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»

Лабораторная работа №5

Работа с регулярными выражениниями

Выполнил: Студент группы

БВТ2303

Кунецкий Владислав

Москва

2024

Цели работы:

- Изучить принцип работы регулярных выражений

- Применить на практике полученные знания

Ход работы:

Перед тем как начать выполнять задания лабораторной работы я познакомился с синтаксисом регулярных выражений и запомнил основные конструкции.

После приступил к первому заданию, в котором нужно было написать программу для поиска чисел. В тексте задания уже был дан рабочий код, но он не учитывал все случаи, поэтому я его модифицировал.

Во-первых, было учтено то, что дробные числа могут содержать не только «.» но и «,». Регулярное выражение приобрело вид «\\d+[.,]\\d+». Но данное выражение находит только дробные числа. Добавим условие для чисел без дробной части «(\\d+[.,]\\d+)|(\\d+(?![.,]))». Получается, что регулярное выражение ищет все подряд идущие цифры, разделенные точкой или запятой, или цифры, не заканчивающиеся точкой/запятой.

Во-вторых, числа с более чем одной точкой/запятой не являются числами, поэтому их нельзя учитывать. Я обработал такие случаи как ошибки использовав следующие регулярное выражение «\\d+[.,]\\d+[.,]\\d+».

```java
package lab5;

import java.util.regex.*;

public class NumberFinder {
    Run | Debug
    public static void main(String[] args) {

        String text = "The price of the product is $19,99. The price of the item is $5.99. 2312312312.321312312.0 dasdasdasd 21312dasda22312312 dasd 213123 312312 1321312qwe312312312a2131231";
        Pattern pattern = Pattern.compile(regex:"(\\d+[.,]\\d+)|(\\d+(?![.,]))");
        Pattern errPattern = Pattern.compile("\\d+[.,]\\d+[.,]\\d+");
        Matcher matcher = pattern.matcher(text);
        Matcher errMatcher = errPattern.matcher(text);
        while (errMatcher.find()) {
            try {
                throw new IllegalStateException("Incorrect number format:" + errMatcher.group());
            } catch (IllegalStateException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
        while (matcher.find()) {
                System.out.println(matcher.group());
        }
    }
}
```

Скрин 1 – Программа для поиска чисел

Во втором задание нужно было написать валидатор пароля. Пароль должен состоять из цифр, строчных и заглавных букв, а также иметь длину от 8 до 16 символов.

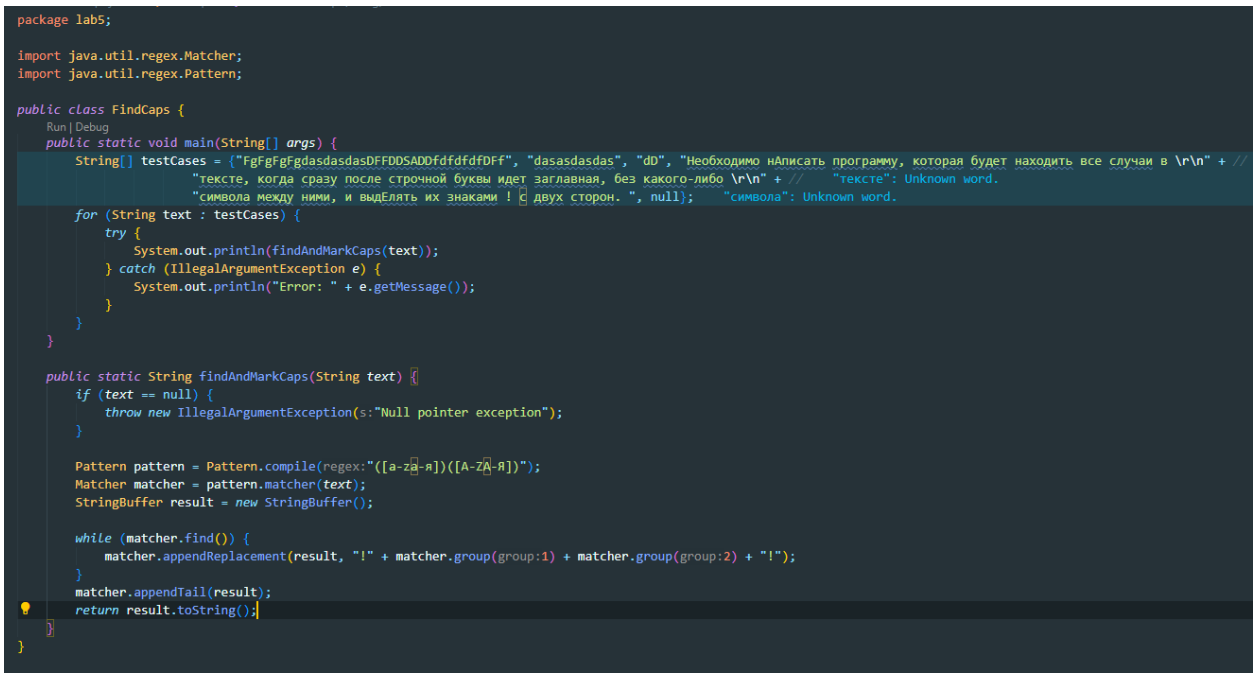
Моя реализация выглядит следующим образом:

```
package lab5;

public class CheckPassword {
    Run | Debug
    public static void main(String[] args) {
        String[] testCases = {"qwerty123", "Qwe123", "QWEwertydasd", "qwerty123", "Qwerty123", "Qwerty123!", "dsassDSAd3123123131231231", null};
        for (String password : testCases) {
            try {
                System.out.println(checkPassword(password));
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    public static boolean checkPassword(String password) {
        if (password == null) {
            throw new IllegalArgumentException(s:"Null pointer exception");
        }
        return password.matches(regex:"^((?=.*[A-Z])(?=.*\\d)[a-z(A-Z)\\d]{8,16})$");
    }
}
```

Скрин 2 – Валидатор пароля

Если в функцию для проверки пароля по какой-то причине пришел нулевой указатель, то она выкидывает исключения. Проверка пароля происходит с помощью регулярного выражения, которое сначала проверят наличие хотя бы одной заглавное буквы и цифры, потом определяет из каких символов может состоять пароль и в конце проверяет длину.

В следующем задании нужно было найти и пометить сочетания пар строчных и заглавных букв, в котором заглавная идет после строчной. Регулярное выражение, подходящее под данный сценарий «([a-za-я])([A-ZA-Я]». Программа ищет все совпадения по данному выражению и добавляет их в результирующую строку, пометив символами «!». В код также была добавлена проверка на нулевой указатель.



```java
package lab5;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class FindCaps {
    Run | Debug
    public static void main(String[] args) {
        String[] testCases = {"FgFgFgFgdasdasdasDFFDDSADDfdfdfdfDFf", "dasasdasdas", "dD", "Необходимо нАписать программу, которая будет находить все случаи в \r\n" + //
                        "тексте, когда сразу после строчной буквы идет заглавная, без какого-либо \r\n" + //     "тексте": Unknown word.
                        "символа между ними, и выдЕлять их знаками ! с двух сторон. ", null};     "символа": Unknown word.
        for (String text : testCases) {
            try {
                System.out.println(findAndMarkCaps(text));
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    public static String findAndMarkCaps(String text) {
        if (text == null) {
            throw new IllegalArgumentException(s:"Null pointer exception");
        }

        Pattern pattern = Pattern.compile(regex:"([a-zа-я])([A-ZА-Я])");
        Matcher matcher = pattern.matcher(text);
        StringBuffer result = new StringBuffer();

        while (matcher.find()) {
            matcher.appendReplacement(result, "!" + matcher.group(group:1) + matcher.group(group:2) + "!");
        }
        matcher.appendTail(result);
        return result.toString();
    }
}
```

Скрин 3 – Программа, отмечающая комбинации символов.

Далее, по 4 заданию, я написал валидатор ipv4 адреса. Он имеет вид 0-255.0-255.0-255.0-255. Для такой строки несложно составить регулярное выражение. Я использовал следующее: «^((\\d{1,3}.){3}\\d{1,3})$». Первые 3 группы имеют от 1 до 3 цифр и точку, а последняя группа только цифры.

```java
package lab5;

import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class IPValidator {
    Run | Debug
    public static void main(String[] args) {
        String[] testCases = {"111.22.32.231", ".322", "0.0.0.0", "", "333.333.333.333", null};
        for (String ip : testCases) {
            try {
                System.out.println(isValidIP(ip));
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }


    static boolean isValidIP(String ip) {
        if (ip == null) {
            throw new IllegalArgumentException(s:"Null pointer exception");
        }
        Pattern pattern = Pattern.compile(regex:"^((\\d{1,3}.){3}\\d{1,3})$");
        Matcher matcher = pattern.matcher(ip);
        if (!matcher.matches()) {
            return false;
        }

        String[] numbers = matcher.group().split(regex:"\\.");
        for (String number : numbers) {
            int octet = Integer.parseInt(number);
            if (octet < 0 || octet > 255) {
                return false;
            }
        }
        return true;
    }

}
```

Скрин 4 – Валидатор IP.

В последнем задании нужно было найти все слова в тексте, начинающиеся с заданной буквы. Использования регулярных выражение упрощает решение задачи. Я воспользовался такой конструкцией «\\b{*letter*}\\p{L}*\\b». Сначала данное выражение проверяет начало слова, потом проверяет наличие буквы, далее доходит до конца слова, учитывая, что в слове присутствуют другие буквы. Итоговый код:

```java
package lab5;

import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class FindWords {
    Run | Debug
    public static void main(String[] args) {
        String[] testCases = {"Hello, my name is John", "I am a student", "Необходимо написать программу", "При этом программа должна использовать регулярные выражения", "", null};
        for (String text : testCases) {
            try {
                System.out.println(findWords(text, letter:'a'));
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }


    static String findWords(String text, char letter) {
        if (text == null) {
            throw new IllegalArgumentException(s:"Null pointer exception");
        }
        if (!Character.isLetter(letter)) {
            throw new IllegalArgumentException(s:"Illegal letter format");
        }

        String formatedText = text.toLowerCase();    "formated": Unknown word.
        letter = Character.toLowerCase(letter);

        StringBuilder result = new StringBuilder();

        String regex = "\\b" + letter + "\\p{L}*\\b";
        Pattern pattern = Pattern.compile(regex, Pattern.UNICODE_CHARACTER_CLASS);
        Matcher matcher = pattern.matcher(formatedText);    "formated": Unknown word.
        while (matcher.find()) {
            result.append(matcher.group()).append(str:" ");
        }

        if (result.isEmpty()){
            return "No words with " + letter + " letters found";
        }

        return result.toString();
    }

}
```

Скрин 5 – Программа для нахождения слов