

CS2003 Regular Expressions

Remember definition of protocol from week 1



- Definitions of message formats:
 - data structures.
 - application programming interface (API).
 - messages for data, signalling & management.
- A set of rules:
 - algorithm.
 - finite state machine (FSM).
- Error handling:
 - part of the FSM, API, special messages.



And our protocol from Practical 1

```
HELLO
```

HELLO

```
ADDRESS: <something that looks like an address>
```

```
ADDRESS: <something that looks like an address>
```

BYE

BYE

CS2003 2020/21 3





ADDRESS: <something that looks like an address>

 How do we check that <something> looks like an IP address?

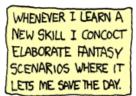
CS2003 2020/21 4

Matching and checking string patterns the Andrews

- Regular expressions (regex for short) are used in many languages.
 - match a pattern of characters.
 - specify the pattern to be matched.
- Very popular in UNIX text-processing languages and utilities.
- Java also supports regular expressions.
- Beware: regular expressions work slightly differently in different languages
 - There are standards (POSIX, PCRE) but there are more than one!
- Extremely useful to learn, as a lot of data processing involves manipulation of strings:
 - pattern matching.
 - searching.
 - search and replace.
- For example, can be used to check and parse simple, text-based application-level protocol messages.



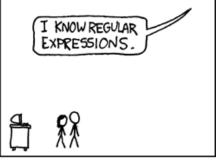
Regex is very powerful!



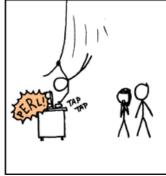














https://xkcd.com/208/

Regular Expressions in Java



Create a regular expression as a string:

```
String r = "FindMe";
```

Use Pattern "compile" regex:

```
Pattern p = Pattern.compile(r);
```

Use Matcher to apply Pattern to a string/input:

```
Matcher m = p.matcher("Can you FindMe in here?");
if (m.find()) { /* found "FindMe" */ }
```

CS2003/Examples/wk05/regex/FindMe.java

Meta-characters have special meaning, e.g. :

```
- ^ beginning of a line
- $ end of a line
- . * + ? \ / [ ] { } ( ) plus others
- \M escape for meta-character M
```

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html



Regex classes in Java

- Commonly used character sets/classes are denoted by a special escape sequence, e.g.:
 - \d same as [0-9]
 \D same as [^0-9]
 \w word character, same as [A-Za-z0-9_]
 \b word boundary
- You can define your own classes, e.g.:

```
- [abc] match 'a', 'b', or 'c'
- [^abc] match anything but 'a', 'b', or 'c'
- [a-z] match characters in the range 'a' - 'z'
```

 In Java, use "\\" in pattern string so that a `\' is correctly interpreted!, e.g. "\\d", "\\w", etc.



Repetition

Repetition qualifiers modify pattern match, e.g.:

```
"\\d{2,5}" between 2 and 5 digits
```

"\\d $\{4,\}$ " 4 or more digits

"\\d{2}\\w?" 2 digits followed by zero or more of [a-zA-Z_]

"\\^\\d+" '^' followed by at least one digit

"\\s+glob\\s+" 'glob' with spaces either side

CS2003/Examples/CS2003-Examples-wk05/regex/RegexClasses.java



Flags

Flags modify the matching behaviour, e.g.:

i "(?i:X)" case insensitive

m "(?m:X)" multi-line

d "(?dm:X)" UNIX mode line terminator only ("\n") (some text files use "\r" and/or "\m")

(also other flags)



Named capture group [1]

- Define a group within the regex:
 - A part of the overall pattern.
- If the group is matched, assign the match to a name (a reference to match).
- Very useful for parsing a complex pattern, and 'splitting' the matched parts.
- The use of brackets "(" and ")" denote a group:
 - "(?<myName>X)" myName is assigned to a match for X
 - (groups can be used without name capture, also)



Named capture groups [2]

Named groups can be used with user input, e.g.:

"(?<**year**>\\b\\d{4}\\b)"

year -> 4-digit number

"(?<number5>\\b\\d{5}\\b)"

number5 -> 5-digit number

"(?<tla>\\b[A-Z]{3}\\b)"

tla -> 3-uppercase-letter word (abbreviation)

"(?<d2word>\\b\\d{2}[a-zA-Z-_]{1,})"

d2word -> a word that starts with 2 digits

"($?<uid>\b[a-z]{1,5}\d+\b]$ "

uid -> 1 to 5 letters, 1 or more digits

CS2003/Examples/CS2003-Examples-wk05/regex/RegexGroups.java



Regex Examples in Java

- CS2003/Examples/CS2003-Examples-wk05/regex/:
 - FindMe.java
 - RegexClasses.java
 - RegexCSV.java
 - RegexGroups.java
 - IPNetMask.java



Further reading

- https://docs.oracle.com/javase/tutorial/essential/regex/
- https://www.regexplanet.com/
- http://www.regular-expressions.info/tutorial.html