Laboratory work nr.4
Regular Expressions

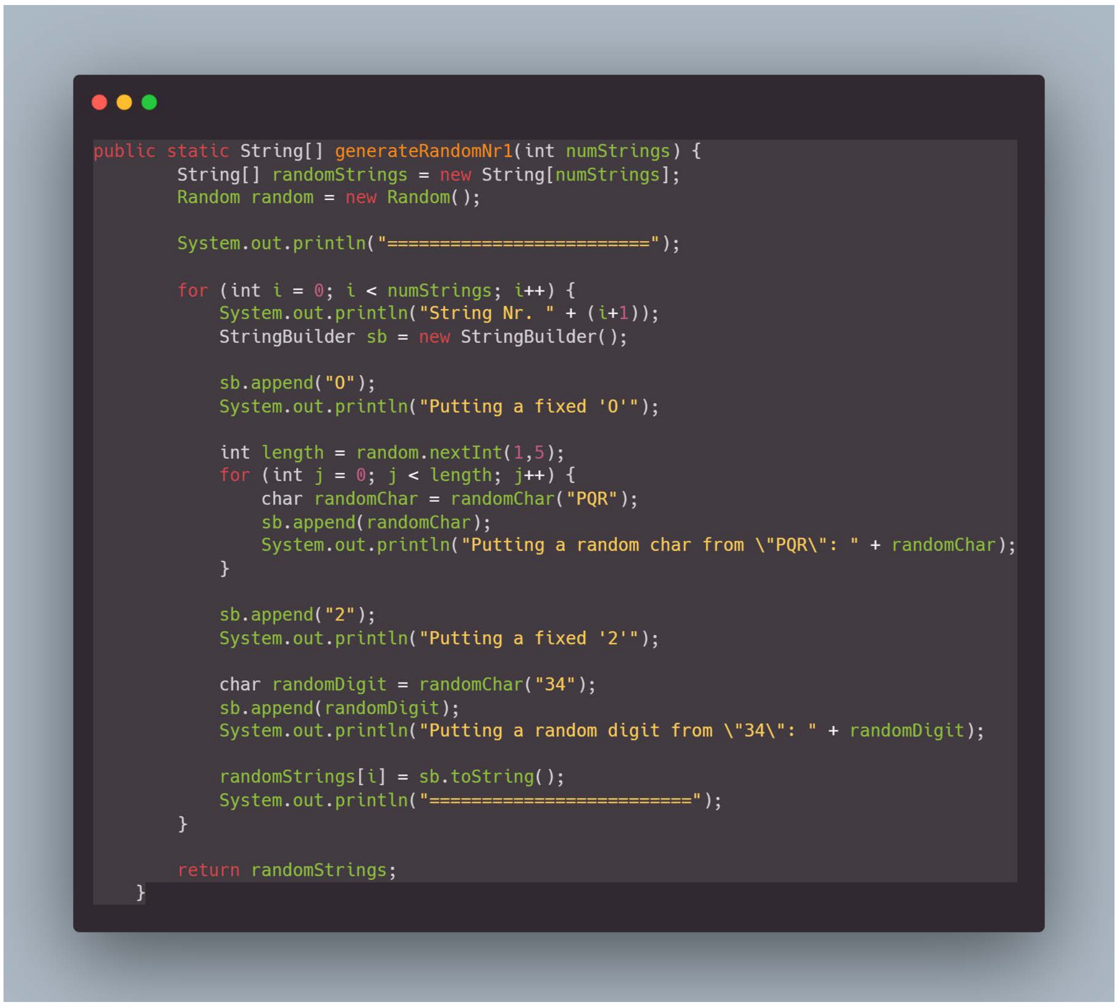
Course: Formal Languages & Finite Automata
Student: Schipschi Daniil

Objectives:
Write and cover what regular expressions are, what they are used for;

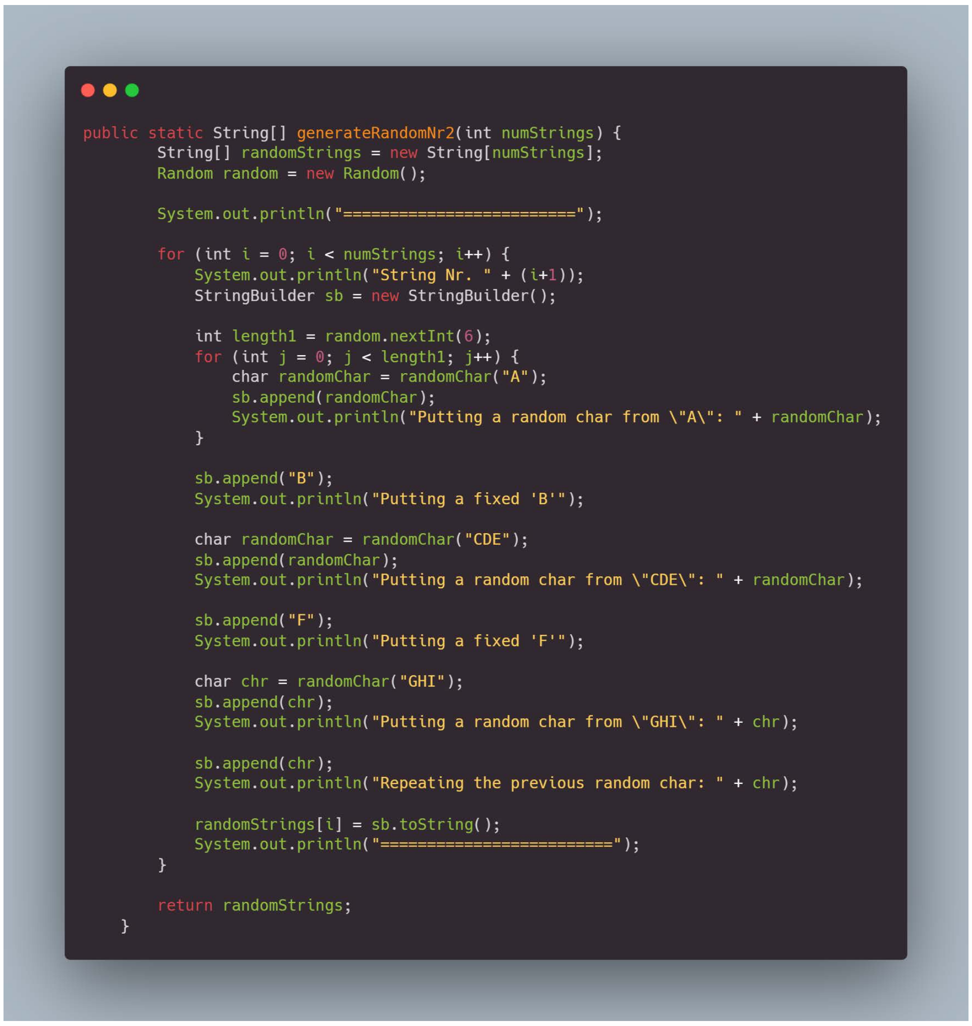
Below you will find 3 complex regular expressions per each variant. Take a variant depending on your number in the list of students and do the following:

a. Write a code that will generate valid combinations of symbols conform given regular expressions (examples will be shown).
b. 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);
c. Bonus point: write a function that will show sequence of processing regular expression (like, what you do first, second and so on)
Write a good report covering all performed actions and faced difficulties.

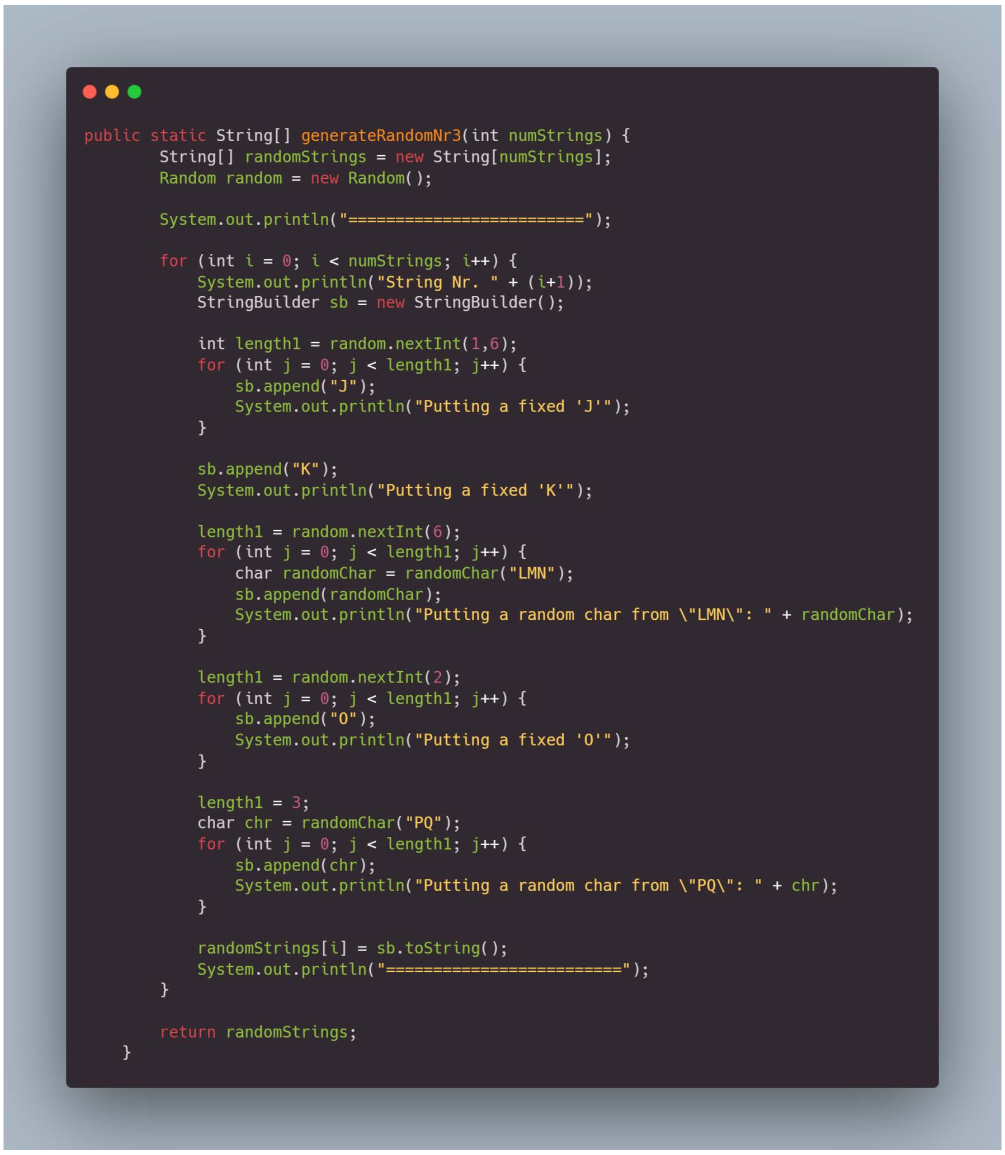
-> Regular expressions, often abbreviated as regex or regexp, are powerful tools for searching, matching, and manipulating text patterns in strings. They provide a concise and flexible syntax for specifying search patterns, allowing complex text processing tasks to be performed with ease. Regular expressions are widely used across programming languages, text editors, and command-line tools for tasks such as:
1 Pattern Matching:
Regular expressions can be used to find patterns within strings, such as specific words, characters, or sequences. For example, searching for all email addresses in a document or extracting phone numbers from a web page.
2 Validation:
Regular expressions are commonly used to validate input strings against predefined patterns. This can include validating email addresses, phone numbers, dates, or any other structured data format.
3 Text Manipulation:
Regular expressions enable the manipulation of text by searching and replacing specific patterns with desired substitutions. This is useful for tasks like data cleaning, formatting, or transformation.
4 Tokenization:
Regular expressions can be employed to tokenize strings by splitting them into smaller, meaningful components based on specified delimiters or patterns. This is essential for parsing and analyzing structured text data.
5 Syntax Highlighting:
Text editors and IDEs utilize regular expressions for syntax highlighting, allowing different parts of code or text to be visually distinguished based on predefined rules.
6 Web Scraping:
Regular expressions are valuable for extracting specific information from HTML or XML documents when scraping web pages. They enable the identification and extraction of data embedded within markup.
7 Search and Replace:
Regular expressions are extensively used in search and replace operations across text editors, IDEs, and command-line tools. They provide a powerful mechanism for finding and replacing text based on patterns.



```java
public static String[] generateRandomNr1(int numStrings) {
        String[] randomStrings = new String[numStrings];
        Random random = new Random();

        System.out.println("=========================");

        for (int i = 0; i < numStrings; i++) {
            System.out.println("String Nr. " + (i+1));
            StringBuilder sb = new StringBuilder();

            sb.append("0");
            System.out.println("Putting a fixed '0'");

            int length = random.nextInt(1,5);
            for (int j = 0; j < length; j++) {
                char randomChar = randomChar("PQR");
                sb.append(randomChar);
                System.out.println("Putting a random char from \"PQR\": " + randomChar);
            }

            sb.append("2");
            System.out.println("Putting a fixed '2'");

            char randomDigit = randomChar("34");
            sb.append(randomDigit);
            System.out.println("Putting a random digit from \"34\": " + randomDigit);

            randomStrings[i] = sb.toString();
            System.out.println("=========================");
        }

        return randomStrings;
    }
```



```java
public static String[] generateRandomNr2(int numStrings) {
        String[] randomStrings = new String[numStrings];
        Random random = new Random();

        System.out.println("=========================");

        for (int i = 0; i < numStrings; i++) {
            System.out.println("String Nr. " + (i+1));
            StringBuilder sb = new StringBuilder();

            int length1 = random.nextInt(6);
            for (int j = 0; j < length1; j++) {
                char randomChar = randomChar("A");
                sb.append(randomChar);
                System.out.println("Putting a random char from \"A\": " + randomChar);
            }

            sb.append("B");
            System.out.println("Putting a fixed 'B'");

            char randomChar = randomChar("CDE");
            sb.append(randomChar);
            System.out.println("Putting a random char from \"CDE\": " + randomChar);

            sb.append("F");
            System.out.println("Putting a fixed 'F'");

            char chr = randomChar("GHI");
            sb.append(chr);
            System.out.println("Putting a random char from \"GHI\": " + chr);

            sb.append(chr);
            System.out.println("Repeating the previous random char: " + chr);

            randomStrings[i] = sb.toString();
            System.out.println("=========================");
        }

        return randomStrings;
    }
```



```java
public static String[] generateRandomNr3(int numStrings) {
        String[] randomStrings = new String[numStrings];
        Random random = new Random();

        System.out.println("=========================");

        for (int i = 0; i < numStrings; i++) {
            System.out.println("String Nr. " + (i+1));
            StringBuilder sb = new StringBuilder();

            int length1 = random.nextInt(1,6);
            for (int j = 0; j < length1; j++) {
                sb.append("J");
                System.out.println("Putting a fixed 'J'");
            }

            sb.append("K");
            System.out.println("Putting a fixed 'K'");

            length1 = random.nextInt(6);
            for (int j = 0; j < length1; j++) {
                char randomChar = randomChar("LMN");
                sb.append(randomChar);
                System.out.println("Putting a random char from \"LMN\": " + randomChar);
            }

            length1 = random.nextInt(2);
            for (int j = 0; j < length1; j++) {
                sb.append("O");
                System.out.println("Putting a fixed 'O'");
            }

            length1 = 3;
            char chr = randomChar("PQ");
            for (int j = 0; j < length1; j++) {
                sb.append(chr);
                System.out.println("Putting a random char from \"PQ\": " + chr);
            }

            randomStrings[i] = sb.toString();
            System.out.println("=========================");
        }

        return randomStrings;
    }
```

Each of the generateRandomNr1, generateRandomNr2, and generateRandomNr3 functions generates random strings following specific patterns. They all start by initializing an array to store the generated strings and a random number generator. Then, for each string to be generated, they iterate through a series of steps to construct the string according to the defined pattern. These steps typically involve appending a combination of fixed characters and randomly selected characters from predefined sets to a StringBuilder. Once the string is constructed, it is added to the array of generated strings. Throughout the process, there are print statements that provide insight into the construction steps for each string. Finally, the functions return the array containing all generated strings. Overall, these functions showcase different approaches to generating random strings with specific characteristics, providing flexibility for various string generation requirements.

CONCLUSION:
In this lab, we immersed ourselves in the realm of regex functions, exploring their utility and versatility in text processing tasks. Through dissecting the generateRandomNr1, generateRandomNr2, and generateRandomNr3 functions, we gained insights into the intricacies of generating random strings with distinct patterns. Each function showcased unique methodologies for constructing strings, blending fixed characters with randomly selected ones from predefined sets. Throughout the journey, the print statements provided a window into the step-by-step construction process of each string, enhancing our understanding of the underlying logic. By unraveling these functions, we deepened our appreciation for regex's role in facilitating diverse string generation requirements, empowering us with invaluable skills for tackling text manipulation challenges in various domains.