Регулярні вирази в Java.

При роботі з даними часто доводиться виконувати операції пошуку та заміни за складними алгоритмами. Для виконання таких операцій у мові Java використовуються **регулярні вирази**.

Регулярний вираз у мові Java є рядком.

Регулярні висловлювання використовуються для вирішення наступних завдань:

- перевірка даних на наявність деякої послідовності даних, заданих за допомогою певного зразка, який називається шаблоном (**pattern**);
- заміна чи видалення даних;
- витяг деякої послідовності з даних.

Синтаксис регулярного виразу

- алфавітно-цифрові символи, включаючи букви кирилиці;
- символ '\\' – зворотна коса риса (зворотний слеш);
- символ '\0num' – вісімкове число, де num – одна, дві чи три вісімкові цифри;
- символ '\xhh' – код символу ASCII, де hh – дві шістнадцяткові цифри;
- символ '\uhhhh' – код символу Unicode, де hhhh – чотири шістнадцяткові цифри;
- символ табуляції ('\t' или '\u0009');
- символ нового рядка ('\n' або '\u000A');
- символ повернення каретки ('\r' або '\u000D');
- символ переходу до нової сторінки ('\f' або '\u000C');
- символ звукового сигналу ('\a' або '\u0007');
- символ **Escape** (**Esc**) ('\u001B');
- символ '\cx' – відповідає керуючому символу x (наприклад, \cM відповідає символу **Ctrl+M** або символу повернення каретки).

Операція альтернації

У регулярних виразах можна поєднувати кілька шаблонів, так щоб знайдений рядок відповідав хоча б одному з них. Для цього служить **операція альтернації**, яка в регулярних виразах задається символом**"|"**.

Приклад:

шаблон **"Ім'я | Прізвище"** означає пошук у вихідному рядку або рядка **"Ім'я"**, або рядка **"Прізвище"**

Одиночний метасимвол

Метасимвол крапка "." всередині регулярного виразу точка відповідає будь-якому одиночному символу, крім символу перекладу рядка.

Приклад :

шаблону **"w.r"** відповідають слова **war, world, forward** і т.д.

Квантіфікатори

Квантифікатори – це метасимволи, що використовуються для вказівки кількісних відносин між символами у шаблоні та в рядку, що шукається. Квантифікатор може бути поставлений після одиночного або після групи символів.

Метасимвол "+" означає, що символ, що йде перед ним, відповідає кільком таким символам, що йдуть поспіль, у рядку пошуку. Кількість символів може бути будь-якою, але має бути хоча б один символ.

Метасимвол "*" вказує, що символ, що йде перед ним, зустрічається нуль або більше разів.

Метасимвол "?" вказує, що попередній йому символ повинен зустрічатися або один раз, або взагалі не зустрічатися взагалі.

Приклад використання метасимволу "+"

"wor+"

Цьому шаблону будуть відповідати слова **world** і **worry**, а слово **woman** відповідати не буде.

Приклад використання метасимволу "*"

"wor*"

Цьому шаблону будуть відповідати слова **world**, **worry** і **woman** .

Приклад використання метасимволу "?"

"wor?"

Цьому шаблону будуть відповідати слова **world** і **woman**, а слово **worry** відповідати не буде.

Якщо необхідно **вказати точно кількість повторень символу**, можна скористатися конструкцією

{n,m}

n – мінімально допустима кількість повторень попереднього символу, **m** – максимально допустима кількість повторень. Один із параметрів **n** або **m** можна опустити.

Фактично квантифікатори "+", "*" та "?" є окремими випадками конструкції **{n,m}**: відповідно, **{1,}**, **{0,}** і **{0,1}**.

Пример:

"10{3,5}1" - 0 зустрічається щонайменше 3 рази, але не більше 5 разів.

"10{3,}1" - 0 зустрічається 3 або більше разів.

"10{0,3}1" - 0 зустрічається не більше 3 разів, але може взагалі не зустрітися.

"10{3}1" - 0 зустрічається рівно 3 рази.

В регулярних виразах часто використовують **поєднання метасимволів".*"** . Йому відповідають будь-які символи. За правилами обробки регулярних виразів знаходиться найдовший рядок, що все ще задовольняє шаблон пошуку.

Якщо необхідно обмежити пошук, слід після квантифікатора (у тому числі символу "**?**") вказати символ "**?**".

Приклад:

Початковий рядок:

"перший може стати як останній і останній може стати як перший."

Шаблон: **"перший.*останній"**

Результат: **"Перший може стати як останній і останній"**

Шаблон: **"перший.*?останній"**

Результат: **"Перший може стати як останній"**

Класи символів

Для пошуку в регулярних виразах можна задавати **класи символів**, укладені у квадратні дужки. Під час пошуку всі символи класу розглядаються як один символ. Всередині класу можна встановити діапазон символів, поміщаючи дефіс між межами діапазону.

Якщо першим символом класу є знак вставки "^", то значення виразу інвертується, такому класу відповідає ***будь-який символ, що не входить до класу***.

Оскільки у класах символи "]", "^" і "-" мають спеціальне значення, їхнього використання у класі існують певні правила:

- літерал "^" не повинен бути першим символом класу;
- перед літералом "]" повинен стояти символ зворотної косої межі;
- для приміщення у клас символу "-" досить або поставити його на першу позицію, або помістити перед ним символ зворотної риси.

Примеры задания классов символов

"[абвг]" або **"[а-г]"**

Рядок "**вода**" задовольняє шаблону, оскільки в ньому є символ "**г**", а рядок "**земля**" – не задовольняє, оскільки в ньому немає жодного із символів шаблону.

"Глава [0-9]+"

Рядки "**Глава 5**" і "**Глава 18**" задовольняє шаблону, оскільки в них після рядка "**Глава**" і пробілу йдуть цифри, а рядок "**Глава десять**" – не задовольняє, тому що в ньому після слова "**Глава**" та пробілу немає цифр.

"[А-Я][а-я]+"

Рядок "**Іванів**" задовольняє шаблону, оскільки він починається з великої літери, за якою слідують малі літери, а рядок "**іванівський**" – не задовольняє, оскільки він починається з малої літери.

"[.?!]"

Рядки "**Як справи?**", "**Чудово!**" і "**добре.**" відповідає шаблону, оскільки вони містять символи закінчення пропозиції. Символи "**.**" та "**?**" тут використовуються як звичайні символи, а не як метасимволи.

Спеціальні символи

Найбільш поширені класи символів можна задати за допомогою таких спеціальних символів:

\d – відповідає будь-якому цифровому символу (еквівалентно **[0-9]**);

\D – відповідає будь-якому нецифровому символу (еквівалентно **[^0-9]**);

\w – відповідає будь-якій латинській літері або цифрі (еквівалентно **[A-Za-z0-9]**);

\W – відповідає будь-якому небуквенному (латинському) та нецифровому символу (еквівалентно **[^A-Za-z0-9]**);

\s – відповідає будь-якому пробіловому символу (еквівалентно **[\f\n\r\t\v]**);

\S – відповідає будь-якому непробільному символу (еквівалентно **[^\f\n\r\t\v]**).

Спеціальні символи **\w** і **\W** не можна використовувати для літер алфавітів, відмінних від латинських літер. У цьому випадку необхідно задавати діапазон символів, як це робиться для класів символів.

Приклад використання класів символів

Шаблон для номера мобільного телефону має такий вигляд:

```
"\\d{3}-\\d{3}-\\d\\d-\\d\\d"
```

Цьому шаблону відповідає номер телефону

067-745-12-18

та не відповідає номер

055-867-1567

тому що в ньому немає тире перед передостанньою цифрою номера.

Анкери

За допомогою анкерів можна вказати, де рядки має бути знайдена відповідність до шаблону:

^ – відповідає позиції на початку рядка;
$ – відповідає позиції наприкінці рядка;
\b – відповідає межі між словом та пробельним символом;
\B – відповідає не межі слова.
Анкери **\b** та **\B** діють лише для рядків, що складаються з латинських букв.

Групування елементів

Операція групування елементів у круглі дужки дозволяє розглядати цю групу елементів як один елемент.
Якщо в регулярних виразах використовуються дужки, частини рядка, що відповідають фрагментам у дужках, запам'ятовуються в спеціальних змінних **$1** (перший фрагмент у дужках), **$2** (другий фрагмент у дужках), **$3** і т.д. Така операція називається захопленням (***capture***) змінною.

Приклад використання анкерів

Шаблон **"^Глава \\d{1,2}\..*"**

Шукає у вихідному рядку наступні відповідності: рядок "Глава" на початку рядка, потім пробіл, потім одна або дві цифри, потім точка, потім будь-який вміст до кінця рядка.

Приклад використання групування елементів

Необхідно знайти у рядку одне зі слів:

білий, червоний, зелений, жовтий чи **чорний**

Шаблон: **"білий|червоний|зелений|жовтий|чорний"**

або: **"(біл|червон|зелен|жовт|чорн)ий"**

"(біл|жовт|(черво|зеле|чор)н)ий"

Клас Pattern

Об'єкт класу **Pattern** є відкомпільованим поданням шаблону регулярного виразу і створюється не за допомогою ключового слова **new**, а за допомогою статичних **compile()** методів класу **Pattern**.

Методи класу Pattern:

public static Pattern compile(String *шаблон)*

повертає об'єкт класу **Pattern** для заданого у параметрі ***шаблону***

public static Pattern compile(String *шаблон, int прапорці)*

повертає об'єкт класу **Pattern** для заданого у параметрі шаблону із заданими ***прапорцями***

Прапорці представлені Java як статичні поля типу **int** класу **Pattern (public static final int):**

CASE_INSENSITIVE – включає пошук відповідності без урахування верхнього або нижнього регістру;

UNICODE_CASE – якщо цей прапорець увімкнений разом з прапорцем **CASE_INSENSITIVE**, то верхній та нижній регістри букв у коді *Unicode* не враховуються при пошуку відповідності;

UNIX_LINES –тільки символ "**\n**" враховується як символ закінчення рядка, в якому виконується пошук відповідності;

MULTILINE– якщо всередині рядка, в якому виконується пошук відповідності, є символи "**\n**", то вважається, що рядок складається з кількох рядків;

LITERAL – всі символи шаблону, включаючи метасимволи, розглядаються як звичайні символи;

DOTALL – якщо у шаблоні є метасимвол "**.**", то йому відповідатиме будь-який символ, включаючи символ "**\n**";

COMMENTS – у рядку шаблону, допустимі пробіли та коментарі, що починаються із символу "#" до кінця рядка;

CANON_EQ – при пошуку відповідності враховуватиметься відповідність між кодом символу і символом, тобто. при увімкненому прапорці латинська літера "**a**" відповідатиме коду **Unicode** цієї літери "\u00E5" у шаблоні.

Якщо необхідно задати одночасно кілька прапорців, то вони повинні бути розділені знаком операції АБО – "|".

Приклад

```
Pattern pattern1 =  Pattern.compile("abc");
```

Завдання шаблону – рядку **"abc".**

```
Pattern pattern2 =  Pattern.compile("string",
Pattern.CASE_INSENSITIVE);
```

Завдання шаблону – рядку "**string**" із пошуком відповідності без урахування регістру.

Прапорці можна включати **безпосередньо у шаблоні**, використовуючи наступну синтаксичну форму:

(? рядок-символів)

символи у ***рядку-символів*** можуть мати одне з наступних значень

i – для прапорця **CASE_INSENSITIVE**;

d – для прапорця **UNIX_LINES**;

m – для прапорця **MULTILINE**;

s – для прапорця **DOTALL**;

u – для прапорця **UNICODE_CASE**;

x – для прапорця **COMMENTS**.

Приклад

"(?ium)комп'ютер"

Методи класу Pattern

public static boolean matches(String *шаблон*, **CharSequence** *рядок-пошуку*)

перевіряє відповідність ***шаблону рядку-пошуку*** та повертає значення **true**, якщо рядок-пошуку відповідає шаблону та **false** – в іншому випадку.

public String pattern()

повертає рядок шаблону для об'єкта класу **Pattern**;

public int flags()

повертає числове значення прапорця для об'єкта класу **Pattern**; (якщо задано кілька прапорців, повертає суму їх числових значень);

public static String quote(String *рядок*)

повертає рядковий шаблон для заданого **рядка**;

Методи класу Pattern

public String[] split(CharSequence *рядок-пошуку*)

створює з ***рядка-пошуку*** масив, поділений на елементи за шаблоном, заданим в об'єкті класу **Pattern**;

public String[] split(CharSequence *рядок-пошуку*, int *межа*)

створює з ***рядка-пошуку*** масив, поділений на елементи за шаблоном, заданим в об'єкті класу **Pattern**, і із заданою в параметрі ***межа*** кількістю елементів. Якщо значення параметра більше або дорівнює кількості елементів, або менше 0, виводяться всі елементи, якщо менше кількості елементів - всі відповідності, що залишилися, виводяться в останньому елементі масиву;

public String toString() повертає рядкове представлення відкомпільованого шаблону.

Клас Matcher

Клас Matcher забезпечує виконання пошуку чи заміни відповідності заданому об'єктом класу **Pattern** шаблону.

Об'єкт класу **Matcher** створюється за допомогою методу

```
public Matcher matcher(CharSequence рядок-пошуку)
```

класу **Pattern** для ***рядка-пошуку***.

Рядкове представлення об'єкта класу **Matcher** можна отримати за допомогою методу

```
public String toString()
```

Операції з регіонами

Пошук відповідності виконується у підстроці вихідного рядка, який називається **регіоном** (***region***). За умовчанням регіоном є вся послідовність символів, що вводиться.

Встановлення меж регіону

public Matcher region(int *початковий-індекс*, **int** *кінцевий-індекс*)

метод повертає об'єкт класу **Matcher** для підрядку, що починається з ***початкового-індексу*** і закінчується індексом, на одиницю меншою, ніж ***кінцевий-індекс***.

Поточні значення початкового та кінцевого індексів регіону

public int regionStart()
public int regionEnd().

Характер меж регіону

public Matcher useAnchoringBounds(boolean *прапорець*)
public Matcher useTransparentBounds(boolean *прапорець*)

Методи пошуку відповідностей

public boolean matches()

виконує для об'єкта класу **Matcher** пошук відповідності всього регіону, починаючи з початку регіону. Повертає **true**, якщо відповідність знайдена і **false** – інакше.

public boolean lookingAt()

виконує для об'єкта класу **Matcher** пошук, починаючи з початку регіону на наявність шаблону в регіоні, але необов'язково відповідності всього регіону шаблону. Повертає **true**, якщо відповідність знайдена і **false** – інакше.

public boolean find()

виконує для об'єкта класу **Matcher** пошук, починаючи з початку регіону або якщо попередній виклик методу був успішним, і об'єкт класу **Matcher** не був скинутий, з першого символу після знайденої попередньої відповідності. Повертає **true**, якщо відповідність знайдена і **false** – інакше.

public boolean find(int *початковий-індекс*)

виконується так само, як і попередній метод, але пошук починається не з початку регіону, а заданого ***початкового-індексу***. Якщо необхідно знайти всі відповідності шаблону в рядку, починаючи з ***початкового-індексу***, цей метод можна використовувати тільки для пошуку першої відповідності. Решта відповідності визначаються за допомогою методу **find()** без параметрів.

Методи заміни

Методи класу **Matcher** дозволяють не тільки виконати пошук у рядку за заданим шаблоном, а й замінити знайдені відповідності заданими послідовностями символів – рядками заміни.

public String replaceFirst(String *рядок-заміни*)

public String replaceAll(String *рядок-заміни*)

дозволяють замінити лише першу відповідність або всі відповідності у рядку пошуку ***рядком-заміни***. Обидва методи повертають змінений рядок.

public StringBuffer appendTail(StringBuffer *новий-рядок*)

пересилає символи рядка пошуку в ***новий-рядок***, починаючи з кінцевої позиції і до кінця рядка пошуку. Цей метод використовується разом із методом **appendReplacement()** для завершення процесу пошуку та заміни у рядку.

public Matcher appendReplacement(StringBuffer *новий-рядок*, **String** *рядок-заміни*)

формує ***новий-рядок*** за наступним алгоритмом:

- пересилає символи рядка пошуку в ***новий-рядок***, починаючи з кінцевої позиції (append position) до символу на одиницю меншого, ніж символ, що визначається методом**start()** об'єкта **Match**;
- до нового рядка додається ***рядок-заміни***;
- кінцева позиція в новому рядку стає рівною позицією, що визначається методом **end()** об'єкта **Match**.

На початку перегляду та заміни значення кінцевої позиції дорівнює 0.

Методи **appendReplacement()** і **appendTail()** виконують ті ж дії, що й методи **replaceFirst()** і **replaceAll()**, однак вони дозволяють керувати як кількістю замін, так і самими замінами в рядку.

Клас PatternSyntaxException

Клас **PatternSyntaxException** кидає виняток, якщо регулярний вираз (шаблон) містить синтаксичну помилку.

У класі визначено такі методи:

public String getPattern() повертає шаблон, що містить помилку

public String getDescription() повертає опис помилки;

public int getIndex() повертає позицію символу помилки у шаблоні;

public String getMessage()

повертає повідомлення про помилку, що містить усі перелічені вище компоненти: опис помилки та її індекс, шаблон, що містить помилку та візуальну індикацію індексу помилки всередині шаблону.

Методи класу String
для роботи з регулярними виразами

public boolean matches(String *шаблон*)

якщо об'єкт класу **String** відповідає ***шаблону***, повертає значення **true**, інакше повертає **false** (діє аналогічно методу **matches()** класу **Pattern**);

public String[] split(String *шаблон*)

створює для об'єкта класу **String** масив рядків, поділений на елементи по заданому ***шаблону*** (діє аналогічно до відповідного методу **split()** класу **Pattern**);

public String[] split(String *шаблон*, int *межа*)

- створює для об'єкта класу **String** масив рядків, розділений на елементи за заданим шаблоном, і з заданою в параметрі ***межа*** кількістю елементів (якщо значення параметра більше або дорівнює кількості елементів, або менше 0, виводяться всі елементи, якщо менше кількості елементів - всі відповідності виводяться в останньому елементі масиву) (діє аналогічно відповідному методу**split()** класу **Pattern**);

public String replaceFirst(String *шаблон*, String *рядок-заміни*)

замінює в об'єкті **String** першу відповідність ***шаблону*** на ***рядок-заміни*** та повертає змінений рядок (діє аналогічно методу **replaceFirst()** класу **Matcher**);

public String replaceAll(String *шаблон*, String *рядок-заміни*)

замінює в об'єкті **String** всі відповідності ***шаблону*** на ***рядок-заміни*** та повертає змінений рядок (діє аналогічно методу**replaceAll()** класу **Matcher**).

Лектор:

Старший викладач кафедри Електроніки и комп'ютерної техніки Сумського державного університету

Горячев О. Є.

В лекції використано матеріали авторів:

Шонін В.А.
Монахов В.В.