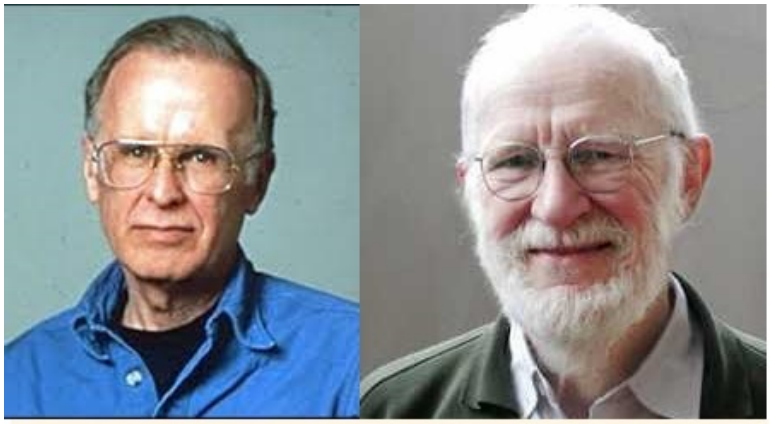
QUIZ 3

Issue Date : 26.03.2024 - Friday
Due Date : 28.03.2024 - Sunday (23:00)
Advisor : Berra Nur ÖZTÜRK, R.A. Görkem AKYILDIZ
Programming Language : Java 8u401 (1.8.0_401)



```
<expression>  ::=  <expression> + <term>
              |    <expression> - <term>
              |    <term>
<term>        ::=  <term> * <factor>
              |    <term> / <factor>
              |    <factor>
<factor>      ::=  number
              |    name
```

1 Introduction

In this quiz, you are expected to store Backus-Naur Form (BNF) using Java Collections (such as list, map, set etc.), which is for storing and manipulating group of objects, and then, you are expected to printout all possible strings that can be generated from given structured BNF Grammer through a **recursive** manner.

2 Backus-Naur Form (BNF)

In computer science, BackusNaur form (BNF or Backus normal form) is a notation used to describe the syntax of programming languages or other formal languages. It was developed by John Backus and Peter Naur. BNF can be described as a metasyntax notation for context-free grammars. BackusNaur form is applied wherever exact descriptions of languages are needed, such as in official language specifications, in manuals, and in textbooks on programming language theory. BNF can be used to describe document formats, instruction sets, and communication protocols.[1]

A BNF consists of:

- a set of terminal symbols
- a set of non-terminal symbols
- a set of production rules

Left hand side of a production rule represents a non-terminal symbols where right-hand side is a sequence of symbols. A BNF grammar example for a phone number like `(123)456-7890` (or like `123-4567`) is given below:

```
<phone-number> -> <area-code><7-dig>|<7-dig>
<area-code> -> "("<digit><digit><digit>")"
<7-dig> -> <digit><digit><digit>"-"<digit><digit><digit><digit>
<digit> -> "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
```

where all items between angle brackets are the non-terminal symbols and others stated within . are the terminals representing the end items. Note that all non-terminal symbols should be expanded until they are parsed into the terminal symbols, which you will do at the second part of this quiz. The vertical bar symbol (|) is used as in the meaning of or operator to tell that given non-terminal item can be expanded as either one among the given, for example `<phone-number>` can be either expanded as `<area-code><7-dig>` or `<7-dig>`.

3 Storing BNF in Java Collections

For this part, you are supposed to read the input file that contains Structured BNF Grammar, then you must store it using Java Collections, note that you must use appropriate collection class to store them and you must comment about your reasoning while making your selection at your code. For the sake of simplification, you do not have to worry about the recursive cases in BNF such as follows (where the symbol `item` recalls itself which results with infinite recursion): `<item> -> <item>"A"|"A"` Also, for the sake of simplification, in the BNF Grammar, instead of angle brackets, letter case is used to distinguish the terminal and non-terminal symbols which means each non-terminal will be represented with upper-case letters while terminals will be represented with lower-case letters. Sample input format for BNF Grammer can be as follows:

```
S->AB|aB
A->ab|BB|aB
B->ab|a|b
```

According to Grammer, S (which is short for start as S is the starting symbol) can be either expanded to AB or aB, A can be expanded to either ab or BB or aB, B can be expanded to either ab or a or b. Note that, S, A, and B are non-terminals while a and b are the terminals, which means A and a is not the same. So, S must be expanded until the output is consist of only a and/or b, which means the output must not contain any S, A, or B. Note that at each input, starting symbol will be the S.

4 Printing All Possible Strings

For this part, you are supposed to write each possible string in one-liner form to the output file. Say that the given Structured BNF Grammar is as follows:

```
S->A|Aa
A->a|aa
```

The output must be as follows: `((a|aa)|(a|aa)a)`

Step-by-step evaluation is as follows: S → (A|Aa) → ((a|aa)|Aa) → ((a|aa)|(a|aa)a)

5 Restrictions

- Your code must be able to execute on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- **You must benefit from Java Collections (java.util.Collections) and Recursion; any solution that does not contain even one of these concepts will not be accepted, moreover, partial point deductions may exist due to partially erroneous usage of these concepts.**
- Your code must be clean, do not forget that main method is just a driver method that means it is just for making your code fragments run, not for using them as a main container, create classes and methods in necessary situations but use them as required. Moreover, use four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, Polymorphism) if there is such a need, remember that your code must satisfy Object-Oriented Programming Principles.
- You are encouraged to use lambda expressions which are introduced with **Java 8**.
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that if someone wants to read your code they should be able to easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- Do not miss the submission deadline.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as clean code.
- Use UNDERSTANDABLE names for your variables, classes, and functions regardless of the length. The names of classes, attributes and methods must obey to the Java naming convention. This expectation will be graded as coding standards.

- You can ask your questions through courses Piazza group, and you are supposed to be aware of everything discussed in the Piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All quizzes must be original, individual work. Duplicate or very similar quizzes are both going to be considered as cheating.

6 Execution and Test

Your code must be executed under **Java 8u401 (1.8.0_401)** at **dev.cs.hacettepe.edu.tr**. If your code does not run at developer server during the testing stage, then you will be graded as 0 for code part even if it works on your own machine. Sample run command is as follows:

- Either `javac8 BNF.java` or `javac8 *.java` command for compilation.
- `java8 BNF input.txt output.txt` command for run.

7 Grading

Task	Point
Correct Output	100*
Total	100

*** Even though producing correct output seems like enough to get full credit, you must obey to the given rules in the PDF (for example using Java Collections and recursion, commenting etc.) otherwise you may face with some point deductions which may result with a grade that is as low as zero.**

8 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only not compressed .zip files because the system just supports .zip files).

```
- b<StudentID>.zip
    - <src>
       - BNF.java
       - *.java (Optional)
```

References

[1] Backus-naur form - wikipedia. https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form (Last access: 24.04.2024).

[2] Cs 331 spring 2013: Lecture notes for wednesday, january 30, 2013. https://www.cs.uaf.edu/2013/spring/cs331/lectures/lec-2013_01_30-bnf.html (Last access: 24.04.2024).