**Lab: Regular Expressions C#**

Problems for in-class lab for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/2363/csharp-fundamentals-may-2019)  
You can check your solutions in [Judge](https://judge.softuni.bg/Contests/1667/)

* **Match Full Name**

Write a C# Program to **match full names** from a list of names and **print** them on the console.

**Writing the Regular Expression**

First, write a regular expression to match a valid full name, according to these conditions:

* A valid full name has the following characteristics:
* It consists of **two words**.
* Each word **starts** with a **capital letter**.
* After the first letter, it **only contains lowercase letters afterwards**.
* **Each** of the **two words** should be **at least two letters long**.
* The **two words** are **separated** by a **single space**.

To help you out, we've outlined several steps:

* Use an online regex tester like <https://regex101.com/>
* Check out how to use **character sets** (denoted with square brackets - "**[]**")
* Specify that you want **two words** with a space between them (the **space character ' '**, and **not** any whitespace symbol)
* For each word, specify that it should begin with an uppercase letter using a **character set**. The desired characters are in a range – **from** ‘**A**’ **to** ‘**Z**’.
* For each word, specify that what follows the first letter are only **lowercase letters**, one or more – use another character set and the correct **quantifier**.
* To prevent capturing of letters across new lines, put "**\b**" at the beginning and at the end of your regex. This will ensure that what precedes and what follows the match is a word boundary (like a new line).

In order to check your RegEx, use these values for reference (paste all of them in the **Test String** field):

|  |  |
| --- | --- |
| **Match ALL of these** | **Match NONE of these** |
| Ivan Ivanov | ivan ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Ivan IvAnov, Ivan Ivanov |

By the end, the matches should look something like this:



After you’ve constructed your regular expression, it’s time to write the solution in C#.

**Implementing the Solution in C#**

Create a new C# project and copy your **regular expression** into a **string** variable:



Note: It’s usually a good idea to use a **verbatim string** (@in front of the string literal) to store **regular expressions**, since characters like the backslash “**\**” can clash with **string escaping**.

Now, it’s time to **read the input** and **extract all the matches** from it. For this, we can use the **MatchCollection** class:



After we extract all the matches, we need to **iterate** over the **MatchCollection** and **print** every match that we found:



**Examples**

|  |
| --- |
| **Input** |
| Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov, Ivan Ivanov |
| **Output** |
| Ivan Ivanov Test Testov |

* **Match Phone Number**

Write a regular expression to match a **valid phone number** from **Sofia**. After you find all **valid phones**, **print** them on the console, separated by a **comma and a space** “**,** ”.

**Compose the Regular Expression**

A valid number has the following characteristics:

* It starts with "**+359**"
* Then, it is followed by the area code (always **2**)
* After that, it’s followed by the **number** itself:
* The number consists of **7 digits** (separated in **two** **groups** of **3** and **4** **digits** respectively).
* The different **parts** are **separated** by **either a space or a hyphen** ('**-**').

You can use the following RegEx properties to **help** with the matching:

* Use **quantifiers** to match a **specific number** of **digits**
* Use a **capturing group** to make sure the delimiter is **only one of the allowed characters** **(space or hyphen)** and **not** a **combination** of both (e.g. **+359 2-111 111** has **mixed delimiters**, it is **invalid**). Use a **group backreference** to achieve this.
* Add a **word boundary** at the **end** of the match to avoid **partial matches** (the last example on the right-hand side).
* Ensure that before the **'+'** sign there is either a **space** or the **beginning of the string**.

You can use the following table of values to test your RegEx against:

|  |  |
| --- | --- |
| **Match ALL of these** | **Match NONE of these** |
| +359 2 222 2222  +359-2-222-2222 | 359-2-222-2222, +359/2/222/2222, +359-2 222 2222  +359 2-222-2222, +359-2-222-222, +359-2-222-22222 |

**Implement the Solution in C#**

Now it’s time to write the solution, so let’s start writing!

First, just like in the previous problem, put your RegEx in a variable:



After that, let’s make a **MatchCollection** for our matches:



Let’s try to print **all the matches**, using only a **single line** **of code**. Since **MatchCollection** is, as its name suggests, a **collection**, we can use **LINQ** methods on it.

In order to get all of the matches and put them into a string array, we need to perform several manipulations on the **MatchCollection**:

* Cast every single element of the **MatchCollection** to the **Match** type using **Cast<Match>()**.
* Since every element is a **Match** now, we can extract just the **Value** property of the match itself, which holds the **match value** as a **string**, using **Select()**. We can also **Trim() the value**, to get rid of any **leading** or **trailing spaces**.
* After getting the match value, we can use **ToArray()** to **convert** the collection to an **array**.

Here’s what that looks like as a **LINQ** query:



After that, just print the valid phone number array, using **string.Join()**:



**Examples**

|  |
| --- |
| **Input** |
| +359 2 222 2222,359-2-222-2222, +359/2/222/2222, +359-2 222 2222 +359 2-222-2222, +359-2-222-222, +359-2-222-22222 +359-2-222-2222 |
| **Output** |
| +359 2 222 2222, +359-2-222-2222 |