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»

Отчёт по Лабораторной работе №5

По дисциплине: «Информационные технологии и программирование»

«Строки и регулярные выражения»

Выполнила: студентка группы БПИ2401

Алексеева Татьяна Игоревна

Проверил: Харрасов Камиль Раисович

Москва

2025

Цель работы

Освоить принципы работы со строками в языке программирования Java, изучить их особенности (неизменяемость, интернирование, представление символов Unicode), а также приобрести навыки использования регулярных выражений для поиска, анализа и обработки текстовой информации.

Выполнение 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.

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

public class FindAllDigits {
    public static void main(String[] args) {
        String text = "The price is $19.99, discount 5%, amount: 120 and 0.75 liters.
Spent: -40$";
        Pattern pattern = null;

        try {
            pattern = Pattern.compile("\\d+(\\.\\d+)?");
        } catch (PatternSyntaxException e) {
            System.out.println("ошибка в регулярном выражении" + e.getDescription());
            return;
        }

        try {
            Matcher matcher = pattern.matcher(text);

            while (matcher.find()) {
                System.out.println(matcher.group());
            }
        } catch (Exception e) {
            System.out.println("Ошибка обработки текста" + e.getMessage());
        }
    }
}
```

Для начала подключаем классы для работы с регулярными выражениями (Pattern, Matcher). Объявляем переменную для скомпилированного шаблона регулярного выражения.

В боке try-catch компилируем строку-шаблон в объект Pattern. Регулярка записана как Java-строка \\d+(\\.\\d+)?. Проверяем на корректность синтаксис регулярного выражения.

В следующем блоке try-catch создаём объект Matcher, который применяет ранее скомпилированный pattern к конкретной входной строке. В цикле while ищем совпадения в троке с помощью matcher.find(), если найдено, возвращаем true. Каждый вызов matcher.group() возвращает строку с текущим совпадением, которое выводим в консоль.

```
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkFive>  c:; cd 'c:
:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\taale\AppDat
AllDigits'
19.99
5
120
0.75
40
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkFive>
```

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.

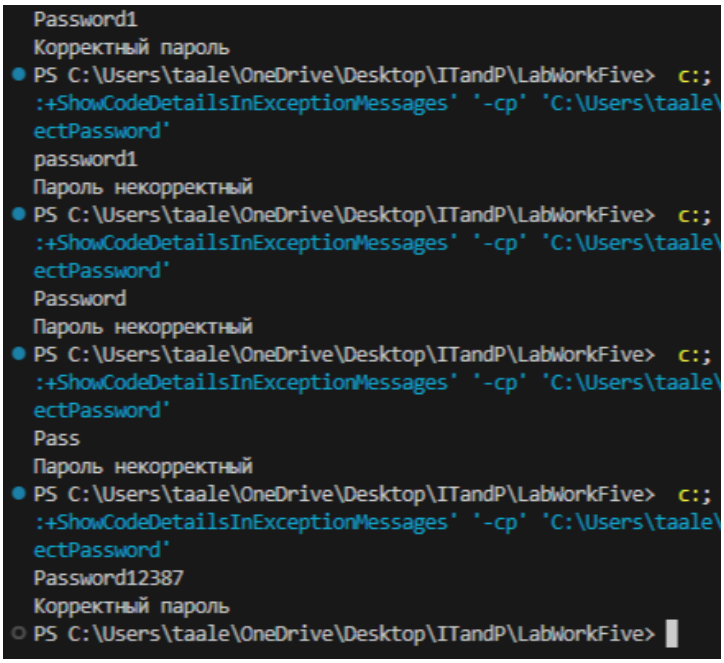
```java
import java.util.regex.*;
import java.util.Scanner;

public class CorrectPassword {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String password = s.nextLine();
        s.close();

        try {
            if (checkPassword(password)) {
                System.out.println("Корректный пароль");
            } else {
                System.out.println("Пароль некорректный");
            }
        } catch (Exception e) {
            System.out.println("Произошла ошибка при проверке пароля" + e.getMessage());
        }
    }

    public static boolean checkPassword(String pass) {
        Pattern pattern = null;
        try {
            pattern = Pattern.compile("^(?=.*[A-Z])(?=.*[0-9])[A-Za-z0-9]{8,16}$");
        } catch (PatternSyntaxException e) {
            System.out.println("Ошибка в регулярном выражении");
            return false;
```

```
        }

        Matcher matcher = pattern.matcher(pass);
        return matcher.matches();
    }
}
```

Импортируем классы для работы с регулярными выражениями, а также Scanner для чтения пароля с клавиатуры.

В блоке try-catch защищаемся от неожиданных исключений при проверке пароля.

Прописываем метод checkPassword(), который проверяет пароль и возвращает или true, или false. Здесь мы компилируем регулярное выражение-шаблон. Далее получаем Matcher и проверяем, совпадает ли вся строка с шаблоном (метод matcher() требует полного совпадения со всем шаблоном).



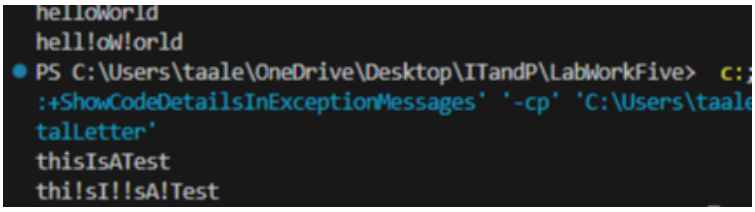
Задание 3. Поиск заглавной буквы после строчной
Написать программу, которая будет находить все случаи в тексте, когда сразу после строчной буквы идет заглавная без какого-либо символа между ними и выделять их знаками «!» с двух сторон.

```
import java.util.regex.*;
import java.util.Scanner;

public class CapitalLetter {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String text = s.nextLine();
```

```
        s.close();

        Pattern pattern = Pattern.compile("([a-z][A-Z])");
        Matcher matcher = pattern.matcher(text);
        String result = matcher.replaceAll("!$1!");
        System.out.println(result);
    }
}
```

Подключаем классы для работы с регулярными выражениями, Scanner для чтения строки с клавиатуры.

Далее компилируем регулярное выражение в объект Pattern, создаем Matcher: он применяет шаблон к строке. Для каждого совпадения шаблона заменяем найденный фрагмент на !$1!, где $1 – это захваченная в первой скобочной группе подстрока (в нашем случае пара символов – строчная + заглавная). Метод replaceAll возвращает новую строку с произведёнными заменами.



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.

```
import java.util.regex.*;
import java.util.Scanner;

public class CorrectIPAddress {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String ip_address = s.nextLine().trim();

        try {
            if (isRightIPAddress(ip_address)) {
                System.out.println("Корректный IP-адрес");
            } else {
                System.out.println("Некорректный IP-адрес");
            }
        } catch (Exception e) {
            System.out.println("Ошибка при проверки IP-адреса" + e.getMessage());
        }
```

```
        s.close();
    }

    public static boolean isRightIPAddress(String ip) {
        Pattern pattern = null;

        try {
            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])$");
        } catch (PatternSyntaxException e) {
            System.out.println("Ошибка в регулярном выражении" + e.getDescription());
            return false;
        }

        Matcher matcher = pattern.matcher(ip);
        return matcher.matches();
    }
}
```

Импортируем классы для работы с регулярными выражениями, испортируем Scanner для чтения ввода пользователя.

Прописываем метод isRightIPAddress(), который будет выполнять проверку IP-адреса. Здесь мы компилируем шаблон регулярного выражения, создаём объект Matcher, применяющий шаблон к строке, метод matcher() возвращает true, если вся строка целиком совпадает с шаблоном.

```
127.0.0.1
Корректный IP-адрес
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkFive>  c:; cd 'c:
:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\taale\AppDat
ectIPAddress'
256.100.0.1
Некорректный IP-адрес
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkFive>  c:; cd 'c:
:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\taale\AppDat
ectIPAddress'
192.168.1
Некорректный IP-адрес
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkFive>  c:; cd 'c:
:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\taale\AppDat
ectIPAddress'
192.168.01.1
Некорректный IP-адрес
```

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.

```
import java.util.regex.*;
import java.util.Scanner;

public class FindWord {
```

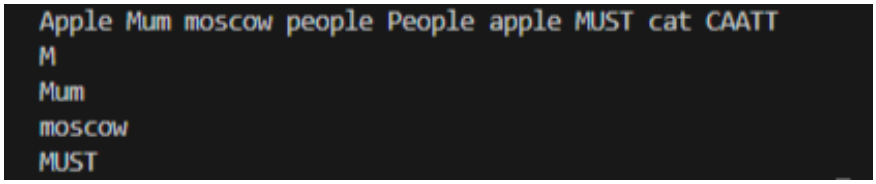
```
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String text = s.nextLine();
        String letter = s.nextLine();
        s.close();

        try {

            if (text.isEmpty()) {
                System.out.println("Нет текста для проверки");
                return;
            }

            if (letter.length() != 1 || !Character.isLetter(letter.charAt(0))) {
                throw new IllegalArgumentException("Введите ровно одну букву");
            }

            Pattern pattern = null;
            try {
                pattern = Pattern.compile("\\b" + letter + "\\w*\\b",
Pattern.CASE_INSENSITIVE);
            } catch (PatternSyntaxException e) {
                System.out.println("Ошибка в регулярном выражении" +
e.getDescription());
            }

            Matcher matcher = pattern.matcher(text);
            boolean flag = false;
            while (matcher.find()) {
                System.out.println(matcher.group());
                flag = true;
            }

            if (!flag) {
                System.out.println("Слов, начинающихся на эту букву, не найлено");
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Импортируем классы для работы с регулярными выражениями, Scanner для считывания ввода пользователя (текст и буква).

Проверяем, не ввели ли пустой текст, и корректность введённой буквы (должна быть одна буква и символ должен быть буквой по Character.isLetter(). Если нет бросаем исключение.

Далее компилируем шаблон регулярного выражения, создаём объект Matcher, который будет искать совпадения по этому шаблону в строке.

Цикл while находит все неперекрывающиеся совпадения шаблона в тексте. matcher.group() возвращает текст найденного совпадения.



Вывод

В ходе выполнения лабораторной работы были изучены и применены на практике различные возможности работы со строками и регулярными выражениями в языке Java. В процессе решения заданий были рассмотрены базовые и расширенные правила построения регулярных выражений, а также механизмы их безопасного использования через классы Pattern и Matcher.

Ответы на контрольные вопросы

1. String – это класс в Java, который представляет последовательность символов (текст). На практике это текстовые данные: слова, предложения и т.д. В Java строки являются объектами, а не примитивными типами. Строки неизменяемы. Уникальные строковые литералы хранятся в специальной области памяти – куче.
2. Строки в Java иммутабельны (неизменяемы), потому что после создания их содержимое нельзя изменить. Любая операция, которая кажется изменением строки, на самом деле создаёт новый объект класса String. Плюсы: безопасность и возможность кэширования строк, хэширование для хэш-таблиц.
3. Интернирование – это процесс, при котором компилятор и виртуальная машина JVM пытаются минимизировать количество дублирующихся строковых литералов путём сохранения их в специальном пуле (pool). Когда создаётся новый строковый литерал, JVM сначала проверяет, есть ли уже такая строка в пуле. Если да, то возвращается ссылка на существующий объект вместо создания нового.
4. String – иммутабельный, не применяется в обеспечении потокобезопасности, используется, когда строки редко изменяются. StringBuilder – изменяемый, нет потокобезопасности, применяется, когда строки часто изменяются и нужна скорость. StringBuffer – изменяемый, есть потокобезопасность, применяется в многопоточных приложениях, где нужна безопасность и строки меняются одновременно.

5. == - сравнивает ссылки на объект, а не их содержимое
   equals() – сравнивает содержимое строк с учетом регистра
   equalsIsIgnoreCase() – сравнивает содержимое без учёта регистра.
6. Строки хранятся в heap-memory. Литералы строк JVM хранит в String Pool – специальной области, где для одинаковых литералов создаётся один объект, чтобы экономить память и ускорять сравнение.
7. Code unit – минимальная единица хранения символа в кодировке UTF-16 (16 бит)
   Code point – фактический Unicode-символ. Для символа за пределами BSM требуется 2 code unit, чтобы представить один code point.
8. Чтобы использовать регулярные выражения в Java, нужно импортировать пакет java.util.regex и использовать классы Pattern (шаблон регулярного выражения) и Matcher (объект для поиска по шаблону).
9. Квантификаторы определяют количество повторений символа или группы: жадный (greedy) – пытается найти максимально длинное совпадение (.*), ленивый (reluctant) – начинает с начала строки, добавляя символ за символом, пока не найдется совпадение, сверхжадная – просматривает сразу всю строку единожды, не убирая символы, как в жадной.
10. Для проверки соответствия строки регулярному выражению в Java можно использовать метод matcher() класса String() или комбинацию Pattern и Matcher. Метод matcher() возвращает true, если вся строка полностью соответствует заданному шаблону. Если требуется более гибкая проверка, например поиск подстрок, используется Pattern для компиляции регулярного выражения и Matcher для поиска совпадений в строке.
11. Можно использовать Matcher.find(), который последовательно извлекает все совпадения, каждый вызов find() возвращает следующее совпадение до конца строки, и Matcher.group(), который возвращает текст текущего совпадения, найденного последним вызовом find()
12. Для разделения строки по шаблону используется метод split() класса String. Он принимает регулярное выражение в качестве разделителя и возвращает массив строк, которые получаются при разделении исходной строки по совпадениям с шаблоном.
13. Замена подстрок по регулярному выражению выполняется с помощью методов replaceAll() (заменяет все совпадения шаблона в строке) и replaceFirst() (заменяет только первое совпадение) класса String.
14. Чтобы экранировать спецсимволы в регулярных выражения используют двойной слеш \\.