**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования

**«Московский технический университет связи и информатики»**

**ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

Кафедра «МКиИТ»

**Информационные технологии и**

**программирование**

**Лабораторная работа по ИТП №5**

Выполнил: студент группы БВТ2205

Белов Егор Олегович

import java.util.regex.\*;  
  
//Задание: 1  
  
public class NumberFinder {  
 public static void main(String[] args) {  
 String text = "The price20.45 of the product is $19.99. It is 3times more 5 5.4 than the previous price.";  
 try {  
 Pattern pattern = Pattern.*compile*("\\d+([.]\\d+)?");  
 Matcher matcher = pattern.matcher(text);  
 while (matcher.find()) {  
 System.*out*.println(matcher.group());  
 }  
 } catch (PatternSyntaxException e) {  
 System.*out*.println("Некорректный синтаксис регулярного выражения: " + e.getMessage());  
 } catch (IllegalStateException e) {  
 System.*out*.println("Проверь последовательность методов: " + e.getMessage());  
 }  
 }  
}

Внутри метода main есть строковая переменная text, которая содержит текст для поиска чисел.

Далее создается объект Pattern с использованием регулярного выражения "\d+([.]\d+)?". Это регулярное выражение ищет числа в тексте, которые могут быть целыми или с десятичной частью.

Затем создается объект Matcher, который будет использоваться для поиска совпадений с заданным регулярным выражением в тексте.

Затем с помощью цикла while происходит поиск всех совпадений с помощью метода find() объекта matcher. Когда совпадение найдено, выводится найденное число с помощью метода group().

Таким образом, код найдет и выведет все числа в строке text, такие как "20.45", "19.99", "3", "5", "5.4".

import java.util.regex.\*;  
  
//Задание: 2  
  
public class checkPassword {  
 public static void main(String[] args) {  
 String text = "sdkHGH16mm204";  
 try {  
 *validatePassword*(text);  
 System.*out*.println("Пароль корректен!");  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Ошибка: " + e.getMessage());  
 } catch (IllegalStateException e) {  
 System.*out*.println("Проверь последовательность методов: " + e.getMessage());  
 }  
 }  
  
 public static void validatePassword(String password) {  
 try {  
 String regex = "^(?=.\*[A-Z])(?=.\*\\d)[A-Za-z\\d]{8,16}$";  
  
 Pattern pattern = Pattern.*compile*(regex);  
 Matcher matcher = pattern.matcher(password);  
  
 if (!matcher.matches()) {  
 throw new IllegalArgumentException("Пароль не соответствует требованиям.");  
 }  
 } catch (PatternSyntaxException e) {  
 System.*out*.println("Некорректный синтаксис регулярного выражения: " + e.getMessage());  
 }  
 }  
}

Внутри метода main есть строковая переменная text, которая содержит пароль для проверки.

Затем вызывается метод validatePassword с передачей строки text в качестве аргумента. Затем используется конструкция try-catch, чтобы обработать возможное исключение.

Метод validatePassword проверяет, соответствует ли переданный пароль заданным требованиям. Требования определены с использованием регулярного выражения ^(?=.\*[A-Z])(?=.\*\d)[A-Za-z\d]{8,16}$. Это регулярное выражение проверяет, что пароль содержит хотя бы одну заглавную букву ((?=.\*[A-Z])), хотя бы одну цифру ((?=.\*\d)), и состоит из букв (как заглавных, так и строчных) и цифр ([A-Za-z\d]) в количестве от 8 до 16 символов ({8,16}).

Если пароль не соответствует требованиям, то метод validatePassword генерирует исключение типа IllegalArgumentException и выбрасывает сообщение "Пароль не соответствует требованиям."

В блоке catch в методе main обрабатывается исключение, если оно возникло. В случае исключения будет выведено сообщение "Ошибка: " и сообщение исключения (e.getMessage()).

Если исключение не возникло (т.е. пароль прошел проверку), будет выведено сообщение "Пароль корректен!".

import java.util.regex.\*;  
  
//Задание: 3  
  
public class LinkReplacer {  
 public static void main(String[] args) {  
 String text = "Ссылка 1: http://chirrup.ru, " +  
 "Ссылка 2: https://www.blablabla.com";  
  
 try {  
 String regex = "(http|https)://[\\w.-]+(/[\\w-./?%&=]\*)?";  
  
 Pattern pattern = Pattern.*compile*(regex);  
 Matcher matcher = pattern.matcher(text);  
 String result = matcher.replaceAll("<a href=$0>$0</a>");  
 System.*out*.println(result);  
 } catch (PatternSyntaxException e) {  
 System.*out*.println("Некорректный синтаксис регулярного выражения: " + e.getMessage());  
 } catch (IllegalStateException e) {  
 System.*out*.println("Проверь последовательность методов: " + e.getMessage());  
 }  
 }  
}

В методе main есть переменная text, которая содержит текст, в котором нужно найти ссылки для замены. В данном случае, текст содержит две ссылки: "[http://chirrup.ru](http://chirrup.ru/)" и "[https://www.blablabla.com](https://www.blablabla.com/)".

Далее определена строковая переменная regex, в которой задано регулярное выражение для поиска ссылок. Регулярное выражение "(http|https)://[\w.-]+(/[\w-./?%&=]\*)?" ищет подстроку, начинающуюся с "http://" или "https://", за которыми может следовать набор букв, цифр, точек, дефисов и подчеркиваний ([\w.-]+). Затем может следовать / и набор символов, включая буквы, цифры, дефисы, точки, слеши и специальные символы (/[\w-./?%&=]\*). Все это является необязательной частью (? в конце). Таким образом, регулярное выражение ищет все ссылки в тексте.

Затем код в блоке try компилирует регулярное выражение в объект Pattern и создает объект Matcher, связанный с текстом text.

Далее вызывается метод replaceAll на объекте matcher, в котором каждая найденная ссылка заменяется на <a href=$0>$0</a>. $0 обозначает найденную ссылку. То есть, каждая найденная ссылка будет заменена на <a href=ссылка>ссылка</a>, где ссылка - это сама ссылка.

Результат замены сохраняется в строковую переменную result.

В случае возникновения исключения типа RuntimeException (например, ошибки в регулярном выражении), будет выполнен блок catch, где будет выведено сообщение "Ошибка при использовании регулярного выражения:" и сообщение об ошибке (e.getMessage()).

Если исключение не возникло, то будет выведен результат замены ссылок в тексте.

import java.util.regex.\*;  
  
//Задание: 4  
  
public class checkIP {  
 public static void main(String[] args) {  
 String text1 = "0156.0044.353AAAAA5.7776";  
 String text2 = "5005.0011.0222.0144";  
  
 try {  
 *isValid*(text2);  
 System.*out*.println("IP корректен.");  
 } catch (IllegalStateException e) {  
 System.*out*.println("Проверь последовательность методов: " + e.getMessage());  
 } catch (IllegalArgumentException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public static void isValid(String text) {  
 try {  
 String regex = "([0-9]{4}.){3}[0-9]{4}";  
  
 Pattern pattern = Pattern.*compile*(regex);  
 Matcher matcher = pattern.matcher(text);  
  
 if (!matcher.matches()) {  
 throw new IllegalArgumentException("Неверный IP");  
 }  
 } catch (PatternSyntaxException e) {  
 System.*out*.println("Некорректный синтаксис регулярного выражения: " + e.getMessage());  
 }  
 }  
}

В методе main есть две строки: text1 и text2, которые содержат текст, предположительно представляющий IP-адреса.

Затем в блоке try вызывается метод isValid с аргументом text2.

Метод isValid принимает строковый аргумент text, который предположительно представляет IP-адрес.

В методе isValid определяется регулярное выражение regex для валидации IP-адресов. Регулярное выражение ([0-9]{0,4}\.){3}[0-9]{0,4} проверяет, что строка состоит из четырех сегментов, разделенных точками, каждый из которых содержит от 0 до 4 цифр ([0-9]{0,4}). То есть, оно проверяет, что каждый сегмент IP-адреса содержит от 0 до 4 цифр.

Затем код компилирует регулярное выражение в объект Pattern и создает объект Matcher, связанный с текстом text.

Далее проверяется, совпадает ли текст с регулярным выражением с помощью метода matches() на объекте matcher. Если не совпадает, значит, IP-адрес некорректен, и бросается исключение типа IllegalArgumentException с сообщением "IP адрес некорректный.".

Если совпадение было успешным, то в блоке try в методе main будет выведено сообщение "IP корректен.".

В случае возникновения исключения типа IllegalArgumentException, будет выполнен блок catch, где будет выведено сообщение "Ошибка при использовании регулярного выражения:" и сообщение об ошибке (e.getMessage()).

import java.util.regex.\*;  
  
//Задание: 5  
  
public class searchWords {  
 public static void main(String[] args) {  
 String text = "why you bullying me? u cool man, why u bullying me, every one asking. Likvid asking you. why u bulling, sympathetic person?";  
 char letter = 'w';  
  
 *searchWordsStartingWithLetter*(text, letter);  
 }  
  
 public static void searchWordsStartingWithLetter(String text, char letter) {  
 try {  
 String pattern = "\\b" + letter + "\\w+\\b";  
  
 Pattern regex = Pattern.*compile*(pattern);  
 Matcher matcher = regex.matcher(text);  
  
 while (matcher.find()) {  
 System.*out*.println(matcher.group());  
 }  
 } catch (PatternSyntaxException e) {  
 System.*out*.println("Некорректный синтаксис регулярного выражения: " + e.getMessage());  
 }  
 }  
}

В методе main определены две переменные: text, которая содержит строку текста, и letter, которая содержит символ w.

Затем вызывается метод searchWordsStartingWithLetter с аргументами text и letter.

Метод searchWordsStartingWithLetter принимает два аргумента: text (строка текста) и letter (символ). Он выполняет поиск слов, начинающихся на заданную букву в тексте.

В методе searchWordsStartingWithLetter определяется регулярное выражение pattern, которое состоит из следующих элементов:

\b - граница слова.

letter - заданная буква.

\w+ - один или несколько символов слова (буквы, цифры, подчеркивание).

\b - граница слова.

Затем регулярное выражение компилируется в объект Pattern и создается объект Matcher, связанный с текстом text.

Далее выполняется цикл while, в котором метод find() вызывается на объекте matcher. Этот метод выполняет поиск соответствия регулярному выражению в тексте. Если найдено соответствие, метод group() возвращает найденное совпадение, которое затем выводится на экран с помощью метода println().