Character Classes: Ranges

- `[nvj]` matches any character that is either **n**, **v** or **j**

```
node.js v0.12.2
```

- `[^abc]` - matches any character that is **not** **a**, **b** or **c**

```
Abraham
```

- `[0-9]` - character range matches any digit from **0** to **9**

```
John is 8 years old.
```

Predefined Classes

- **\w** - matches any **word character** (a-z, A-Z, 0-9, _)
- **\W** - matches any **non-word character** (the opposite of \w)
- **\s** - matches any **white-space** character
- **\S** - matches any **non-white-space** character (opposite of \s)
- **\d** - matches any **decimal digit** (0-9)
- **\D** - matches any **non-decimal character** (the opposite of \d)

Quantifiers

- * - matches the previous element zero or more times

```
\+\d*   +359885976002 a+b
```

- + - matches the previous element one or more times

```
\+\d+   +359885976002 a+b
```

- ? - matches the previous element zero or one time

```
\+\d?   +359885976002 a+b
```

- {3} - matches the previous element exactly 3 times

```
\+\d{3}   +359885976002 a+b
```

Grouping Constructs

- `(subexpression)` - captures the matched subexpression as numbered group

```
\d{2}-(\w{3})-\d{4}
```

```
22-Jan-2015
```

- `(?:subexpression)` - defines a non-capturing group

```
^(?:Hi|hello),\s*(\w+)$
```

```
Hi, Peter
```

- `(?<name>subexpression)` - defines a named capturing group

```
(?<day>\d{2})-(?<month>\w{3})-
(?<year>\d{4})
```

```
22-Jan-2015
```

Backreferences Match Previous Groups

- \number - matches the value of a numbered capture group

```
<(\w+)[^>]*>.*?<\/\1>
```

```
<b>Regular Expressions</b> are cool!
<p>I am a paragraph</p> … some text after
Hello, <div>I am a<code>DIV</code></div>!
<span>Hello, I am Span</span>
<a href="https://softuni.bg/">SoftUni</a>
```

Regex in Java

- Regex in Java library
  - `java.util.regex.Pattern`
  - `java.util.regex.Matcher`

```
Pattern pattern = Pattern.compile("a*b");
Matcher matcher = pattern.matcher("aaaab");

boolean match = matcher.find();
String matchText = matcher.group();
```

Searches for the next match

Gets the matched text

Checking for a Single Match

- `find()` - gets the first pattern match

```
String text = "Andy: 123";
String pattern = "([A-Z][a-z]+): (?<number>\\d+)";

Pattern regex = Pattern.compile(pattern);
Matcher matcher = regex.matcher(text);

System.out.println(matcher.find());           // true
System.out.println(matcher.group());          // Andy: 123
System.out.println(matcher.group(0));         // Andy: 123
System.out.println(matcher.group(1));         // Andy
System.out.println(matcher.group(2));         // 123
System.out.println(matcher.group("number")); // 123
```

+ - Matches the element one or more times

Replacing with Regex

- To replace **every**/**first** subsequence of the input sequence that matches the pattern with the given replacement string
  - `replaceAll(String replacement)`
  - `replaceFirst(String replacement)`

```
String regex = "[A-Za-z]+";
String string = "Hello Java";
Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(string);
String res = matcher.replaceAll("hi");    // hi hi
String res2 = matcher.replaceFirst("hi"); // hi Java
```

Splitting with Regex

- `split(String pattern)` - splits the text by the pattern
  - Returns `String[]`

```
String text = "1   2 3      4";
String pattern = "\\s+";

String[] tokens = text.split(pattern);
```

Matches whitespaces

tokens = {"1", "2", "3", "4"}

Solution: Match Full Names

```
String listOfNames = reader.readLine();

String regex = "\\b[A-Z][a-z]+ [A-Z][a-z]+";
Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(listOfNames);

while (matcher.find()) {
    System.out.print(matcher.group() + " ");
}
```

Check your solution here: https://judge.softuni.org/Contests/1672/

Solution: Match Dates

```
String input = reader.readLine();

String regex =
"\\b(?<day>\\d{2})(\\.|\\/|\\-)(?<month>[A-Z][a-
z]{2})\\2(?<year>\\d{4})\\b";

Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(dates);

while (matcher.find()) {
      System.out.println(String.format("Day: %s, Month: %s, Year: %s",
      matcher.group("day"), matcher.group("month"),
      matcher.group("year")));
}
```

Check your solution here: https://judge.softuni.org/Contests/1672/