# Lab: Regular Expressions

Problems for exercises and homework for the ["Java Advanced" course @ SoftUni](https://softuni.bg/courses/java-fundamentals).

You can check your solutions here: <https://judge.softuni.bg/Contests/Practice/Index/458#0>.

# Part I: Basic Regex

## Match Count

Find the **count of occurrences** of a word in a given text using regex.

The **word** to search for is given on the **first line**.

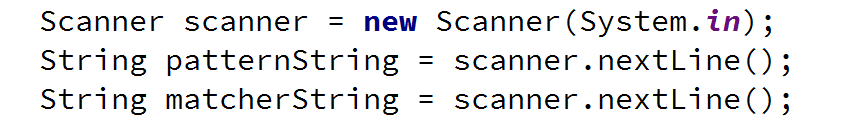
The **text** to search in is given on the **second line**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| regex  A regular expression, regex or regexp (sometimes called a rational expression) is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. | 2 |
| re  A regular expression, regex or regexp (sometimes called a rational expression) is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. | 6 |

### Hints

* Read the input using a Scanner or a BufferedReader:



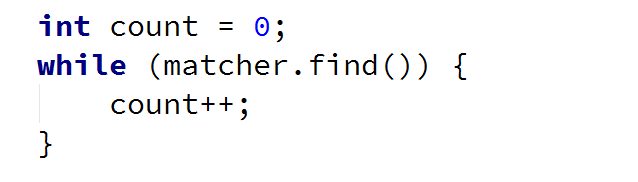
* Compile the pattern:



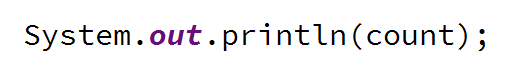
* Create a Matcher object:



* Count the occurrences:



* Finally, print the result:



## Vowel Count

Find the **count** of **all vowels** in a given **text** using a regex.

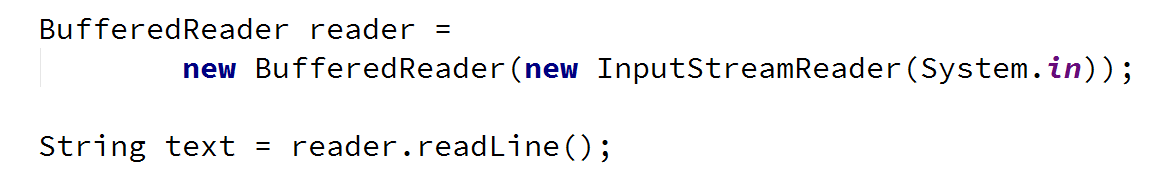
The vowels that you should be looking for are **upper** and **lower** **a**, **e**, **i**, **o**, **u** and **y**.

### Examples

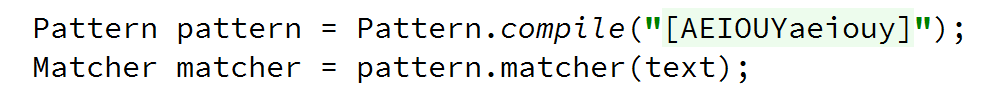
|  |  |
| --- | --- |
| **Input** | **Output** |
| Abraham Lincoln | Vowels: 5 |
| In 1519 Leonardo da Vinci died at the age of 67. | Vowels: 15 |
| n vwls. | Vowels: 0 |

### Hints

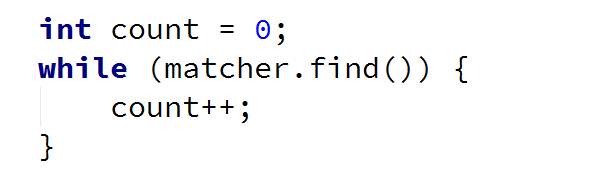
* Read the input using a Scanner or a BufferedReader:



* Compile the pattern and create a Matcher object:



* Count the occurrences:



* Finally, print the result:



## Non-Digit Count

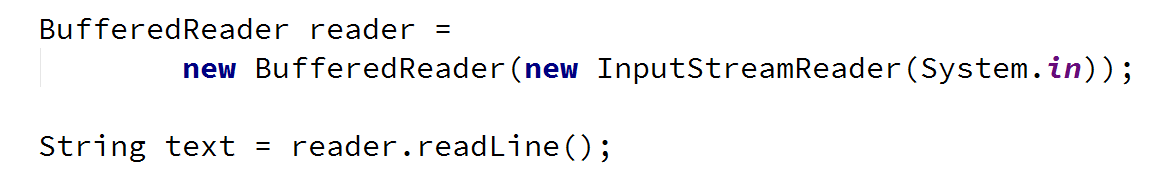
Find the count of all **non-digit characters** in a given text using regex.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Abraham Lincoln | Non-digits: 15 |
| In 1519 Leonardo da Vinci died at the age of 67. | Non-digits: 42 |
| 0 | Non-digits: 0 |

### Hints

* Read the input using a Scanner or a BufferedReader:



* Compile the pattern:



* Count the occurrences and print the result

## Extract Integer Numbers

Extract all **integer numbers** from a given **text** using regex.

Ignore number signs or decimal separators (See the examples below).

If there are **no numbers**, don't print anything.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| In 1519 Leonardo da Vinci died at the age of 67. | 1519  67 |
| Decimals: 7.22, 3.14. Negative -25. | 7  22  3  14  25 |
| No integers. | *(no output)* |

### Hints

* Use regex character classes [] and one of the regex quantifiers: +, \* or ?.

## Extract Tags

Extract all **tags** from a given HTML using regex.

Read lines until you get the **"END"** command.

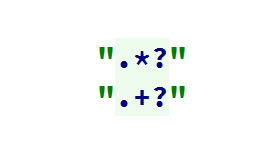
If there are **no tags**, don’t print anything.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Title</title>  </head>  </html>  END | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>  </title>  </head>  </html> |
| No tags.  END | *(no output)* |

### Hints

* Use the special character dot "." and one of the regex quantifiers **made lazy**:



* Design your own regex to get a complete solution

# Part II: Regex Constructs

## Valid Usernames

**Scan** through the lines for **valid usernames**.

A valid username:

* Has **length** between 3 and 16 characters
* **Contains** only letters, numbers, hyphens and underscores
* Has **no redundant symbols** before, after or in between

Read lines until you get the **"END"** command.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| sh  too\_long\_username  !lleg@l ch@rs  jeff\_butt  END | invalid  invalid  invalid  valid |
| END | *(no output)* |

### Hints

* Use character classes [], quantifiers {} and anchors ^$

## Valid Time

Scan through the lines for valid times.

A **valid time**:

* Is **in the interval** 12:00:00 AM to 11:59:59 PM
* Has **no redundant symbols** before, after or in between

Read lines until you get the **"END"** command.

### Examples

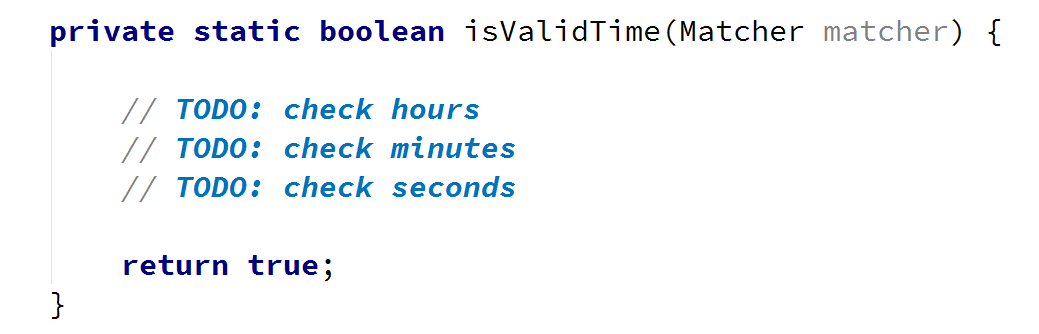
|  |  |
| --- | --- |
| **Input** | **Output** |
| 12:33:24 AM  33:12:11 PM  inv 23:52:34 AM  00:13:23 PM  9:3:12 АМ  END | valid  invalid  invalid  invalid  invalid |
| END | *(no output)* |

### Hints

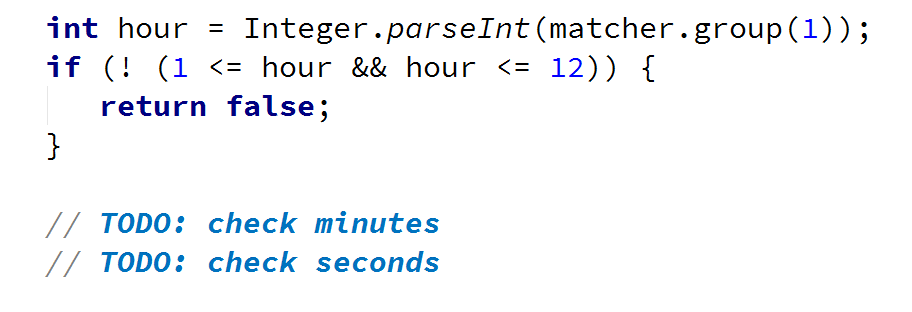
* Create a **regex** that matches **three pairs of digits** separated by ":" and followed by AM or PM, **using groups** to extract the hours, minutes and seconds:



* Create a method that tests if the match has valid hour, minute or second



* The method should **extract each pair of digits** and test if it is a **valid** hour, minute or second
  1. **Hours** should be in the range [1..12]
  2. **Minutes** should be in the range [0..59]
  3. **Seconds** should be in the range [0..59]



* If there is a match, call the method

## Extract Quotations

**Extract** all **quotations** from a given text. The text will be on a single line.

A **valid quotation** should:

* Start and end with either single quotes (**valid: 'quotation'**) or double quotes (**valid: "quotation"**)
* Not have mixed quotes (**invalid: 'quotation"**)

**Skip nested** quotations.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <a href='/' id="home">Home</a><a class="selected"</a><a href = '/forum'> | /  home  selected  /forum |
| <a href='/' id='home"> | / |
| Skip "nested 'quotes'" | nested 'quotes' |
| No quotes. | *(no output)* |

### Hints

* Use grouping and backreference constructs

