Pattern – класс, используемый для простой обработки строк.

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. 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;
* 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.

String regex = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\\d)[a-zA-Z\\d\_]{8,}$";

* ^ - начало строки
* $ - конец строки
* [\\d](file:///\\d) – любая цифра от 0-9
* [\\D](\\\\D) – любой нецифровой символ
* ?= - буквально если правда, то
* ?! – если ложь, то
* .\* - перед символами может быть любое кол-во символов
* [a-z] - в строке должна быть хотя бы 1 буква в ниж регистре
* [A-Z] - в строке должна быть хотя бы 1 буква в верх регистре
* [a-zA-Z\d\_] - в строке может содержаться любые лат буквы, цифры, и \_
* {8,} - кол-во символов не менее 8
* 0[1-9] - числа от 01-09
* [12]\\d - числа от 11-29
* 3[01] - числа 30 и 31
* [^...] - любой символ, не входящий в указанный набор
* | - или
* **\* -** Соответствует 0 или более повторениям предыдущего элемента.
* **+ -**Соответствует 1 или более повторениям предыдущего элемента
* **? -** Соответствует 0 или 1 повторению предыдущего элемента.
* **{n} -** Соответствует ровно `n` повторениям предыдущего элемента.
* **{n,}** - Соответствует `n` или более повторениям.
* **{n,m}** - Соответствует от `n` до `m` повторениям.

Matcher – класс используется для более сложной обработки строк. Начальное состояние типа объекта не определено. Чтобы начать работу с данным классом 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.
* reset() или reset(CharSequence input) - сбросить состояние объекта класса Matcher в исходное. Этот способ также устанавливает новую последовательность символов для поиска
* replaceAll(String replacement) – заменяет все последовательности символов, которые удовлетворяют шаблон на заданную строку
* region(int start, int end) – ограничивает поиск границами входной последовательности
* regionEnd() и regionStart() – получение значений границ входной последовательности

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.

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.

StringTokenizer- класс, который разбирает строку по лексемам (tokens).