

Лабораторная работа №14 Регулярные выражения

Цель работы: научиться работать с регулярными выражениями в языке Java.

Теоретическая часть

Регулярные выражения (RegEx) – это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Само регулярное выражение представляет шаблон (англ. pattern) для поиска совпадений в строке.

Синтаксис регулярных выражений основан на использовании символов `<([{\^-= $!|]})? * + . >`, которые можно комбинировать с буквенными символами. В зависимости от роли их можно разделить на несколько групп:

1. Метасимволы для поиска символьных классов

- | | |
|-----------------|---|
| <code>\d</code> | цифровой символ |
| <code>\D</code> | нецифровой символ |
| <code>\s</code> | символ пробела |
| <code>\S</code> | непробельный символ |
| <code>\w</code> | буквенно-цифровой символ или знак подчёркивания |
| <code>\W</code> | любой символ, кроме буквенного, цифрового или знака подчёркивания |
| <code>.</code> | любой символ |

2. Метасимволы для поиска символов редактирования текста

\t	символ табуляции
\n	символ новой строки
\r	символ возврата каретки
\f	переход на новую страницу

3. Метасимволы для группировки символов

[абв]	любой из перечисленных (а, б или в)
[^абв]	любой, кроме перечисленных (не а, б, в)
[a-zA-Z]	слияние диапазонов (от а до z без учета регистра)

4. Метасимволы для обозначения количества символов – квантификаторы.

Квантификатор всегда следует после символа или группы символов.

X?	X один раз или ни разу
X*	X ноль или более раз
X+	X один или более раз
X{n}	X n раз
X{n,}	X n или более раз
X{n,m}	X от n до m

5. Логические операции

XY	после X следует Y
X Y	X либо Y
(X)	X

6. Метасимволы для поиска совпадений границ строк или текста

^	начало строки
\$	конец строки
\b	граница слова
\B	не граница слова
\A	начало ввода
\G	конец предыдущего совпадения
\Z	конец ввода
\z	конец ввода

Практическая часть №1

Задание 1. Отработать нахождение определенных символов и их соединений в <https://regexr.com/> (провести не менее 10 различных по сложности регулярных выражений). Нельзя использовать регулярные выражения, подходящие для решения задач из второго задания.

Задание 2. Написать и протестировать в <https://regexr.com/> следующие регулярные выражения:

1. Поиск даты рождения.
2. Нахождение номера сотового телефона.
3. Проверка корректного ввода email.
4. Дана строка '23 2+3 2++3 2+++3 345 567'. Напишите регулярное выражение, которое найдет строки 2+3, 2++3, 2+++3, не захватив остальные («+» может быть любое количество).

Pattern и Matcher

Большая часть функциональности по работе с регулярными выражениями в Java сосредоточена в пакете `java.util.regex`, Основными классами для работы с регулярными выражения являются класс **`java.util.regex.Pattern`** и класс **`java.util.regex.Matcher`**.

Класс `java.util.regex.Pattern` применяется для определения регулярных выражений, для которого ищется соответствие в строке, файле или другом объекте, представляющем собой некоторую последовательность символов. При наличии ошибок в строке – генерируется исключение **`PatternSyntaxException`**.

1. Класс Pattern

Класс `Pattern` используется для простой обработки строк. Для более сложной обработки строк используется класс `Matcher`, рассматриваемый ниже.

В классе `Pattern` объявлены следующие методы:

- `compile(String regex)` – возвращает `Pattern`, который соответствует `regex`;
- `matcher(CharSequence input)` – возвращает `Matcher`, с помощью которого можно находить соответствия в строке `input`;
- `matches(String regex, CharSequence input)` – проверяет на соответствие строки `input` шаблону `regex`;
- `pattern()` – возвращает строку, соответствующую шаблону;
- `split(CharSequence input)` – разбивает строку `input`, учитывая, что разделителем является шаблон;
- `split(CharSequence input, int limit)` – разбивает строку `input` на не более чем `limit` частей.

С помощью метода `matches()` класса `Pattern` можно проверять на соответствие шаблону целой строки, но если необходимо найти соответствия внутри строки, например, определять участки, которые соответствуют шаблону, то класс `Pattern` не может быть использован. Для таких операций необходимо использовать класс `Matcher`.

2. Класс Matcher

Начальное состояние объекта типа `Matcher` не определено. Попытка вызвать какой-либо метод класса для извлечения информации о найденном соответствии

приведет к возникновению ошибки `IllegalStateException`. Для того чтобы начать работу с объектом `Matcher` нужно вызвать один из его методов:

- `matches()` – проверяет, соответствует ли вся строка шаблону;
- `lookingAt()` – пытается найти последовательность символов, начинающуюся с начала строки и соответствующую шаблону;
- `find()` или `find(int start)` – пытается найти последовательность символов, соответствующих шаблону, в любом месте строки.

Параметр `start` указывает на начальную позицию поиска.

Иногда необходимо сбросить состояние объекта класса `Matcher` в исходное, для этого применяется метод `reset()` или `reset(CharSequence input)`, который также устанавливает новую последовательность символов для поиска.

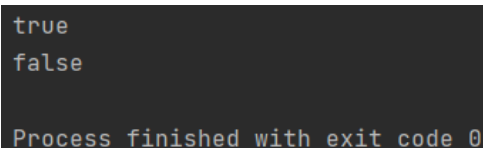
Для замены всех подпоследовательностей символов, удовлетворяющих шаблону, на заданную строку можно применить метод `replaceAll(String replacement)`.

Примеры использования возможностей классов `Pattern` и `Matcher` для поиска, разбора и разбиения строк

Пример 1. Проверка на числовую строку.

```
String s1 = "1234";
String s2 = "1234s";
Pattern pt = Pattern.compile("\\d+");
Matcher mt1 = pt.matcher(s1);
System.out.println(mt1.matches());
Matcher mt2 = pt.matcher(s2);
System.out.println(mt2.matches());
```

Вывод в консоль:



```
true
false

Process finished with exit code 0
```

Пример 2. Программа, которая проверяет, содержит ли строка только определенный набор символов (в данном случае a-z, A-Z и 0-9) (с помощью метода `matches()` класса `String`).

```
import java.util.Scanner;

public class ExampleRegular {
    public static void main(String[] args) {
        System.out.println(checkRegEx("ABCDabcd1234"));
        System.out.println(checkRegEx("Aa1"));
        System.out.println(checkRegEx("A1."));
        System.out.println(checkRegEx("*&%@#!}{"));
    }
}
```

```

        public static boolean checkRegex(String s) {
            return s.matches("^[\\w]+$");
        }
    }
}

```

Вывод в консоль:

```

true
true
false
false

Process finished with exit code 0

```

Пример 3. Программа, которая возвращает строку, в которой за буквой *a* следует ноль или более букв *b* (с помощью метода `matches()` класса `String`).

```

import java.util.Scanner;

public class ExampleRegular {
    public static void main(String[] args) {
        System.out.println(checkRegex("a"));
        System.out.println(checkRegex("ab"));
        System.out.println(checkRegex("abb"));
        System.out.println(checkRegex("aba"));
        System.out.println(checkRegex("abab"));
    }

    public static boolean checkRegex(String s) {
        if (s.matches("ab*"))
            return "Found a match!";
        else
            return "Not matched!";
    }
}

```

Вывод в консоль:

```

Found a match!
Found a match!
Found a match!
Not matched!
Not matched!

Process finished with exit code 0

```

Пример 4. Нахождение не полного соответствия, а отдельных совпадений в строке.

```

import java.util.regex.*;

public class ExampleRegular {

```

```

public static void main(String[] args) {
    String input = "Hello Java! Hello JavaScript! JavaSE 8.";
    Pattern pattern = Pattern.compile("Java(\\w*)");
    Matcher matcher = pattern.matcher(input);
    while(matcher.find())
        System.out.println(matcher.group());
}
}

```

Пример 5. Замена всех совпадений с помощью метода `replaceAll()`.

```

String input = "Hello Java! Hello JavaScript! JavaSE 8.";
System.out.println(input);
Pattern pattern = Pattern.compile("Java(\\w*)");
Matcher matcher = pattern.matcher(input);
String newStr = matcher.replaceAll("HTML");
System.out.println(newStr);

```

Вывод в консоль:

```

Hello Java! Hello JavaScript! JavaSE 8.
Hello HTML! Hello HTML! HTML 8.

Process finished with exit code 0

```

В классе `String` также имеется метод `replaceAll()` с подобным действием:

```

String input = "Hello Java! Hello JavaScript! JavaSE 8.";
System.out.println(input);
String myStr = input.replaceAll("Java(\\w*)", "HTML");
System.out.println(myStr);

```

Вывод в консоль:

```

Hello Java! Hello JavaScript! JavaSE 8.
Hello HTML! Hello HTML! HTML 8.

Process finished with exit code 0

```

Практическая часть №2

Задание 3. Напишите программу для поиска последовательностей строчных букв, соединенных знаком подчеркивания.

Задание 4. Напишите программу, чтобы проверить, является ли данная строка шестнадцатеричным кодом или нет.

Задание 5. Напишите программу для удаления начальных нулей с заданного IP-адреса.

Задание 6. Напишите программу для разделения символов в заданной строке на группы: буквы, цифры и знаки препинания. Вывод результата осуществить в три разные подстроки.

Задание 7. Напишите программу для проверки надежности пароля. Пароль считается надежным, если он состоит из 8 или более символов, где символом может быть английская буква, цифра и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру.

– пример правильных выражений: C00l_Pass, SupperPas1.

– пример неправильных выражений: Cool_pass, C00l.

Задание 8. Напишите программу для обратного преобразования слов длиной более 5 символов в заданной строке.

Задание 9. Напишите программу для перемещения всех строчных букв в начало заданного слова, сохраняя относительное положение всех букв (как в верхнем, так и в нижнем регистре) одинаковым. Например, JavaScript -> avascriptJS.