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 «Программная инженерия»

Отчет по лабораторной работе № 5.

по дисциплине «Информационные технологии и программирование»

по разделу:

«5. Строки и регулярные выражения»

Выполнила: студентка группы БПИ2401 Рябова Екатерина

Проверил: Харрасов Камиль Раисович

Москва

2025

Содержание

Цель работы:

Познакомиться с регулярными выражениями в Java, научиться выполнять операции со строками – поиск, замена, проверка формата.

Ход работы:

Задание 1: Поиск всех чисел в тексте

Написать программу, которая будет искать все числа в заданном тексте и выводить их на экран. При этом программа должна использовать регулярные выражения для поиска чисел и обрабатывать возможные ошибки.

Реализуем метод опираясь на предложенный в методичке пример: используем pattern.compile, pattern.matcher и matcher.group. Однако добавляем ввод в случае не нахождения чисел. Обернем все в конструкцию try-catch где будем ловить ошибки NPE и ошибку синтаксиса паттерна (на случай, если написанное нам регулярное выражение составлено некорректно). Ловим так

же все остальные ошибки на всякий случай. Реализуем ввод текста через аргументы командной строки.

Видоизменим предложенную в методичке регулярку, чтобы находились все числа:

("-?\\d+(\\.\\d+)?")

- -?: может быть, а может и не быть минуса.
- \\d + : одна или более цифра подряд
- \\.\\d+ : экранированная точка и одна или более цифра
- ? и вся эта конструкция с точкой и дробной частью может быть, а может её и не быть.

Листинг 1.1. Программа, которая находит все числа в тексте.

```
package fifthlab;

import java.util.regex.*;

public class NumberFinder {
    public static void main(String[] args){
        try{
            String text = String.join(" ", args);
            Pattern pattern = Pattern.compile("-?\\d+(\\.\\d+)?");
            Matcher matcher = pattern.matcher(text);
            System.out.println("Найденные числа: ");
            boolean found = false;

            while (matcher.find()) {
                System.out.println(matcher.group());
                found = true;
            }

            if (!found) {
                System.out.println("Числа не найдены");
            }
        } catch (PatternSyntaxException e){
            System.out.println("Ошибка в регулярном выражении " +
e.getMessage());
        } catch (NullPointerException e){
            System.out.println("Текст не может быть null " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Произошла ошибка: " + e.getMessage());
```

```
        }
    }
}
```

Результат представлен на рисунке 1:

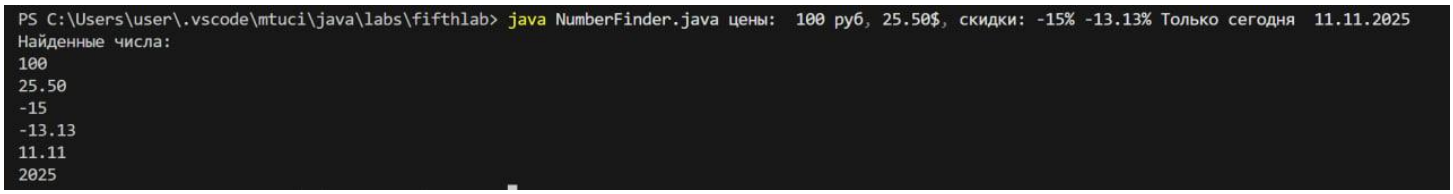

Рисунок 1. Результат корректного выполнения программы.

Задание 2: Проверка корректности ввода пароля.

Написать программу, которая будет проверять корректность ввода 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. При этом программа должна использовать регулярные выражения для проверки пароля и обрабатывать возможные ошибки.

Используем matcher.matches() для проверки на соответствие формату (проверка полного совпадения). Напишем следующее регулярное выражение:

"^(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]{8,16}$"

- ^ и $ говорят о том, что рассматривается вся строка
- ?= : позитивный просмотр вперед, который проверяет
    - .*[A-Z] : наличие заглавной латинской буквы после какого-то кол-ва символов
    - *[0-9] : наличие одной цифры после любого кол-ва символов.
- [a-zA-Z0-9]{8,16} : говорит, что может использоваться буквы от a до z, от A до Z, и цифры от 0 до 9. А общее кол-во символов от 8 до 16.

Листинг 2.1. Программа, которая проверяет корректность ввода пароля.

```java
package fifthlab;
import java.util.regex.*;

public class PasswordValidator {
    public static void main(String[] args){
        try{
            String password = String.join(" ", args);
            Pattern pattern = Pattern.compile("^(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-
9]{8,16}$");
            Matcher matcher = pattern.matcher(password);
            if (matcher.matches()) {
                System.out.println("Valid password!");
            } else{
                System.out.println("Invalid password!");
            }
        } catch (PatternSyntaxException e){
            System.out.println("Ошибка в регулярном выражении " +
e.getMessage());
        } catch (NullPointerException e){
            System.out.println("Текст не может быть null " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Произошла ошибка: " + e.getMessage());
        }
    }
}
```

Результат работы программы представлен на рисунке 2.

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234ABC
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234567890
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234567890ABC
Valid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234567abcde
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 1234567АБС
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java abcABCabcABC
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java abcABCabcABC12345678
Invalid password!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java PasswordValidator.java 123456778AB@
Invalid password!
```

Рисунок 2. Результат корректной работы программы

Задание 3: Поиск заглавной буквы после строчной

Написать программу, которая будет находить все случаи в тексте, когда сразу после строчной буквы идет заглавная без какого-либо символа между ними и выделять их знаками «!» с двух сторон.

Используем replaceAll() ддя замены всех найденных пар на !найденная пара!. Используем следующие паттерны:

- [a-z][A-Z]: подряд два символа от a до z и от A до Z.
- !$0! : оборачивает восклицательными знаками всю найденную строку ($0).

Листинг 3.1. Программа, , которая оборачивает паттерн строчная_буквaЗаглавная_буква в восклицательные знаки.

```
package fifthlab;
import java.util.regex.*;
public class CaseFinder {
    public static void main(String[] args){
        try {
            String text = String.join(" ", args);
            Pattern pattern = Pattern.compile("[a-z][A-Z]");
            Matcher matcher = pattern.matcher(text);
            String result = matcher.replaceAll("!$0!");
            System.out.println(result);
        } catch (PatternSyntaxException e){
            System.out.println("Ошибка в регулярном выражении " +
e.getMessage());
        } catch (NullPointerException e){
            System.out.println("Текст не может быть null " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Произошла ошибка: " + e.getMessage());
        }
    }
}
```

Результат работы программы представлен на рисунке 3:

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java CaseFinder.java Ui iS the most bEaUtiFFul Thing. LiTTeRelLy...
Ui !iS! the most !bE!!aU!t!iF!Ful Thing. L!iT!T!eR!e!lL!y...
```

Рисунок 3. Результат корректной работы программы.

Задание 4: Проверка корректности ввода IP-адреса

Написать программу, которая будет проверять корректность ввода IP-адреса. IP-адрес должен состоять из 4 чисел, разделенных точками, и каждое число должно быть в диапазоне от 0 до 255. При этом программа должна использовать регулярные выражения для проверки IP-адреса и обрабатывать возможные ошибки.

Используем matcher.matches() для проверки полного совпадения «шаблона» с ip-адрессом. Используем следующее регулярное выражение для проверки:

^((25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])$

- ^ и $ говорят о том, что рассматривается вся строка
- 25[0-5] – числа от 250 до 255
- 2[0-4][0-9] – числа от 200 до 249
- 1?[0-9]?[0-9] – числа от 0 до 199 (первая цифра либо 1 либо ее нет, вторая – от 0 до 9 либо ее может не быть (если число однозначное), третья цифра – от 0 до 9.
- (25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])\\.){3} – через или прописываем все эти варианты составления чисел, добавляем в конце точку (экранируемую точку) и такое выражение (с точкой на конце) должно встречаться 3 раза.
- Потом еще одно такое же выражение но без точки.

Единственное, что формат 01.01.01.01 не будет считать ошибочным (так как мы допускаем ноль в разряде десятков).

Листинг 4.1. Программа, проверяет ip-адрес на корректность.

```
package fifthlab;
import java.util.regex.*;

public class IPValidator {
```

```
    public static void main(String[] args){
        try {
            String ip = String.join(" ", args);
            Pattern pattern = Pattern.compile("^((25[0-5]|2[0-4][0-9]|1?[1-9]?[0-
9]|10[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])$");
            Matcher matcher = pattern.matcher(ip);
            if (matcher.matches()) {
                System.out.println("Valid ip!");
            } else{
                System.out.println("Invalid ip!");
            }
        } catch (PatternSyntaxException e){
            System.out.println("Ошибка в регулярном выражении " +
e.getMessage());
        } catch (NullPointerException e){
            System.out.println("Параметр не может быть null " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Произошла ошибка: " + e.getMessage());
        }
    }

}
```

Результат работы программы представлен на рисунке 4:

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 1.1.1.1
Valid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 001.001.001.001
Invalid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 01.01.01.01
Invalid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 100.100.100.101
Valid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 192.168.1.1
Valid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 255.255.255.255
Valid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 256.168.1.1
Invalid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 0.0.0.0
Valid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 2.3.4
Invalid ip!
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java IPValidator.java 2.3.4.5.6
Invalid ip!
```

Рисунок 4. Результат корректной работы программы.

Задание 5: Поиск всех слов, начинающихся с заданной буквы

Написать программу, которая будет искать все слова в заданном тексте, начинающиеся с заданной буквы, и выводить их на экран. При этом

программа должна использовать регулярные выражения для поиска слов и обрабатывать возможные ошибки.

Напишем два паттерна: для проверки первого символа на то, что он – буква и для поиска слов, начинающихся на букву. Используем matcher.find()) и сообщаем пользователю, что ничего не нашли. Так как тут мы требуем ввода и параметра и текста и пытаемся вытащить 0 и 1 элемент args было бы правильно добавить ловец ошибки выхождения за пределы массива.

[a-zA-Za-яA-Я]

- Ровно 1 символ
- От a до z или от A до Z или от a до я или от A до Я

"\\b(?i)" + letter + "[a-zA-Za-яA-Я]*\\b"

- \\b – границы слова (чтобы рассматривались только целиком слова а не их подстроки)
- (?i) – говорит, что регистр не важен
- letter – поиск самой нашей буквы, записанной в переменную
- [a-zA-Za-яA-Я]* - поиск любого кол-ва (в том числе и 0) букв следом за первой буквой слова

Однако \\b в некоторых версиях java не работает с кириллицей. Для проверки работы с кирилиицей видоизменим регулярное выражение, которое будет искать перед первой буквой пробел (чтобы обозначить начало слова). Для этого в текст в начале добавим пробел.

```
 \\s(?i)" + letter + "[a-zA-Za-яA-Я]*", Pattern.CASE_INSENSITIVE |
Pattern.UNICODE_CASE
```

Листинг 5.1. Программа, которая ищет слова в тексте, начинающиеся с определенной буквы.

```
package fifthlab;
import java.util.regex.*;
import java.util.Arrays;
```

```
public class WordFinder {
    public static void main(String[] args) {
        try {
            String letter = args[0];
            String text = String.join(" ", Arrays.copyOfRange(args, 1,
args.length));
            if (!letter.matches("[a-zA-Za-яA-Я]")) {
                System.out.println("Первый параметр должен быть буквой");
                return;
            }
            Pattern pattern = Pattern.compile("\\b(?i)" + letter + "[a-zA-Za-яA-
Я]*\\b");
            Matcher matcher = pattern.matcher(text);
            System.out.println("Слова, начинающиеся с буквы '" + letter + "': ");
            boolean found = false;
            while (matcher.find()) {
                System.out.println(matcher.group());
                found = true;
            }
            if (!found) {
                System.out.println("Слова не найдены");
            }
        } catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Текст не найден " + e.getMessage());
        }catch (PatternSyntaxException e){
            System.out.println("Ошибка в регулярном выражении " +
e.getMessage());
        } catch (NullPointerException e){
            System.out.println("Параметр не может быть null " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Произошла ошибка: " + e.getMessage());
        }
    }
}
```

Результат работы программы представлен на рисунках 5-6:

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java a apple Array play
Слова, начинающиеся с буквы 'a':
apple
Array
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java A apple Array play
Слова, начинающиеся с буквы 'A':
apple
Array
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java o apple Array play pops
Слова, начинающиеся с буквы 'o':
Слова не найдены
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java o
Слова, начинающиеся с буквы 'o':
Слова не найдены
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java
Текст не найден Index 0 out of bounds for length 0
```

Рисунок 5. Результат корректной работы программы.

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java 1 123 array
Первый параметр должен быть буквой
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java a a11b1 vVVv@
Слова, начинающиеся с буквы 'a':
Слова не найдены
```

Рисунок 6. Результат работы программы

```
PS C:\Users\user\.vscode\mtuci\java\labs\fifthlab> java WordFinder.java ч черепаха Чехия сочельник
Слова, начинающиеся с буквы 'ч':
 черепаха
 Чехия
```

Рисунок 6. Результат работы видоизмененной для кириллицы программы

Работа загружена на гитхаб по ссылке: https://github.com/Kateriabova/Riaboava_Kate_BPI2401_IT2.git

Ответы на контрольные вопросы:

1. Что такое класс String?

String - это final-класс в Java, представляющий неизменяемую последовательность символов Unicode. Внутри данные хранятся в массиве char[] (или byte[] в новых версиях для оптимизации).

2. Почему объект класса String является иммутабельным?

- Безопасность - строки можно безопасно передавать между потоками
- Кэширование хэшей - хэш-код вычисляется один раз

- Пул строк - возможность переиспользования строк
- Безопасность - защита от изменений в критических операциях (пароли, сетевые соединения)

3. Что такое интернирование строк?

Интернирование - процесс помещения строки в пул строк (String Pool). Метод intern() помещает строку в пул или возвращает существующую.

4. В чем разница между String, StringBuilder и StringBuffer??

- String - неизменяемый, потокобезопасный
- StringBuilder - изменяемый, НЕ потокобезопасный, быстрее
- StringBuffer - изменяемый, потокобезопасный (synchronized), медленнее

5. Как сравнить две строки? В чем разница между ==, equals() и equalsIgnoreCase()?

- == - сравнивает ссылки (адреса в памяти)
- equals() - сравнивает содержимое с учетом регистра
- equalsIgnoreCase() - сравнивает содержимое без учета регистра.

6. Как хранятся строки в памяти? Что такое пул строк?

Пул строк - специальная область в Heap memory для хранения уникальных строковых литералов. При создании строки через литерал "text" JVM сначала проверяет пул.

7. Что такое code unit и code point?

- Code Unit (кодовая единица) – минимальная битовая единица в кодировке. В UTF-16 это char (16 бит).
- Code Point (кодовая позиция) – числовое значение символа в стандарте Unicode.
- Суррогатные пары: символа за пределами Basic Multilingual Plane (U+10000 и выше) кодируются двумя char (суррогатной парой):

Высокий суррогат U+D800 – U+DBFF, Низкий суррогат U+DC00 – U+DFFF.

8. Что необходимо для использования регулярных выражений в Java?

Классы из пакета java.util.regex:

- Pattern - компилированное регулярное выражение
- Matcher - выполняет операции поиска.

9. Какие есть режимы работы квантификатора?

- Жадный (greedy) .* - максимум символов (по умолчанию)
- Ленивый (reluctant) .*? - минимум символов
- Захватывающий (possessive) .*+ - максимум без отката

10. Как проверить, соответствует ли строка регулярному выражению?

С помощью Pattern.matches():

```
String regex = "\\d+";
String text = "123";
boolean matches = text.matches(regex);
// или
boolean matches = Pattern.matches(regex, text);
```

11. Как найти все вхождения подстроки по шаблону?

Используя matcher.find():

```
Pattern pattern = Pattern.compile("\\d+");
Matcher matcher = pattern.matcher("a1b23c456");
while (matcher.find()) {
   System.out.println(matcher.group());
}
```

12. Как разбить строку по регулярному выражению?

```
String[] parts = "a,b c;d".split("[, ;]");
// ["a", "b", "c", "d"]
```

13. Как заменить подстроку по шаблону?

С помощью replaceAll():

String result = "a1b2c3".replaceAll("\\d", "X");
// "aXbXcX"
String result = "hello WORLD".replaceAll("(?i)world", "Java");
// "hello Java"

14.Как экранировать спецсимволы в регулярных выражениях?

- В регулярке: \\ для экранирования спецсимволов: \., \+, *, внутри [] экранировать не нужно.
- В Java-строке: двойное экранирование: \\. для \., так как \ в Java тоже спецсимвол, \\\\ для одного \.

Вывод:

В ходе работы мы изучили регулярные выражения в Java и их применение для работы с текстом и шаблонами.