Wprowadzenie

Dziekanat potrzebuje twojej pomocy w dopasowaniu ważnych dokumentów zawierające dane uniwersytetu. Żeby im pomóc musisz zaimplementować Wyrażenia Regularne (**Regex**).

Podstawową ideą dopasowania regexa z łańcuchem wejściowym (string) jest przechodzenie przez abstrakcyjne stany reprezentowane przez zbiór strumieni wejściowych. Stan początkowy zawsze zawiera jeden strumień wejściowy, a mianowicie ciąg wejściowy (string), który chcemy dopasować. Regex przechodzi do następnych stanów poprzez manipulowanie bieżącym zbiorem strumieni wejściowych. Regex dopasowuje (akceptuje) łańcuch wejściowy, gdy stan końcowy zawiera pusty strumień wejściowy. W przeciwnym razie, nie akceptuje.

Na przykład, niech prosty regex literalny *r*=”a” spróbuje dopasować łańcuch wejściowy *s*=”ab”. Regex *r* początkowo odczytuje strumień wejściowy *s*=”ab” i z powodzeniem pasuje do pierwszej litery łańcucha *s.* Usunie literę “a” ze strumienia wejściowego *s* i zwróci nowy strumień wejściowy s_1=”b”. Ponieważ stan końcowy zawiera niepusty strumień, regex nie akceptuje łańcucha wejściowego.

Wyrażenie regularne podawane jest do systemu w łańcuchu znaków (string) z następującymi zasadami:

- Każde wyrażenie podstawowe ma odstęp 1 spacji: (“ “)
- Wyrażenie alternatywne (**Alternation**) oznaczone jest symbolem “|”
- Wyrażenie sekwencyjne (**Sequence**) oznaczone jest symbolem “&”
- Wyrażenie powtarzania (**Repetition**) oznaczone jest symbolem “*”
- Wyrażenie literalne to jakikolwiek ciąg znaków nie kolidujący z poprzednimi definicjami.

Dodatkowo, wyrażenie zapisane jest w notacji prefiksowej (prefix). Notacja prefiksowa wymaga aby wszystkie operatory (np. **&**), występowały przed swoimi operandami. Przykładowy format wejściowy: **"& MiNI PW"** oznacza wyrażenie (**MiNI&PW**). Wyrażenie w notacji prefiksowej, przetwarzane jest w klasie ***PrefixRegexParser***, która zwraca wyrażenie gotowe do dopasowania. Więcej przykładów znajduję sie w klasie ***ExampleRegexExpressions***.

Twoje zadanie

Twoim zadaniem jest zaimplementowanie czterech typów wyrażeń regularnych (Użyj interfejsu: ***IRegexExpression***) Klasa ***RegexContext***, reprezentuje aktualny stan, i zawiera zbiór strumieni wejściowych.

1) **Literal Expression**, który akceptuje literalne wyrażenia reprezentowane przez wbudowany typ *string*
    a) Przykład: r=“a”, akceptuje s_1=”a” , ale nie akceptuje s_2=”ab”
2) **Alternation Expression** (operator “**|**”), który przyjmuje dwa wyrażenia regularne i zwraca stan reprezentowany sumą dopuszczalnych stanów tych dwóch wyrażeń.
    a) Przykład: r=”(MiNI **|** EiTI)” akceptuje s_1=”MiNI” albo s_2=”EiTI”

3) **Sequence Expression** (operator “&“), który przyjmuje dwa wyrażenia regularne (lewy i prawy) i próbuje zaakceptować je w sekwencji od lewej do prawej.
    a) Przykład: r=”(MiNI & EiTI)” akceptuje s_1=”MiNI EiTI” ale nie akceptuje s_2=”MiNI” ani s_3=”EiTI”
4) **Repetition Expression** (operator “*****”), który przyjmuje jedno wyrażenie regularne i próbuje zaakceptować dowolną liczbę wystąpień tego wyrażenia.
    a) Przykład: r=”(a)*” akceptuje s_1=”a”, s_2=”aa” albo s_3=””
5) Przeciążenie metody ***ToString()*** każdego wyrażenia tak aby wskazywała jej operator. W przypadku **Literal Expression**, zwróć string literalny (patrz na przykład poniżej).

Przetwarzanie notacji prefiksowej:

1) Uzupełnienie klasy ***PrefixRegexParser*** tak aby zwracała wyrażenie regularne przetworzone z notacji prefiksowej. Gdy format wyrażenia na wejściu Parsera jest nie poprawny, powinien wystąpić wyjątek ***SyntaxErrorException***.
2) Poprawnie wykonane zadanie, powinno przechodzić wszystkie testy (patrz Main).

Przykładowy output

```
Regex: 'MiNI'
Input: MiNI
Matched!

Regex: 'MiNI'
Input: PW
Not Matched!

Regex: 'MiNI'
Input: MiNI PW
Not Matched!

Regex: ('MiNI'|'PW')
Input: MiNI
Matched!

Regex: ('MiNI'|'PW')
Input: PW
Matched!

Regex: ('MiNI'|'PW')
Input: MiNI PW
Not Matched!

Regex: 'MiNI'&'PW'
Input: MiNI
Not Matched!

Regex: 'MiNI'&'PW'
Input: PW
Not Matched!

Regex: 'MiNI'&'PW'
Input: MiNIPW
Matched!
```

```
Regex: ('MiNI')*
Input: MiNI
Matched!

Regex: ('MiNI')*
Input: MiNIMiNIMiNI
Matched!

Regex: ('MiNI')*
Input: MiNIMiNIMiNIMiNIMiNIMiNI
Matched!

Regex: ('MiNI')*
Input: MiNIMiNIMiNIPW
Not Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: id:1
Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: id:2
Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: id:3
Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: id:1232
Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: id:4
Not Matched!

Regex: 'id:'&((('1'|'2')|'3'))*
Input: i:2
Not Matched!

Corrently detected invalid format: |
Corrently detected invalid format: &
Corrently detected invalid format: *
Corrently detected invalid format: & MiNI
Corrently detected invalid format: | MiNI
Corrently detected invalid format: | MiNI MiNI MiNI
```