Java Regular Expression

What is a Regular Expression (regex)?

java.util.regex

- A regular expression is **a sequence of characters that forms a search pattern**. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- Regular expressions can be used to perform all types of **text search** and **text replace** operations.
- A regular expression **can be a single character, or a more complicated pattern**.

Source: https://www.w3schools.com/java/java_regex.asp

What is a Regular Expression (regex)?

Example

```
Pattern pattern = Pattern.compile("@");
Matcher matcher = pattern.matcher("NguyenVanA@uit.edu.vn!");
boolean matchFound = matcher.find();
if (matchFound) {
    System.out.println("The email address is valid");
} else {
    System.out.println("The email address is invalid");
}
```

What is a Regular Expression (regex)?

Example

Mẫu kiểm tra chuỗi nhập chỉ được phép chứa một hay nhiều ký tự từ a-z chữ thường, A-Z chữ in, chữ số 0-9, dấu "." và dấu "-"

```
String username = "Nguyen.Van.A.2020";
Pattern pattern = Pattern.compile("[A-Za-z0-9.-]+");
Matcher matcher = pattern.matcher(username);

if (matcher.matches()) {
    System.out.println("The username is valid");
} else {
    System.out.println("The username is invalid");
}
```

Validate an username, password, email?

- **Username: ?**
  - # Match characters and symbols in the list, a-z, 0-9 , underscore , hyphen (-).
  - # Length at least 3 characters and maximum length of 15
- **Password: ?**
  - # must contains one digit from 0-9.
  - # must contains one lowercase characters.
  - # must contains one uppercase characters.
  - # must contains one special symbols in the list "@#$%“.
  - # length at least 6 characters and maximum of 20.

Validate an username/a password/ an email?

- **Email: Validate?**
  - # must start with string in the bracket [_A-Za-z0-9-].
  - # must contains a "@" symbol
  - …

How do we extract metadata from a card visit?

Java.util.regex

- Pattern Class - Defines a pattern (to be used in a search)
- Matcher Class - Used to search for the pattern
- PatternSyntaxException Class - Indicates syntax error in a regular expression pattern

Ref: https://docs.oracle.com/javase/8/docs/api/index.html?java/util/regex/package-summary.html

Java Regex - Syntax

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find one character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Java Regex - Example

```
String username = "NguyenVanX";
Pattern pattern = Pattern.compile("[abc]");
Matcher matcher = pattern.matcher(username);
if (matcher.find()) {
    System.out.println("Found");
} else {
    System.out.println("Not Found");
}
```

Java Regex - Example

```
String username = "aaaaaaaaabbbbbbbccccccc";
Pattern pattern = Pattern.compile("[^abc]");
Matcher matcher = pattern.matcher(username);
if (matcher.find()) {
    System.out.println("Found");
} else {
    System.out.println("Not Found");
}
```

Java Regex - Example

```
String username = "ABCDEF";
Pattern pattern = Pattern.compile("[0-9]");
Matcher matcher = pattern.matcher(username);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

```
String username = "ABCDEF9";
Pattern pattern = Pattern.compile("[0-9]");
Matcher matcher = pattern.matcher(username);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

Java Regex - Syntax

Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b

Java Regex - Example

```
String str = "cat|dog|fish";
Pattern pattern = Pattern.compile("|");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
String[] animals = str.split("[|]");
```

Java Regex - Example

```
String str = "";
Pattern pattern = Pattern.compile(".");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

```
String str = "abcxyz!@#";
Pattern pattern = Pattern.compile(".");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

Java Regex - Example

```java
String str = "Hello Class";
Pattern pattern = Pattern.compile("^Hello");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = " Hello Class";
Pattern pattern = Pattern.compile("^Hello");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

Java Regex - Example

```
String str = "Java la mot ngon ngu HDT. Java ho tro da nen";
Pattern pattern = Pattern.compile("^Java");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println(matcher.start());
    System.out.println(matcher.end());
}
```

Tìm trong chuỗi giá trị của biến str
xem có bắt đầu bằng chuỗi Java không?
Xuất ra vị trí tìm thấy.

Java Regex - Example

```java
String str = "Hello World";
Pattern pattern = Pattern.compile("World$");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World ";
Pattern pattern = Pattern.compile("World$");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

Java Regex - Example

```
String str = "Hello World";
Pattern pattern = Pattern.compile("\\d");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

```
String str = "Hello World 1";
Pattern pattern = Pattern.compile("\\d");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

Java Regex - Example

```
String str = "HelloWorld";
Pattern pattern = Pattern.compile("\\s");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// NOT FOUND
```

```
String str = "Hello World";
Pattern pattern = Pattern.compile("\\s");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World";
Pattern pattern = Pattern.compile("Hello\\b");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World";
Pattern pattern = Pattern.compile("\\bHello");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World";
Pattern pattern = Pattern.compile("\\bWorld");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World";
Pattern pattern = Pattern.compile("World\\b");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Hello World";
// Unicode of H
Pattern pattern = Pattern.compile("\\u0048");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
// FOUND
```

```java
String str = "Gello World";
// Unicode of H
Pattern pattern = Pattern.compile("\\u0048");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
} else {
    System.out.println("NOT FOUND");
}
//NOT FOUND
```

```java
String str = "Java la ngon ngu HDT. Java doc lap nen";
Pattern pattern = Pattern.compile("\\bJ..a\\b");
Matcher matcher = pattern.matcher(str);
while (matcher.find()) {
    System.out.println(matcher.start());
    System.out.println(matcher.end());
}

// [0, 4]; [22, 26]
```

Tìm trong chuỗi giá trị của biến inputStr những nơi xuất hiện từ bắt đầu là J kết thúc là a. Xuất ra vị trí tìm thấy.

Java Regex - Syntax

Quantifier	Description
n+	Matches any string that contains **at least one** *n*
n*	Matches any string that contains **zero or more** occurrences of *n*
n?	Matches any string that contains **zero or one** occurrences of *n*
n{x}	Matches any string that contains a sequence of *X n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

Java Regex - Example

```
String str = "Hello world";
Pattern pattern = Pattern.compile("n+");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// NOT FOUND
```

```
String str = "Hello world n";
Pattern pattern = Pattern.compile("n+");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// FOUND
```

Java Regex - Example

```
String str = "";
// 0 or more occurences of n
Pattern pattern = Pattern.compile("n*");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("TRUE");
}
else
    System.out.println("FALSE");

// TRUE
```

```
String str = "";
// 0 or 1 occurences of n
Pattern pattern = Pattern.compile("n?");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("TRUE");
}
else
    System.out.println("FALSE");

// TRUE
```

Java Regex - Example

```java
String str = "Hello World";
// occurence of "ooo"
Pattern pattern = Pattern.compile("o{3}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// NOT FOUND
```

```java
String str = "Hello Wooorld";
// occurence of "ooo"
Pattern pattern = Pattern.compile("o{3}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// FOUND
```

Java Regex - Example

```
String str = "Hello Woorld";
Pattern pattern = Pattern.compile("o{2,5}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// FOUND
```

```
String str = "Hello World";
Pattern pattern = Pattern.compile("o{2,5}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// NOT FOUND
```

Java Regex - Example

```
String str = "Hello World";
// contains a sequence of at least oo
Pattern pattern = Pattern.compile("o{2,}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// NOT FOUND
```

```
String str = "Hello Woorld";
// contains a sequence of at least oo
Pattern pattern = Pattern.compile("o{2,}");
Matcher matcher = pattern.matcher(str);
if (matcher.find()) {
    System.out.println("FOUND");
}
else
    System.out.println("NOT FOUND");

// FOUND
```

Java Regex - Syntax

- **Meta-characters:** are used to group, divide, and perform special operations in patterns.

Classes	Description
[…]	Khớp bất kỳ ký tự trong []
a-z	Range
[a-e][i-u]	Union
[a-z&&[aeiou]]	Intersection
()	Grouping

Java Regex - Syntax

Character Classes	Character Classes
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z](subtraction)

Java Regex Groups

- We can group multiple characters as a unit by parentheses. For example, **(ab)**.
- Each group in a regular expression has a **group number**, which **starts at 1**.
- Method **groupCount()** from Matcher class **returns the number of groups** in the pattern associated with the Matcher instance.

Java Regex - Example

Mẫu kiểm tra chuỗi nhập gồm ký tự
từ a-z chữ thường, A-Z chữ in, chữ số 0-9
và khoảng trắng (\s).
Chuỗi nhập chứa ít nhất 5, nhiều nhất 20 ký tự.

```
String username = "Nguyen Van A 123";
Pattern pattern = Pattern.compile("[A-Za-z0-9\\s]{5,20}");
if (pattern.matcher(username).matches()) {
    System.out.println("Username is valid");
} else {
    System.out.println("Username is invalide");
}
```

```
String regex = "\\b(\\d{3})(\\d{3})(\\d{4})\\b";
Pattern p = Pattern.compile(regex);
String inputStr = "1234567890, 12345,  and  9876543210";
Matcher m = p.matcher(inputStr);
while (m.find()) {
    System.out.println(
            "Phone: " + m.group() +
            ", Formatted Phone:  (" + m.group(1) + ") "
            + m.group(2) + "-" + m.group(3));
}
```

Tìm, khớp chuỗi nhập những nơi xuất hiện mẫu
Theo 3 nhóm "3 chữ số, 3 chữ số, 4 chữ số".
Xuất ra các nhóm và vị trí tìm thấy.

```
Phone:1234567890, Formatted Phone: (123)456-7890
Phone:9876543210, Formatted Phone: (987)654-3210
```

Java Regex - Syntax

- For the details of the regular expression **constructs**

https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

Examples: Matching/Validating

```
public static void main(String[] args) {
    List<String> input = new ArrayList<String>();
    input.add("123-45-6789");
    input.add("987-50-4321 ");
    input.add("987-65-4321 (attack)");
    input.add("987-65-4321 ");
    input.add("19283-7465");
    for (String ssn : input) {
        if (ssn.matches("^(\\d{3}-?\\d{2}-?\\d{4})$")) {
            System.out.println("Found good SSN: " + ssn);
        }
    }
}
```

```
Found good SSN: 123-45-6789
Found good SSN: 19283-7465
```

Dissecting the pattern:

```
"^(\\d{3}-?\\d{2}-?\\d{4})$"
```

```
^               match the beginning of the line
()              group everything within the parenthesis as group 1
\d{n}           match n digits, where n is a number equal to or greater than zero
-?              optionally match a dash
$               match the end of the line
```

Examples: Extracting/Capturing

```
String input = "I have a cat, but I like my dog better.";
Pattern p = Pattern.compile("(mouse|cat|dog|wolf|bear|human)");
Matcher m = p.matcher(input);
List<String> animals = new ArrayList<String>();
while (m.find()) {
    System.out.println("Found a " + m.group() + ".");
    System.out.println("start(): " + m.start());
    System.out.println("end(): " + m.end());
    animals.add(m.group());
}
```

```
Found a cat.
start(): 9
end(): 12
Found a dog.
start(): 28
end(): 31
```

Examples: Extracting/Capturing

- String INPUT = "cat cat cat cattie cat";
- Count/extract “cat” word?

```
private static final String REGEX = "\\bcat\\b";
private static final String INPUT = "cat cat cat cattie cat";

public static void main(String args[]) {
    Pattern p = Pattern.compile(REGEX);
    Matcher m = p.matcher(INPUT);   // get a matcher object
    int count = 0;

    while (m.find()) {
        count++;
        System.out.println("Match number " + count);
        System.out.println("start(): " + m.start());
        System.out.println("end(): " + m.end());
    }
}
```

Examples: Modifying/Substitution

```
String input = "User clientId=23421. Some more text clientId=33432. "
        + "This clientNum=100";
Pattern p = Pattern.compile("(clientId=)(\\d+)");
Matcher m = p.matcher(input);
StringBuffer result = new StringBuffer();
while (m.find()) {
    System.out.println("Masking: " + m.group(2));
    m.appendReplacement(result, m.group(1) + "***masked***");
}
// It is intended to be invoked after the appendReplacement
// method in order to copy the remainder of the input sequence.
m.appendTail(result);
System.out.println(result);
```

```
Masking: 23421
Masking: 33432
User clientId=***masked***. Some more text clientId=***masked***. This clientNum=100
```

Dissecting the pattern:

```
"(clientId=)(\\d+)"
```

```
(clientId=)     group everything within the parenthesis as group 1
clientId=       match the text 'clientId='
(\\d+)          group everything within the parenthesis as group 2
\\d+            match one or more digits
```

Examples: Validate username

```
List<String> input = new ArrayList<String>();
input.add("&nguyenvana123");
input.add("nguyenvana123");
input.add("_nguyenvana123");
input.add("-nva123");
input.add("nva");
for (String username : input) {
    if (username.matches("^[a-z0-9_-]{3,15}$")) {
        System.out.println("Good username: " + username);
    }
}
```

“^[a-z0-9_-]{3,15}$”

- ^ # Start of the line
- [a-z0-9_-] # Match characters and symbols in the list, a-z, 0-9 , underscore , hyphen
- {3,15} # Length at least 3 characters and maximum length of 15
- $ # End of the line

Examples: Validate password

```
List<String> input = new ArrayList<String>();
input.add("@nguyenvanA123");
input.add("nguyenvana123");
input.add("_nguyenvana123");
input.add("-nva123");
input.add("nva");
for (String pass : input) {
    if (pass.matches("((?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,20})")) {
        System.out.println("Good password: " + pass);
    }
}
```

```
((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,20})
```

```
(           # Start of group
  (?=.*\d)      #   must contains one digit from 0-9
  (?=.*[a-z])       #   must contains one lowercase characters
  (?=.*[A-Z])       #   must contains one uppercase characters
  (?=.*[@#$%])      #   must contains one special symbols in the list "@#$%"
              .     #     match anything with previous condition checking
                {6,20}  #       length at least 6 characters and maximum of 20
)           # End of group
```

Examples: Validate an email?

- Contain ‘@’ symbol?
- …?

Bài tập phần RegEx

1. Viết chương trình kiểm tra định dạng cho dữ liệu nhập là username theo quy ước:
   - Chữ đầu tiên không phải là số
   - Chiều dài trong khoảng từ 4 -> 12 ký tự
   - Chỉ chấp nhận các chữ số từ 0-9, chữ cái thường, chữ cái hoa và dấu gạch dưới (_)
2. Viết chương trình kiểm tra định dạng cho dữ liệu nhập là password theo quy ước:
   - Mật khẩu ít nhất 8 ký tự
   - Phải có chữ hoa, chữ thường, số và 1 ký tự đặc biệt
3. Kiểm tra định dạng ngày tháng năm.
   - Kiểm tra chuỗi nhập có đúng định dạng ngày tháng năm dd/mm/yyyy hay không?
4. Viết chương trình kiểm tra email có đúng định dạng hay không?
5. Viết chương trình kiểm tra một URL nhập vào có đúng định dạng hay không?
6. Viết chương trình cho phép lấy danh sách tất cả các links của trang www.uit.edu.vn (links trong thẻ <a>)
7. Viết chương trình cho phép lấy tất cả các địa chỉ dưới dạng các URL của các hình ảnh từ trang www.uit.edu.vn (URL trong thẻ <img>)