COSC455: "Program 0"

Overview

Make the changes described below using the sample parser (the one attached within Blackboard) as a guide.

Instructions:

1. **The first step is to ensure you can open the sample project in the IDE of your choice.**
   a. The sample project relies on the most recent LTS ("Long Term Support") version of ***Java/JDK***. The current LTS version is ***JDK 21***, so please make sure you have an up-to-date Java Developer's Kit installed. The builds from the *Adoptium* will automatically suggest the correct version for your computer.
   b. The IDE itself can be anything you like. Many students prefer ***VSCode***, but you must ensure the Extension Pack for Java is installed within VSCode.
      - Personally, I prefer either ***Netbeans*** or ***IntelliJ IDEA*** for developing with "JVM" languages like *Java*, *Kotlin*, and *Scala*. Both Netbeans and IntelliJ Community Edition are free.
      - I'm not a fan of the *Eclipse* IDE, but you are welcome to use it if you are already comfortable with it.

2. **Modify the sample `Parser.java` and `TokenSet.java` files to parse the additional grammar elements shown in the next section.** ("Grammar for Program 0" below)

   NOTE:

   The **`TokenSet`** class represents all "tokens" for the "Lexical Analysis" phase.

   The **`Parser`** class represents the "Recursive Decent Parser" logic.

   Do not modify `Main.java`; only the **`Parser.java`** and **`TokenSet.java`** classes require any modifications.

   - The parser should "accept" valid "inputs" and generate the Graphviz code, which illustrates the entire parse tree.
     - ***The existing sample already does the "Code Generation" for you, and you should only need to follow the structure of the*** `Parser.java` ***and*** `TokenSet.java` ***files to get the correct output.***

   - The parser should "reject" invalid "programs" with a descriptive error message; this message should be part of the output tree.
     - ***The sample already does this to some extent, but it is only in the form of "got*** x ***while expecting*** y***".***
     - *While not a strict requirement, you may wish to augment any syntax errors with additional information about the error.*

   The output can be used as input for ***Graphviz***, which can be installed locally, or you can use any of the many online versions:
   - Full installer
     - https://graphviz.org/download/
     - A VSCode Plugin.
   - Web Versions
     - https://edotor.net/
     - https://dreampuf.github.io/GraphvizOnline
     - And many others.

Grammar for Program 0

The Grammar should be extended to include the following production rules in **bold-red**.

```
<START>          ⟶   <SENTENCE> $$
<SENTENCE>       ⟶   <NOUN_PHRASE> <VERB_PHRASE> <NOUN_PHRASE> <PREP_PHRASE> <SENTENCE_TAIL>
<SENTENCE_TAIL>  ⟶   'and' <SENTENCE> | 'or' <SENTENCE> | <PUNCTUATION>

<NOUN_PHRASE>    ⟶   <ARTICLE> <ADJ_LIST> <NOUN>
<ADJ_LIST>       ⟶   <ADJECTIVE> <ADJ_TAIL> | ε
<ADJ_TAIL>       ⟶   <COMMA> <ADJECTIVE> <ADJ_TAIL> | ε

<VERB_PHRASE>    ⟶   <ADVERB> <VERB> | <VERB>
<PREP_PHRASE>    ⟶   <PREPOSITION> <NOUN_PHRASE> | ε

<ARTICLE>        ⟶   'a' | 'an' | 'the'
<NOUN>           ⟶   'dog' | 'cat' | 'rat' | 'fox' |  'tree' | 'house'
<VERB>           ⟶   'jumps' |  'chases' | 'climbs'
<ADJECTIVE>      ⟶   'fast' | 'slow' | 'lazy' | 'tall'
<ADVERB>         ⟶   'quickly' | 'quietly'
<COMMA>          ⟶   ','
<PUNCTUATION>    ⟶   '.' | '!'
<ADVERB>         ⟶   'quickly' | 'slowly'
<PREPOSITION>    ⟶   'around' | 'up' | 'over' | 'under'
```

The grammar above adds “Conjunctions” and “Prepositional Phrases” to allow sentences like:

```
The dog chases the cat and the cat climbs a tree.

The dog chases the cat up the tree.

The slow, lazy dog chases the fast cat around the house.
```

Note: The program should continue to operate as it does in the example, with the only difference being a slightly improved grammar, which will expand the language.