MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Inginerie Software și Automatică

CEBAN VASILE FAF-223

Report

Laboratory work n.4

of Formal Languages and Finite Automata

Checked by:
Dumitru Crețu, *university assistant*
DISA, FCIM, UTM

Chișinău - 2024

Topic: Regular expressions

Objectives:

1. Write and cover what regular expressions are, what they are used for;
2. Below you will find 3 complex regular expressions per each variant. Take a variant depending on your number in the list of students and do the following:
    - Write a code that will generate valid combinations of symbols conform given regular expressions (examples will be shown).
    - In case you have an example, where symbol may be written undefined number of times, take a limit of 5 times (to evade generation of extremely long combinations);
    - Bonus point: write a function that will show sequence of processing regular expression (like, what you do first, second and so on)
3. Write a good report covering all performed actions and faced difficulties.

Source Code:

```
StringBuilder resultBuilder = new StringBuilder();
Random random = new Random();
int startPosition, sequenceNumber = 1;
```

resultBuilder: This StringBuilder is used to construct the final random string that matches the regex pattern.
random: An instance of the Random class, used to generate random numbers.
startPosition: It will keep track of the starting position of each sequence in the regex pattern.
sequenceNumber: It keeps track of the sequence number being generated.

```
for (int i = 0; i < regexPattern.length(); i++) {
```

This loop iterates through each character of the input regex pattern.

```
char currentChar = regexPattern.charAt(i);
```

It retrieves the current character from the regex pattern.

```
if (Character.isLetter(currentChar) ||
Character.isDigit(currentChar))
```

This block checks if the current character is a letter or digit, indicating a character class. It then handles various quantifiers like ^, *, +, and ?.

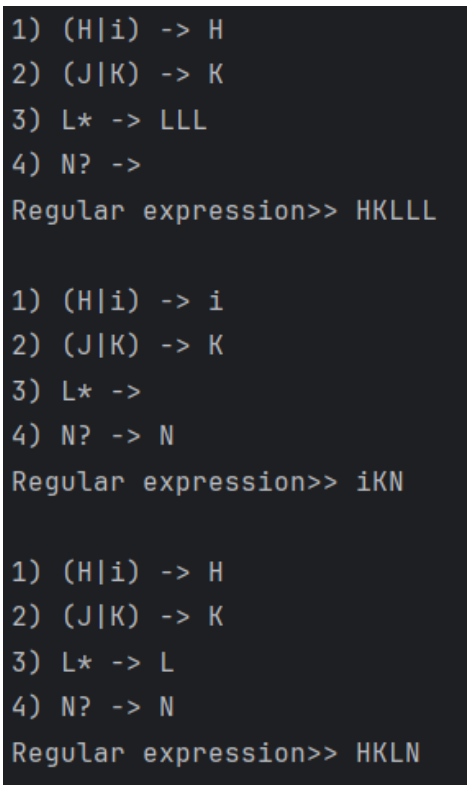
```
if (currentChar == '(') {}
```

This block handles groups by extracting the characters within parentheses and generating random strings for them.

Testing

```
First regular expression: {M?N^2(O|P)^3Q*R+}
1) M? ->
2) N^2 -> NN
3) (O|P)^3 -> PPO
4) Q* -> QQQQ
5) R+ -> RRR
Regular expression>> NNPPOQQQQRRR

Second regular expression: {(X|Y|Z)^38+(9|0)^2}
1) (X|Y|Z)^3 -> ZYY
2) 8+ -> 8888
3) (9|0)^2 -> 99
Regular expression>> ZYY888899

1) (X|Y|Z)^3 -> ZYX
2) 8+ -> 8888
3) (9|0)^2 -> 09
Regular expression>> ZYX888809

1) (X|Y|Z)^3 -> ZZY
2) 8+ -> 88
3) (9|0)^2 -> 00
Regular expression>> ZZY8800
```

```
1) (X|Y|Z)^3 -> ZZX
2) 8+ -> 88
3) (9|0)^2 -> 90
Regular expression>> ZZX8890

1) (X|Y|Z)^3 -> YZY
2) 8+ -> 88
3) (9|0)^2 -> 09
Regular expression>> YZY8809

Third regular expression: {(H|i)(J|K)L*N?}
1) (H|i) -> H
2) (J|K) -> J
3) L* -> LLL
4) N? -> N
Regular expression>> HJLLLN

1) (H|i) -> H
2) (J|K) -> J
3) L* -> L
4) N? ->
Regular expression>> HJL
```



Conclusion

In conclusion, the provided Java code serves as a laboratory exercise for generating random sequences based on predefined regular expression patterns. The RegexGenerator class contains methods to display results for specific regular expressions and to generate sequences based on a given regular expression pattern.

Throughout the lab, the code demonstrates several key programming concepts, including string manipulation, iteration, conditional statements, random number generation, and the use of sets and lists. It effectively breaks down the regular expression patterns and generates sequences according to the specified rules, providing a practical implementation of regex parsing and sequence generation.

The lab not only reinforces understanding of regular expressions but also enhances proficiency in Java programming by implementing algorithms to interpret and manipulate these expressions. Moreover, it offers opportunities for experimentation and exploration with different regular expression patterns and sequence generation strategies.

Overall, this laboratory exercise serves as a valuable hands-on learning experience, combining theoretical knowledge of regular expressions with practical coding skills in Java. It equips students with essential problem-solving abilities and a deeper understanding of both regular expressions and Java programming concepts.