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

По дисциплине «Информационные технологии и программирование»

Выполнил: Студент группы

БПИ 2301

Антонова Ирина

Москва

2024

Цель:

Изучение и применение регулярных выражений в Java

Задание:

Задание 1: Поиск всех чисел в тексте

Необходимо 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.

Пример реализации метода:

```
import java.util.regex.*;
public class NumberFinder {
	public static void main(String[] args) {
		String text = "The price of the product is $19.99";
		Pattern pattern = Pattern.compile("\\d+\\.\\d+");
		Matcher matcher = pattern.matcher(text);
		while (matcher.find()) {
			System.out.println(matcher.group());
		}
	}
}
```

Задание 2: Проверка корректности ввода пароля

Необходимо 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.

Задание 3: Поиск заглавной буквы после строчной

Необходимо 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.

Задание 4: Проверка корректности ввода IP-адреса

Необходимо 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.

Задание 5: Поиск всех слов, начинающихся с заданной буквы

Необходимо 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

Ход работы:

Программы должны использовать классы Pattern и Matcher из пакета java.util.regex. Класс Pattern необходим для компиляции регулярного выражения, а Matcher — для поиска и обработки совпадений в тексте.

Класс Pattern

В классе Pattern конструкторы не определяются. Вместо этого для составления шаблона вызывается фабричный метод compile(). Ниже приведена одна из общих форм этого метода.

Класс Matcher

У этого класса отсутствуют конструкторы. Вместо этого для создания объекта класса Matcher вызывается фабричный метод matcher( ), определяемый в классе Pattern. Как только объект класса Matcher будет создан, его методы можно использовать для выполнения различных операций сопоставления с шаблоном.

- boolean matches() – метод, который определяет совпадает ли последовательность символов с шаблоном.
- boolean find() - находит хотя бы одно совпадение в тексте.
- String group() - находит последнюю совпавшую последовательность.

Откроем папку LR5 в программе VSCode и создадим файл Main.java, в нем импортируем нужные публичный класс с тем же именем.

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

public class Main{
```

Метод main:

Метод main является точкой входа в программу.

Внутри метода main вызываются различные методы для проверки регулярных выражений.

Результаты выполнения методов выводятся на консоль.

Обработка исключений PatternSyntaxException для случаев, когда регулярное выражение имеет некорректный синтаксис.

```
public static void main(String[] args) {
        try{
        System.out.println(NumberFinder("The1 quic4k b7rown fox jumps over the
lazy3 dog"));
        System.out.println(CorrectPassword("pAssword"));
        System.out.println(CorrectPassword("pA1ssword"));
        System.out.println(CorrectPassword("pssworddd"));
        System.out.println(CorrectPassword("a111111111"));
        System.out.println(UppercaseAfterLowercase("aaaWbbzSb"));
        System.out.println(FindWord("Hello my dear friends today IS morning
Figa", "f"));
        System.out.println(CorrectIP("1.243.54.0"));
        System.out.println(CorrectIP("1.263.54.0"));
        }
        catch(PatternSyntaxException e){
            System.out.println("Incorrect template syntax"+ e);
        }
    }
```

Метод NumberFinder:

Метод NumberFinder принимает строку text и возвращает строку, содержащую все числа, найденные в text.

Регулярное выражение \\d+ используется для поиска последовательностей цифр.

Pattern.compile("\\d+") компилирует регулярное выражение.

Matcher matcher = pattern.matcher(text) создает объект Matcher для поиска совпадений в строке text.

Цикл while (matcher.find()) ищет все совпадения и добавляет их к строке res.

Метод возвращает строку res, содержащую все найденные числа.

```
//№1
    public static String NumberFinder(String text){
        Pattern pattern = Pattern.compile("\\d+");
        Matcher matcher = pattern.matcher(text);
        String res = "";
        while (matcher.find()) {
            res += matcher.group();
        }
        return res;
    }
```

Метод CorrectPassword:

Метод CorrectPassword принимает строку text и проверяет, соответствует ли она требованиям пароля.

Регулярное выражение [a-z](?=.*[A-Z])(?=.*[0-9]){8,16} проверяет, что пароль содержит хотя бы одну строчную букву, одну заглавную букву и одну цифру, и его длина от 8 до 16 символов.

Pattern.compile("[a-z](?=.*[A-Z])(?=.*[0-9]){8,16}") компилирует регулярное выражение.

Matcher matcher = pattern.matcher(text) создает объект Matcher для поиска совпадений в строке text.

Метод matcher.find() проверяет, соответствует ли строка регулярному выражению.

Метод возвращает "Correct Password!!!" или "Incorrect Password!!!" в зависимости от результата проверки.

```
//№2
    public static String CorrectPassword(String text){
        Pattern pattern = Pattern.compile("[a-z](?=.*[A-Z])(?=.*[0-9]){8,16}");
// положительный просмотр вперед
        Matcher matcher = pattern.matcher(text);
        if (matcher.find()){
            return "Correct Password!!!";
        } else {
            return "Incorrect Password!!!";
        }
    }
```

Метод UppercaseAfterLowercase:

Метод UppercaseAfterLowercase принимает строку text и заменяет все последовательности, где строчная буква следует за заглавной буквой, на строку с восклицательными знаками.

Регулярное выражение ([a-z])([A-Z]) ищет последовательности, где строчная буква следует за заглавной буквой.

Pattern.compile("([a-z])([A-Z])") компилирует регулярное выражение.

Matcher matcher = pattern.matcher(text) создает объект Matcher для поиска совпадений в строке text.

Метод matcher.replaceAll("!$1$2!") заменяет все найденные последовательности на строку с восклицательными знаками.

Метод возвращает измененную строку res.

```
//№3
    public static String UppercaseAfterLowercase(String text) {
        Pattern pattern = Pattern.compile("([a-z])([A-Z])");
        Matcher matcher = pattern.matcher(text);
        String res = matcher.replaceAll("!$1$2!");
        return res;
    }
```

Метод CorrectIP:

Метод CorrectIP принимает строку text и проверяет, является ли она корректным IP-адресом.

Регулярное выражение ^((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])$ проверяет, что IP-адрес состоит из 4 чисел, разделенных точками, и каждое число находится в диапазоне от 0 до 255.

Pattern.compile("^((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])$") компилирует регулярное выражение.

Matcher matcher = pattern.matcher(text) создает объект Matcher для поиска совпадений в строке text.

Метод matcher.find() проверяет, соответствует ли строка регулярному выражению.

Метод возвращает "Correct IP" или "Incorrect IP" в зависимости от результата проверки.

```
//№4
    public static String CorrectIP(String text) {
        Pattern pattern = Pattern.compile("^((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-
9]?[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])$");
        Matcher matcher = pattern.matcher(text);
        if (matcher.find()){
            return "Correct IP";
        } else {
            return "Incorrect IP";
        }
    }
```

Метод FindWord:

Метод FindWord принимает строку text и строку f, и возвращает все слова, начинающиеся с f, найденные в text.

Регулярное выражение (?i)\\b" + f + "\\w*\\b ищет слова, начинающиеся с f, игнорируя регистр.

Pattern.compile("(?i)\\b" + f + "\\w*\\b") компилирует регулярное выражение.

Matcher matcher = pattern.matcher(text) создает объект Matcher для поиска совпадений в строке text.

Цикл while (matcher.find()) ищет все совпадения и добавляет их к строке res.

Метод возвращает строку res, содержащую все найденные слова.

```
//№5
    public static String FindWord(String text, String f) {
        Pattern pattern = Pattern.compile("(?i)\\b" + f + "\\w*\\b");
        Matcher matcher = pattern.matcher(text);
        String res = "";
        while (matcher.find()){
            res += matcher.group() + " ";
        }
        return res;
    }
```

Вывод:

В лабораторной работе были использованы регулярные выражения, это шаблон для поиска строки в тексте. Регулярные выражения являются уникальным и удобным способом взаимодействия со строками, удаление, замена, нахождение нужных символов при помощи создания шаблонов. В ходе работы был использован класс Pattern для составления шаблонов, то есть регулярного выражения, которые при помощи статического метода compile() преобразует в шаблон символьную строку. Этот метод возвращает объект типа Pattern, содержащий шаблон. Также был использован класс Matcher, для создания объекта Matcher вызывается метод matcher(), определяемый в классе Pattern. После создания объекта Matcher при помощи его методов (matches(), find(), group() и т.д.) можно использовать для выполнения различных операций сопоставления с шаблоном.

Ссылка на репозиторий с кодом
https://github.com/k00kzaAntonovaIra/ITIP_2024.git