Java training

Regular expressions, pattern matching

Session overview

- Regular expressions - patterns, matchers
- Hands-on - using regular expressions
- Exercise - if (enoughTime) building a `ValidationService` for our project

Regular expression

- Defines **a search pattern for strings**
  - The abbreviation for regular expression - **regex**
- The search pattern:
  - Can be:
    - A simple character
    - A fixed string
    - A complex expression containing special characters describing the pattern.
  - May match one or several times, or not at all

Examples & quantifiers

```
“a”.matches(“.”);             // . - matches any character

“.”.matches(“\\.”);           // \\. - matches the dot (‘.’)

“7”.matches(“\\d”);           // \d - a digit

“23”.matches(“\\d+”);         // \d+ - any number of digits

“b”.matches(“[abc]”);         // [] - matching any of the enclosed chars

“c”.matches(“[a-d1-9]”);      // matches any char between a-d and any number 1-9
```

https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

Usefulness

- Used to search, edit, *validate* and manipulate text
- Analyzing or modifying a text with a regex - '*applying the regex* to the text'
- The pattern (defined by the regex) is applied on the text **from left to right**
- Once a source character has been used in a match, it cannot be reused
  - Example: the regex 'eve' will match eveveveve only two times (eve_eve__).

Meta characters / character classes

\d	Any digit, short for [0-9]	
\D	A non-digit, short for [^0-9]	// ^ → negate
\s	A whitespace character, short for [\t\n\x0b\r\f]	
\S	A non-whitespace character, short for [^\s]	
\S+	Several non-whitespace characters	
\w	A word character, short for [a-zA-Z_0-9]	
\W	A non-word character [^\w]	

Quantifiers

Expression	Description	Examples
*	Occurs zero or more times, is short for {0,}	X* - finds no or several letter X * - finds any character sequence
+	Occurs one or more times, is short for {1,}	X+ - finds one or several letter X
?	Occurs no or one times, is short for {0,1}	X? - finds no or exactly one letter X
{X}	Occurs X number of times, {} describes the order of the preceding sequence	\d{3} searches for 3 digits, .{3} for any char sequence of length 3
{X, Y}	Occurs between X and Y times	\d{1,4} - the digit must occur at least once and at a maximum of four.

Capturing groups - example

- **Regex** for DD/MM/YYYY date validation: [0-9]{2}/[0-9]{2}/[0-9]{4}
- **Grouping** the expression - becomes: ([0-9]{2})/([0-9]{2})/([0-9]{4})
- **Groups**:
  - Delimited by ()
  - Numbered from 1; 0 is the entire matched string

- Extracting / capturing the groups:

```
Pattern pattern = Pattern.compile("([0-9]{2})/([0-9]{2})/([0-9]{4})");
Matcher matcher = pattern.matcher(“23/09/2016”);
if (matcher.matches()) {
    int day   = Integer.parseInt(matcher.group(1));
    int month = Integer.parseInt(matcher.group(2));
}
```

`Pattern` and `Matcher` classes

- **`Pattern`** - compiled representation of a regular expression
    - Provides static **`compile()`** methods for building `Pattern` objects
    - They accept a regular expression as the first argument

- **`Matcher`**:
    - Engine that:
        - Interprets the pattern
        - Performs match operations against an input string
    - No public constructors - obtained via the **matcher()** method on a `Pattern` object
- Used when repeated pattern matchings are needed (expensive operation)

Hands-on

- Test the main character classes and quantifiers
- Correct the date pattern regex, so that it accepts proper values for the day, month and year

The email pattern - http://www.ex-parrot.com/pdw/Mail-RFC822-Address.html

Exercise

Implement a `ValidationService`, which validates the following values for a signup form:

- The first name and last name - each one must:
  - Be at least 3 chars long
  - Begin with an uppercase character
- The birth date - must be in the format DD/MM/YYYY
  - The shop won't accept users younger than 18 years
- The email must have:
  - At least 4 characters, @, a name of at least 3 chars
  - A domain name between 2 and 6 characters

The service will be used from a `ValidationMain` class, for now