Задание на практическую работу №25

1.Ознакомиться с классами Pattern, Matcher и PatternSyntaxException.
2. Выбрать IPv4 адреса во всех возможных, представлениях: десятичном, шестнадцатеричном и восьмеричном, с точками и без. Подробнее про IP адреса можно узнать в википедии.
3. Составить регулярное выражение, определяющее является ли заданная строка IP адресом, записанным в десятичном виде.
– пример правильных выражений: 127.0.0.1, 255.255.255.0.
– пример неправильных выражений: 1300.6.7.8, abc.def.gha.bcd.
4. Создать запрос для вывода только правильно написанных выражений со скобками (количество открытых и закрытых скобок должно быть одинаково).
– пример правильных выражений: (3*+*5)*–*9*×*4.
– пример неправильных выражений: ((3*+*5)*–*9*×*4.

Класс Pattern

Класс java.util.regex.Pattern применяется для определения регулярных выражений, для которого ищется соответствие в строке, файле или другом объекте представляющем собой некоторую последовательность символов. Этот класс используется для простой обработки строк. Для более сложной обработки строк используется класс Matcher, рассматриваемый ниже.

В классе Pattern объявлены следующие методы:

- compile(String regex) – возвращает Pattern, который соответствует regex;
- matcher(CharSequence input) – возвращает Matcher, с помощью которого можно находить соответствия в строке input;
- matches(String regex, CharSequence input) – проверяет на соответствие строки input шаблону regex;
- pattern() – возвращает строку, соответствующую шаблону;
- split(CharSequence input) – разбивает строку input, учитывая, что разделителем является шаблон;
- split(CharSequence input, int limit) – разбивает строку input на не более чем limit частей.

С помощью метода matches() класса Pattern можно проверять на соответствие шаблону целой строки, но если необходимо найти соответствия внутри строки, например, определять участки, которые соответствуют шаблону, то класс Pattern не может быть использован. Для таких операций необходимо использовать класс Matcher.

Класс Matcher

С помощью класса java.util.regex.Matcher можно получить больше информации каждом соответствии.

Начальное состояние объекта типа Matcher не определено. Попытка вызвать какой-либо метод класса для извлечения информации о найденном соответствии приведет к возникновению ошибки IllegalStateException. Для того чтобы начать работу с объектом Matcher нужно вызвать один из его методов:

- matches() – проверяет, соответствует ли вся строка шаблону;
- lookingAt() – пытается найти последовательность символов, начинающуюся с начала строки и соответствующую шаблону;
- find() или find(int start) – пытается найти последовательность символов, соответствующих шаблону, в любом месте строки. Параметр start указывает на начальную позицию поиска.

Иногда необходимо сбросить состояние объекта класса Matcher в исходное, для этого применяется метод reset() или reset(CharSequence input), который также устанавливает новую последовательность символов для поиска.

Для замены всех подпоследовательностей символов, удовлетворяющих шаблону, на заданную строку можно применить метод replaceAll(String replacement).

Для того чтобы ограничить поиск границами входной последовательности применяется метод region(int start, int end), а для получения значения этих границ – regionEnd() и regionStart(). С регионами связано несколько методов:

useAnchoringBounds(boolean b) – если установлен в true, то начало и конец региона соответствуют символам ^ и $ соответственно;

hasAnchoringBounds() – проверяет закрепленность границ.

В регулярном выражении для более удобной обработки входной последовательности применяются группы, которые помогают выделить части найденной подпоследовательности. В шаблоне они обозначаются скобками «(» и «)». Номера групп начинаются с единицы. Нулевая группа совпадает со всей найденной подпоследовательностью. Далее приведены методы для извлечения информации о группах:

end() – возвращает индекс последнего символа подпоследовательности, удовлетворяющей шаблону;

end(int group) – возвращает индекс последнего символа указанной группы;

group() – возвращает всю подпоследовательность, удовлетворяющую шаблону;

group(int group) – возвращает конкретную группу;

groupCount() – возвращает количество групп;

start() – возвращает индекс первого символа подпоследовательности, удовлетворяющей шаблону;

start(int group) – возвращает индекс первого символа указанной группы;

hitEnd() – возвращает истину, если был достигнут конец входной последовательности.

Класс PatternSyntaxException

PatternSyntaxException представляет непроверяемое исключение, которое отображает синтаксическую ошибку в шаблоне регулярного выражения. Класс PatternSyntaxException представлен следующими методами, которые помогут определить вам ошибку.

№.	Метод и описание
1	**public String getDescription()** Представляет описание ошибки.
2	**public int getIndex()** Представляет индекс ошибки.
3	**public String getPattern()** Представляет шаблон регулярного выражения, содержащего ошибку.
4	**public String getMessage()** Производит возврат многострочной строки, содержащей описание синтаксической ошибки и ее индекс, ошибочный образец регулярного выражения, а также визуальную индикацию индекса ошибки в шаблоне.