

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Московский технический университет связи и информатики»

Кафедра «Математическая Кибернетика и Информационные технологии»

Дисциплина «Информационные технологии и программирование»

Лабораторная работа №5

«Регулярные выражения в Java»

Выполнила:

Студентка группы БВТ2303

Морозова Ольга

Цель работы:

Изучение регулярных выражений и применение полученных знаний на практике на языке программирования Java.

Ход работы:

Задание 1.

Напишем регулярное выражение для поиска всех чисел в заданном тексте (положительных и отрицательных целых и дробных (натуральных и десятичных) чисел).

Используем экранированный метасимвол `\d` для поиска числовых символов.

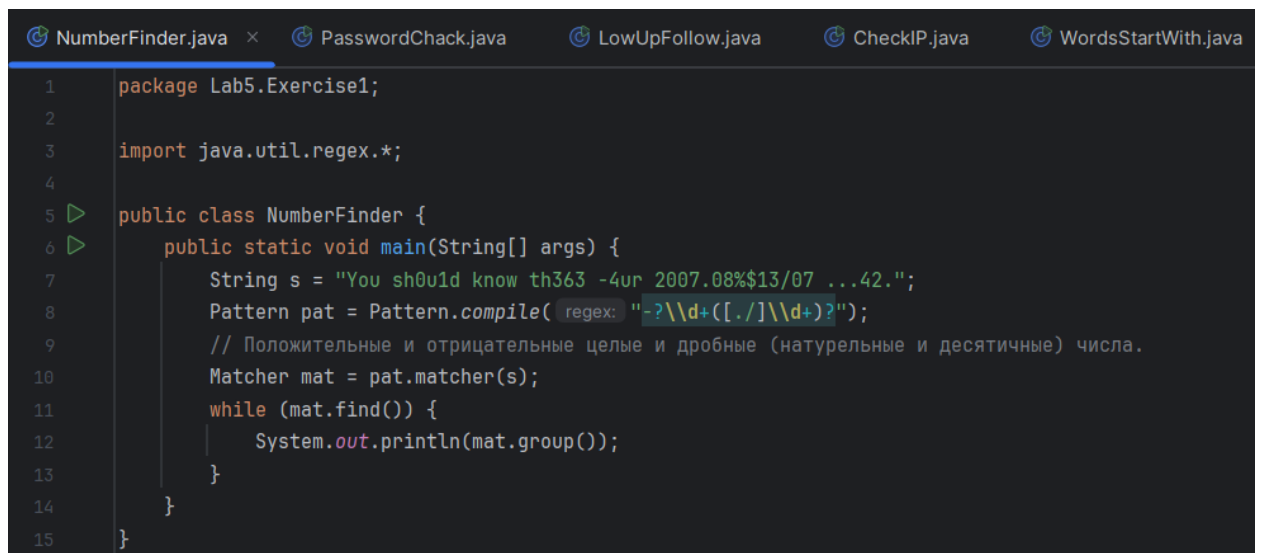
Используем `[./]` для указания, что встречается один из перечисленных в скобках символов.

Также используем квантифакторы, обозначающие количество символов:

* – ноль или более раз;

+ – один или более раз;

? – один раз или отсутствует;



```
1 package Lab5.Exercise1;
2
3 import java.util.regex.*;
4
5 public class NumberFinder {
6     public static void main(String[] args) {
7         String s = "You sh0u1d know th363 -4ur 2007.08%$13/07 ...42.";
8         Pattern pat = Pattern.compile("regex: \"-?\\d+([./]\\d+)?\"");
9         // Положительные и отрицательные целые и дробные (натуральные и десятичные) числа.
10        Matcher mat = pat.matcher(s);
11        while (mat.find()) {
12            System.out.println(mat.group());
13        }
14    }
15 }
```

Задание 2.

Напишем регулярное выражение для проверки корректности введенного пароля. Пароль должен состоять из латинских букв и цифр, быть длиной от 8 до 16 символов и содержать хотя бы одну заглавную букву и одну цифру.

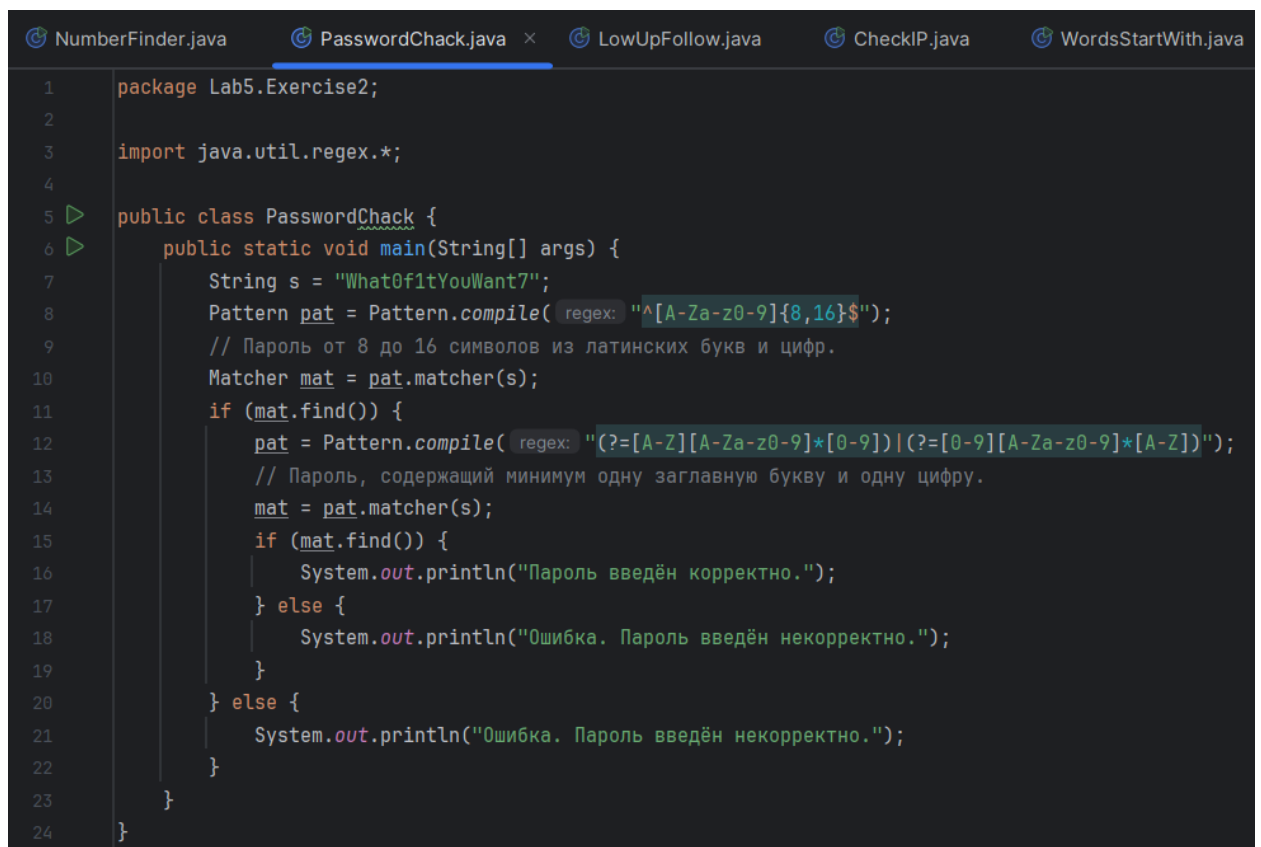
Первое регулярное выражение проверяет количество символов в пароле и их содержание (только латинские буквы (заглавные и строчные) и цифры).

Для этого используем квантифактор {8,16} – значит повторение символа (одного из указанных в квадратных скобках) будет от 8 до 16 раз.

Можно указывать просто {8} – это точное число повторений, или же {8,} – от 8 раз и более, например.

И обязательно при проверке длины указываем символы начала «^» и конца «\$» строки.

Второе регулярное выражение проверяет чes помощью логического ИЛИ - ()|() – обязательное наличие одной заглавной буквы и одного числа (ИЛИ мы используем, чтобы указать два варианта – когда заглавная буква стоит в пароле до числа или после него).



```
1 package Lab5.Exercise2;
2
3 import java.util.regex.*;
4
5 public class PasswordChack {
6     public static void main(String[] args) {
7         String s = "What0f1tYouWant7";
8         Pattern pat = Pattern.compile( regex: "^[A-Za-z0-9]{8,16}$");
9         // Пароль от 8 до 16 символов из латинских букв и цифр.
10        Matcher mat = pat.matcher(s);
11        if (mat.find()) {
12            pat = Pattern.compile( regex: "^(?=.*[A-Z])[A-Za-z0-9]*[0-9]|^(?=.*[0-9])[A-Za-z0-9]*[A-Z]$");
13            // Пароль, содержащий минимум одну заглавную букву и одну цифру.
14            mat = pat.matcher(s);
15            if (mat.find()) {
16                System.out.println("Пароль введен корректно.");
17            } else {
18                System.out.println("Ошибка. Пароль введен некорректно.");
19            }
20        } else {
21            System.out.println("Ошибка. Пароль введен некорректно.");
22        }
23    }
24 }
```

Задание 3.

Напишем регулярное выражение для нахождения всех случаев в тексте, когда сразу после строчной буквы идёт заглавная а после программой выделим эти случаи в тексте знаками «!» с двух сторон.

К слову, в диапазон русских букв от А до Я (что строчных, что заглавных) не включена буква Ё, поэтому пришлось прописывать её дополнительно в группах.

Замену производим с помощью двух циклов (while и for) для прохождения по всем найденным случаям и после нахождения их в строке с последующим преобразованием оной.

```
NumberFinder.java PasswordCheck.java LowUpFollow.java x CheckIP.java WordsStartWith.java
1 package Lab5.Exercise3;
2
3 import java.util.regex.*;
4
5 public class LowUpFollow {
6     public static void main(String[] args) {
7         String s = "iF You th1nK яH!e 007 th@n iF Ты&*. bcĚ";
8         Pattern pat = Pattern.compile("([a-z][A-Z])|([a-яё][A-ЯЁ])|([a-z][A-ЯЁ])|([a-яё][A-Z])");
9         // Сразу после строчной буквы идёт заглавная.
10        Matcher mat = pat.matcher(s);
11        int[] h = new int[] {0, s.length()};
12        while (mat.find()) {
13            for (int i = h[0]; i < h[1] - 1; i++) {
14                if (String.valueOf(mat.group()).equals(String.valueOf(s.charAt(i)) + String.valueOf(s.charAt(i + 1)))) {
15                    s = s.substring(0, i) + "!" + String.valueOf(mat.group()) + "!" + s.substring(beginIndex: i + 2);
16                    h[0] = i + 4;
17                    h[1] += 2;
18                    break;
19                }
20            }
21        }
22        System.out.println(s);
23    }
24 }
```

Задание 4.

Напишем регулярное выражение для проверки корректности введённого IP-адреса. IP-адрес должен состоять из 4 чисел, разделенных точками, и каждое число должно быть в диапазоне от 0 до 255.

Обязательно при проверке длины указываем символы начала «^» и конца «\$» строки.

Через ИЛИ проверяем все варианты для каждой из 4 групп чисел (чтобы исключить ведущие нули и не превысить порог в 255). Между ними ставим обязательные точки.

```
NumberFinder.java PasswordCheck.java LowUpFollow.java CheckIP.java x WordsStartWith.java
1 package Lab5.Exercise4;
2
3 import java.util.regex.*;
4
5 public class CheckIP {
6     public static void main(String[] args) {
7         String s = "107.0.13.255";
8         Pattern pat = Pattern.compile("^((([1-9]?\\d)|(2[0-4]\\d)|(25[0-5]))\\.)(.)([1-9]?\\d)|(2[0-4]\\d)|(25[0-5]))\\.((([1-9]?\\d)|(2[0-4]\\d)|(25[0-5]))\\.)(.)([1-9]?\\d)|(2[0-4]\\d)|(25[0-5]))$");
9         // IP-адрес из четырёх чисел (от 0 до 255), разделённых точками.
10        Matcher mat = pat.matcher(s);
11        if (mat.find()) {
12            System.out.println("IP-адрес введён корректно.");
13        } else {
14            System.out.println("Ошибка. IP-адрес введён некорректно.");
15        }
16    }
17 }
18 }
```

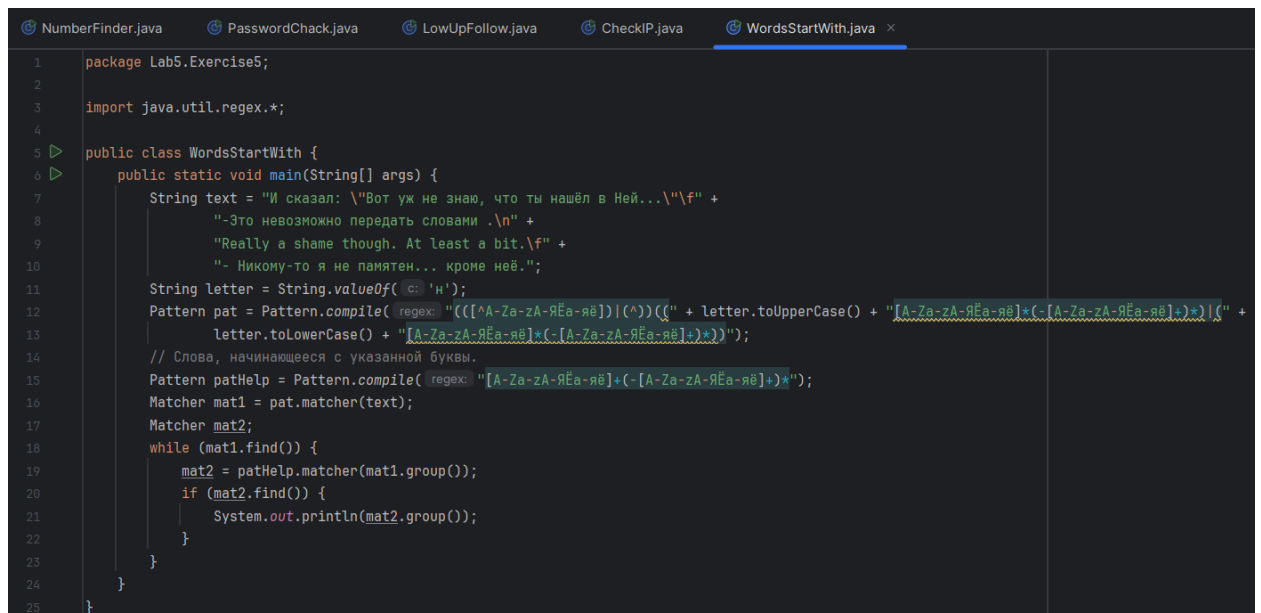
Задание 4.

Напишем регулярное выражение для поиска всех слов в заданном тексте, начинающихся с заданной буквы.

Используем конкатенацию строк, чтобы вставить значение переменной в регулярное выражение. Причём с помощью ИЛИ мы делаем одну и ту же проверку для заданной буквы сперва в её заглавном виде, а потом в строчном.

Чтобы определить начало слова используем `[^...]` с указанием внутри всех латинских и русских букв (заглавных и строчных), тем самым указывая, что символ перед желаемой первой буквой слова не должен являться буквой. Или же это должно быть начало строки.

Вторым же регулярным выражением мы просто избавляемся от этого первого “небуквенного” символа, чтобы на выходе получить чистые слова.



```
1 package Lab5.Exercise5;
2
3 import java.util.regex.*;
4
5 public class WordsStartWith {
6     public static void main(String[] args) {
7         String text = "И сказал: \"Вот уж не знаю, что ты нашёл в Ней...\"\\f\" +
8             \"-Это невозможно передать словами.\\n\" +
9             \"Really a shame though. At least a bit.\\f\" +
10            \"- Никому-то я не памятен... кроме неё.\";
11         String letter = String.valueOf('н');
12         Pattern pat = Pattern.compile( "[([A-Za-zА-ЯЁа-яё])|(^)((\" + letter.toUpperCase() + \"[A-Za-zА-ЯЁа-яё]*(-[A-Za-zА-ЯЁа-яё]+)*)|(\" +
13             letter.toLowerCase() + \"[A-Za-zА-ЯЁа-яё]*(-[A-Za-zА-ЯЁа-яё]+)*)\"));
14         // Слова, начинающиеся с указанной буквы.
15         Pattern patHelp = Pattern.compile( "([A-Za-zА-ЯЁа-яё]+(-[A-Za-zА-ЯЁа-яё]+)*)");
16         Matcher mat1 = pat.matcher(text);
17         Matcher mat2;
18         while (mat1.find()) {
19             mat2 = patHelp.matcher(mat1.group());
20             if (mat2.find()) {
21                 System.out.println(mat2.group());
22             }
23         }
24     }
25 }
```

Вывод:

Мы изучили регулярные выражения, а также написали программы на языке программирования Java с использованием полученных знаний.

GitHub - https://github.com/MiniLynx13/ITaP_Lab5