МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 5

Выполнил: студент группы БПИ2401
Трухина Анастасия Александровна
Проверил: Харрасов Камиль Раисович

Москва,
2025

Оглавление

Цель работы: Изучение и практическое применение средств обработки строк и регулярных выражений в языке Java для решения типовых задач текстовой обработки данных.

Задание:.
Задание 1. Поиск всех чисел в тексте Написать программу, которая будет искать все числа в за данном тексте и выводить их на экран. При этом программа должна использовать регулярные выражения для поиска чисел и обрабатывать возможные ошибки. Пример реализации такого функционала представлен на листинге 5.1.

Листинг 5.1. Класс NumberFinder 1. import java.util.regex.*; 2. 3. public class NumberFinder { 4. public static void main(String[] args) { 5. 6. 7. 8. 9. 10. 11. 12. } 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 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 долж на использовать регулярные выражения для проверки пароля и обрабатывать возможные ошибки.

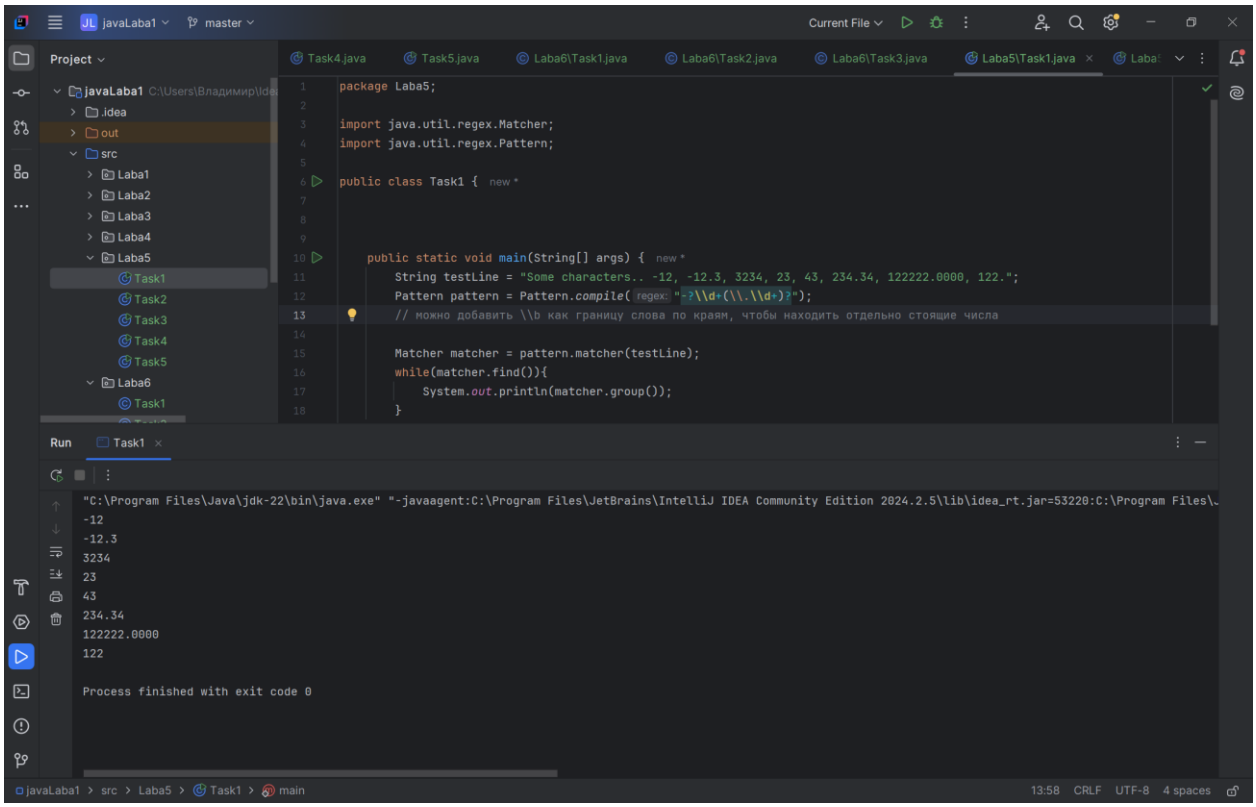
Задание 3. Поиск заглавной буквы после строчной Написать программу, которая будет находить все случаи в тексте, когда сразу после строчной буквы идет заглавная без какого-либо символа между ними и выделять их знаками «!» с двух сторон.

Задание 4. Проверка корректности ввода IP-адреса Написать программу, которая будет проверять корректность ввода IP-адреса. IP-адрес должен состоять из 4 чисел, разде ленных 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. 42 Раздел 5

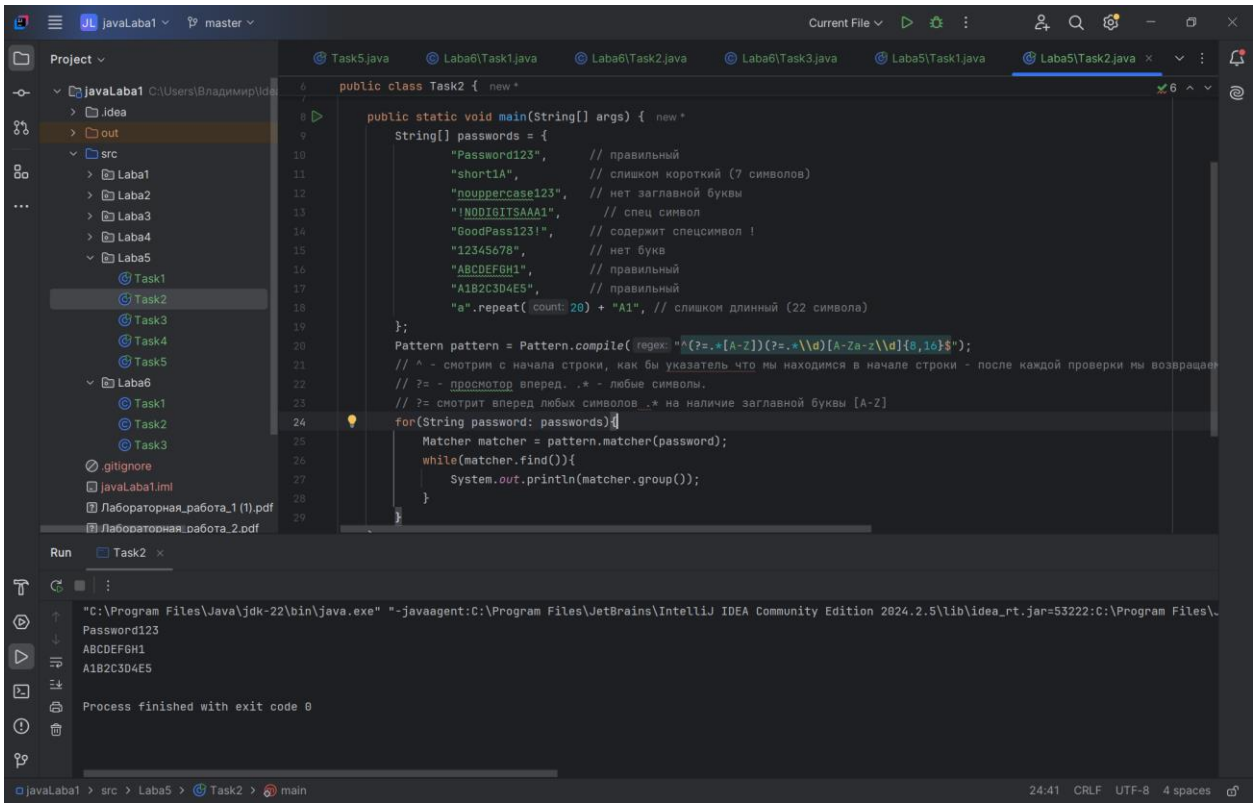
Задание 5. Поиск всех слов, начинающихся с задан ной буквы Написать программу, которая будет искать все слова в за данном тексте, начинающиеся с заданной буквы, и выводить их на экран. При этомпрограммадолжнаиспользоватьрегулярные выражения для поиска слов и обрабатывать возможные ошибки.

Основная часть

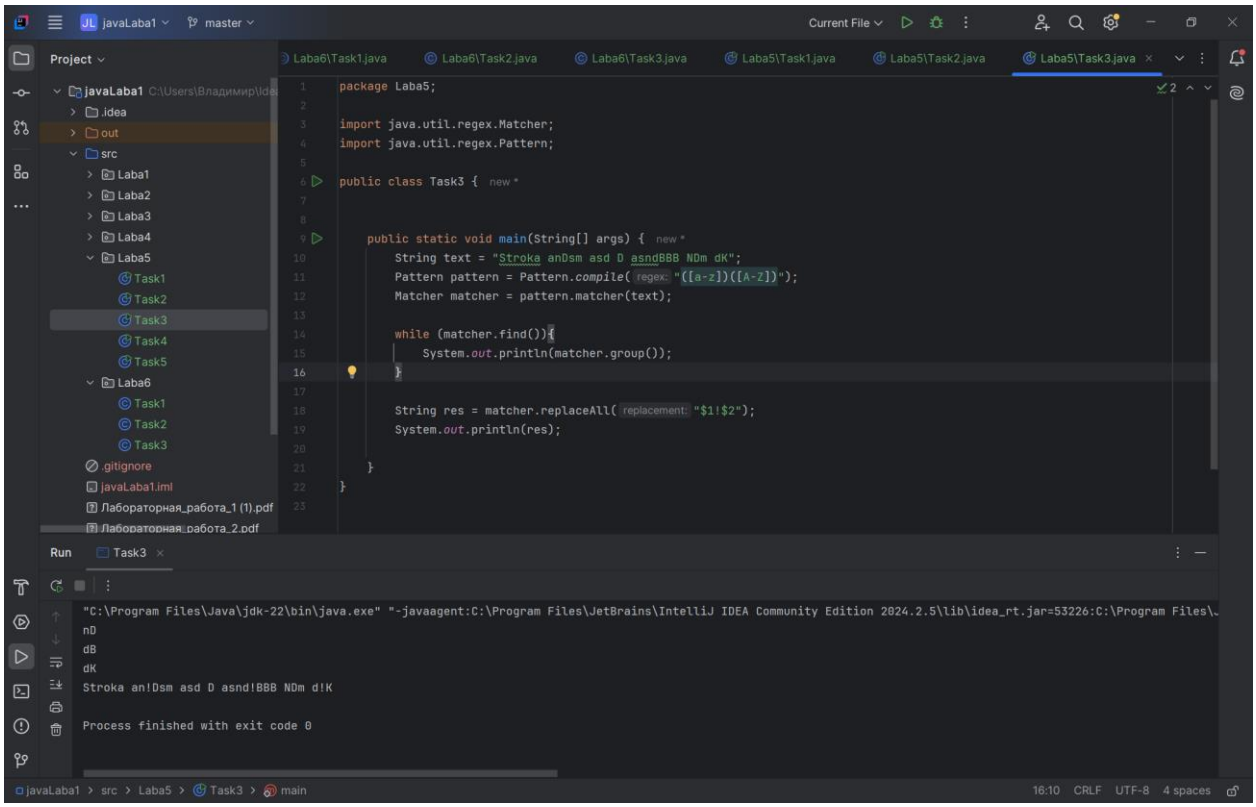
Задание 1:



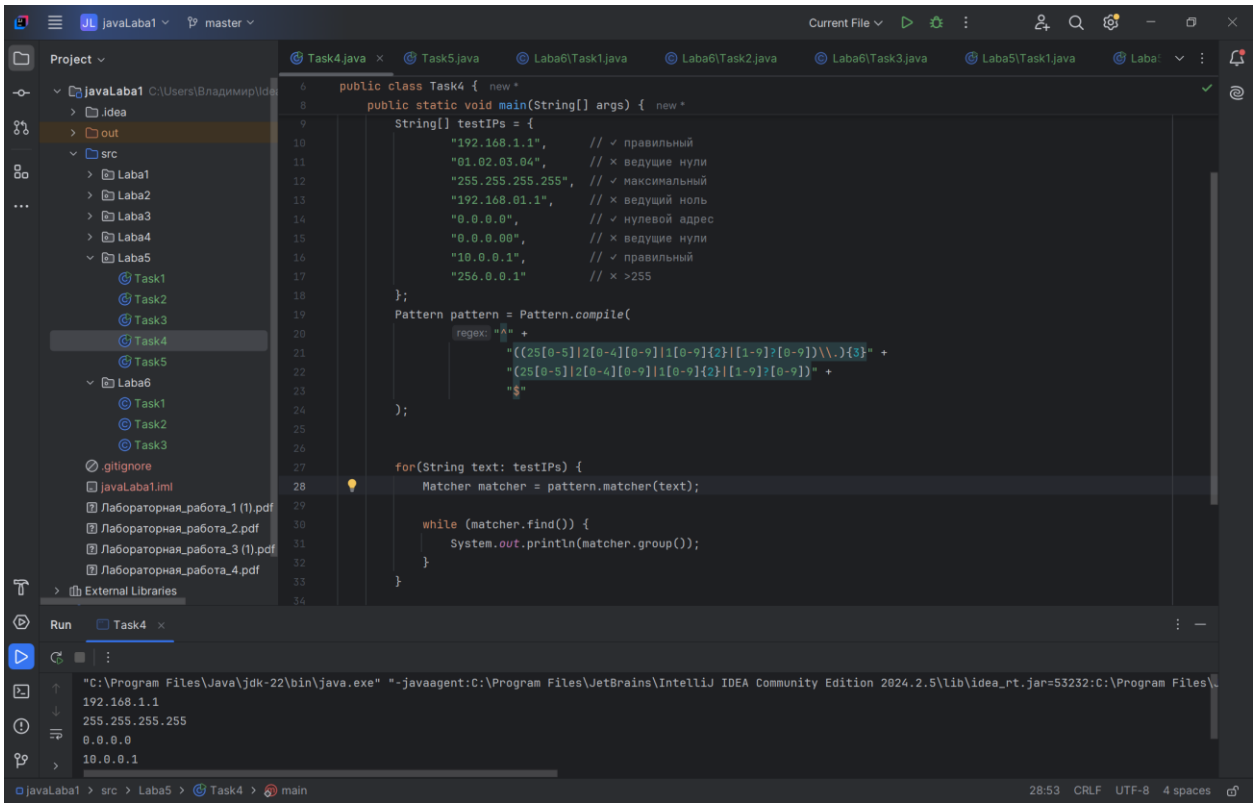
Задание 2



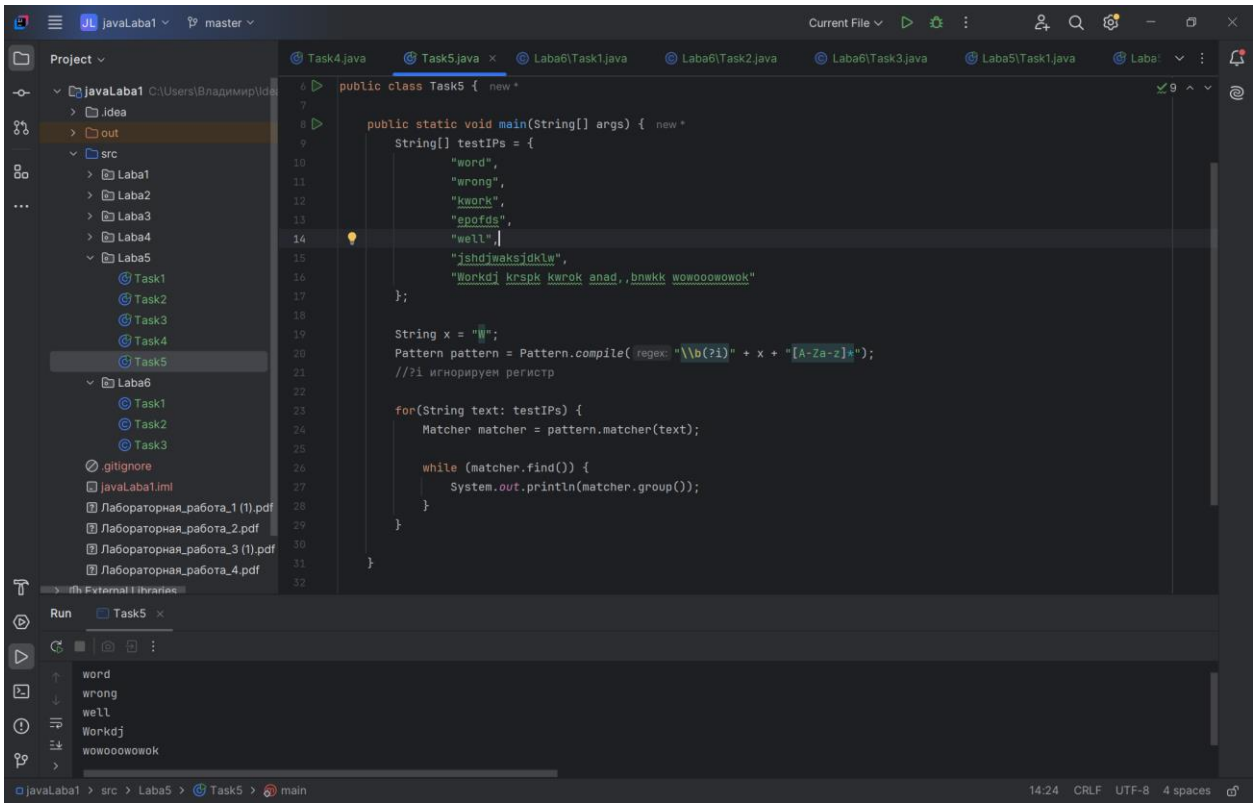
Задание 3



Задание 4



Задание 5



Заключение

Вывод:

В ходе выполнения лабораторной работы №5 «Строки и регулярные выражения» были успешно изучены и применены на практике ключевые возможности языка Java для работы с текстовыми данными. Была реализована серия программ, решающих различные прикладные задачи: от поиска числовых данных в тексте до сложной валидации структур данных, таких как пароли и IP-адреса.

Ответы на контрольные вопросы:

Вопрос 1. Что такое класс String? Класс String в Java представляет собой неизменяемую последовательность символов Unicode. Он предоставляет множество методов для выполнения операций над строками, таких как сравнение, поиск, извлечение подстрок, замена и разбиение. Все объекты String хранятся в пуле строк, что позволяет оптимизировать использование памяти за счет повторного использования существующих литералов.

Вопрос 2. Почему объект класса String является иммутабельным? Неизменяемость объекта String означает, что после создания его состояние изменить нельзя. Любая операция, которая, казалось бы, изменяет строку, на самом деле создает новый объект String. Это обеспечивает ряд ключевых преимуществ: безопасность в многопоточных средах, возможность кэширования хэш-кода для повышения производительности и

безопасное использование строк в качестве ключей в коллекциях. Неизменяемость также упрощает проектирование и делает поведение класса предсказуемым.

Вопрос 3. Что такое интернирование строк? Интернирование строк — это процесс, при котором JVM для экономии памяти сохраняет в специальной области heap, называемой String Pool, только одну копию каждого уникального строкового литерала. Когда создается новый строковый литерал, JVM сначала проверяет его наличие в пуле. Если такой литерал уже существует, возвращается ссылка на существующий объект, а не создается новый. Это происходит автоматически для строковых литералов, объявленных в коде в двойных кавычках. Для строк, созданных с помощью оператора new или динамически, можно вручную добавить строку в пул, вызвав метод intern(), который возвращает каноническое представление строки. Например, после String s2 = new String("Hello").intern(); и String s1 = "Hello"; выражение s1 == s2 вернет true, так как обе переменные ссылаются на один и тот же объект в пуле.

Вопрос 4. В чем разница между String, StringBuilder и StringBuffer? Основное различие заключается в изменяемости и потокобезопасности. String — неизменяемый класс. Любая модификация создает новый объект, что может быть неэффективно при интенсивных операциях со строками в циклах. StringBuilder — изменяемый класс, предназначенный для эффективного построения строк. Он не является синхронизированным, поэтому работает быстрее, но не безопасен для использования из нескольких потоков одновременно. StringBuffer — также изменяемый класс, аналогичный по функциональности StringBuilder, но его методы синхронизированы. Это делает его потокобезопасным, но немного менее производительным по сравнению с StringBuilder. Следует использовать StringBuilder для однопоточных программ и StringBuffer только в случаях, когда необходима безопасность в многопоточной среде.

Вопрос 5. Как сравнить две строки? В чем разница между ==, equals() и equalsIgnoreCase()? Оператор == сравнивает ссылки двух объектов, а не их содержимое. Он возвращает true, только если обе переменные указывают на один и тот же объект. Метод equals(Object anObject) сравнивает содержимое строк посимвольно. Именно его следует использовать для логического сравнения строковых значений. Этот метод переопределен в классе String. Метод equalsIgnoreCase(String anotherString) аналогичен equals(), но игнорирует разницу между заглавными и строчными буквами. Например, "Hello".equalsIgnoreCase("heLLo") вернет true.

Вопрос 6. Как хранятся строки в памяти? Что такое пул строк? Строки хранятся в heap. String Pool — это специальная область heap, созданная JVM для хранения уникальных литералов строк. Его цель — оптимизация использования памяти. Когда компилятор встречает строковый литерал, он проверяет его наличие в пуле. Если строка найдена,

компилятор возвращает ссылка на существующий объект. В противном случае новый объект String создается в пуле, и его ссылка возвращается. Этот механизм не распространяется автоматически на строки, созданные с помощью оператора new String().

Вопрос 7. Что такое code unit и code point? Code unit — это минимальная битовая комбинация, используемая для представления символа в конкретной кодировке. В Java тип char и внутреннее представление String используют кодировку UTF-16, где code unit имеет размер 16 бит. Для большинства распространенных символов этого достаточно. Code point — это уникальное числовое значение, присвоенное каждому символу в стандарте Unicode. Один code point может быть представлен одним или двумя code units в UTF-16. Символы за пределами Basic Multilingual Plane, например, многие эмодзи, требуют для своего хранения пары char. Метод length() возвращает количество code units, а метод codePointCount() — количество code points в строке.

Вопрос 8. Что необходимо для использования регулярных выражений в Java? Для работы с регулярными выражениями в Java используется пакет java.util.regex. Два ключевых класса в нем: Pattern — представляет собой скомпилированное представление регулярного выражения. Для создания объекта используется статический метод Pattern.compile(String regex). Matcher — интерпретирует шаблон и выполняет операции сопоставления с входной строкой. Получить Matcher можно с помощью метода pattern.matcher(CharSequence input). Базовый цикл работы: скомпилировать шаблон в Pattern, создать Matcher для нужной строки, затем использовать методы Matcher для поиска и извлечения результатов.

Вопрос 9. Какие есть режимы работы квантификатора? Квантификаторы в регулярных выражениях определяют, сколько раз должен встретиться предшествующий элемент. Основные квантификаторы: * — ноль или более раз; + — один или более раз; ? — ноль или один раз; {n} — ровно n раз; {n,} — n или более раз; {n,m} — от n до m раз. Квантификаторы по умолчанию работают в жадном режиме: они пытаются захватить максимально возможное количество символов. Добавление знака ? после квантификатора меняет его поведение на ленивый режим: ищет минимально возможное совпадение. Добавление знака + меняет поведение на захватывающий режим.

Вопрос 10. Как проверить, соответствует ли строка регулярному выражению? Для проверки полного соответствия всей строки шаблону используется метод matches() класса Matcher или его сокращенная форма — статический метод Pattern.matches(String regex, CharSequence input). Эти методы возвращают true только если вся строка от начала до конца соответствует регулярному выражению. Для поиска частичного совпадения внутри строки используется метод find().

Вопрос 11. Как найти все вхождения подстроки по шаблону? Для этого применяется цикл с использованием метода find() объекта Matcher. Метод find() ищет следующее вхождение шаблона в строке, начиная с конца предыдущего совпадения. Внутри цикла с помощью метода group() можно получить найденную подстроку, а start() и end() — ее позиции.

Вопрос 12. Как разбить строку по регулярному выражению? Для разбиения строки на массив подстрок используется метод split(String regex) самого класса String или метод split(CharSequence input, int limit) класса Pattern. Метод String.split() внутри создает Pattern и вызывает его метод split.

Вопрос 13. Как заменить подстроку по шаблону? Для замены используются методы replaceAll(String replacement) и replaceFirst(String replacement) класса Matcher, а также аналогичные методы replaceAll() и replaceFirst() в классе String, которые принимают регулярное выражение первым аргументом. В строке замены replacement можно использовать ссылки на захваченные группы вида $1, $2 и т.д.

Вопрос 14. Как экранировать спецсимволы в регулярных выражениях? В синтаксисе regex многие символы имеют специальное значение: . \ [ ] { } ( ) * + ? ^ $ |. Чтобы использовать их как обычные символы, необходимо экранировать их обратным слэшем \. Однако в Java-строке сам обратный слэш также требует экранирования. Поэтому в литерале строки Java нужно ставить двойной слэш \\. Например, для поиска точки в строке шаблон будет выглядеть как "\\\.", а для поиска обратного слэша — как "\\\\". Для экранирования всей строки, которую планируется использовать в regex, можно воспользоваться методом Pattern.quote(String s), который оборачивает строку в \Q...\E.

ГитХаб с файлами кода: NT-005-TN/ITiP_LabWorks_Trukhina