Ministerul Educaţiei și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Laboratory work 4:

Regular expressions

Elaborated:

st. gr. FAF-223 Cravcenco Dmitrii

Verified:

asist. univ. Dumitru Cretu

Chişinău - 2024

Objectives:

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)

Implementation description

I go through all the characters of the regular expression and for each character I check if it is a letter or a digit. If it is a letter or a digit, I add it to the result. If it is a special character, I check which one it is and generate the corresponding string. For example, if it is a star, I add the previous character 5 times to the result. If it is a plus, I add the previous character 5 times to the result and then add the previous character. If it is a dot, I add the previous character and the next character to the result. If it is a pipe, I add the previous character to the result and then add the next character. I also implemented a function that shows the sequence of processing regular expression.

```
if (regex.length() > i + 1 && regex.charAt(i + 1) == '^') {
    int power = Integer.parseInt(regex.charAt(i + 2) + "");


    currentBuilder.append(String.valueOf(ch).repeat(Math.max(0, power)));
    i += 2;
}
```

In the code above I check if the next character is a power sign and if it is, I add the previous character to the result. I also increment the index by 2, because the power sign is followed by a number.

```
else if (regex.length() > i + 1 && regex.charAt(i + 1) == '?') {


    if (r.nextBoolean()) {
        currentBuilder.append(ch);
    }
    i++;
}
```

In the code above I check if the next character is a question mark and if it is, I add the previous character to the result with a 50% chance. I also increment the index by 1, because the question mark is followed by a question mark.

```
if (ch == '(') {
char nextCh = regex.charAt(i + 1);
while (nextCh != ')') {
    if (nextCh != '|') {
        chars.add(nextCh);
    }
    nextCh = regex.charAt(i + 1);
    i++;
}
```

In the code above I check if the current character is an opening bracket and if it is, I add the characters between the brackets to the result. I also increment the index until I reach the closing bracket.

```
System.out.println(regex.substring(c, i + 1) + " -> " + currentBuilder.toString());
result.append(currentBuilder);
```

In the end of the function I print the current regular expression and the result of the current regular expression. In order to keep track of how the regular expression is processed.

Conclusions and Results

In this laboratory, I implemented a function that generates valid combinations of symbols conform given regular expressions. I also implemented a function that shows the sequence of processing regular expression. I tested the functions with different regular expressions and they worked as expected. The functions generate valid combinations of symbols and show the sequence of processing regular expression. In conclusion, I successfully implemented the functions and achieved the objectives of the laboratory.