text to search the pattern in

```
substring found at index [0,2]: aa
substring found at index [2,4]: aa
```

Practice

Exercise: regular expression patterns

write a `WordFilter.java` class with 5 static methods:

- `wordsAtLineBegin()` - find all words at line beginnings
- `wordsOfLength(int len)` - find all words of length len (e.g., 8 characters)
- `wordsAllCaps()` - find all words that consist of capital letters
- `wordsFirstCapital()` - find all words that start with a capital letter
- `wordsInBrackets()` - find all sub-sentences that are enclosed in () brackets

I/O

Use the example string (next slide) as input. Print results of each search to `System.out`.

Exercise: sample text

```
String inputText = "What separates API specifications from a programming guide are examples,\n"
            + "definitions of common programming terms, certain conceptual overviews (such as\n"
            + "metaphors), and descriptions of implementation bugs and workarounds. There is no\n"
            + "dispute that these contribute to a developer's understanding and help a\n"
            + "developer write reliable applications more quickly. However, because these do\n"
            + "not contain API \"assertions\", they are not necessary in an API specification.\n"
            + "You can include any or all of this information in documentation comments (and\n"
            + "can include custom tags, handled by a custom doclet, to facilitate it). At Java\n"
            + "Software, we consciously do not include this level of documentation in doc\n"
            + "comments, and instead include either links to this information (links to the\n"
            + "Java Tutorial and list of changes) or include this information in the same\n"
            + "documentation download bundle as the API spec -- the JDK documentation bundle\n"
            + "includes the API specs as well as demos, examples, and programming guides.";
```

- taken from https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html
- use as the input text