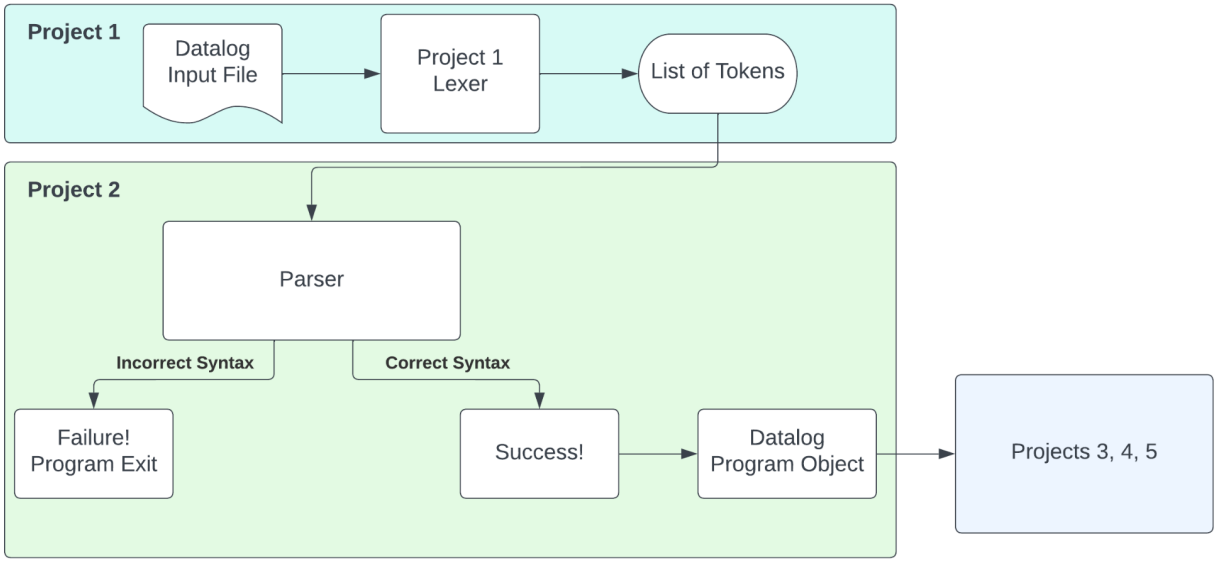
Overview

In this project, you will build a parser for datalog. In project 1, you built a lexer to convert datalog text into Tokens. The parser will use those Tokens along with a datalog grammar and recursive descent parsing to verify that the input has correct datalog syntax. Along with the parser, you will also build a datalog program data structure in preparation for projects 3 and 4. Here is the big picture:



Parser

This represents where the different pieces of the grammar go in the Parser class. See the Project 2 guide slides as well as the Jupyter notebook tutorials for more information on how to implement this.

```
datalogProgram  ->  SCHEMES COLON scheme schemeList FACTS COLON factList RULES COLON ruleList QUERIES COLON query queryList EOF

schemeList      ->  scheme schemeList | lambda
factList        ->  fact factList | lambda
ruleList        ->  rule ruleList | lambda
queryList       ->  query queryList | lambda

scheme          ->  ID LEFT_PAREN ID idList RIGHT_PAREN
fact            ->  ID LEFT_PAREN STRING stringList RIGHT_PAREN PERIOD
rule            ->  headPredicate COLON_DASH predicate predicateList PERIOD
query           ->  predicate Q_MARK

headPredicate   ->  ID LEFT_PAREN ID idList RIGHT_PAREN
predicate       ->  ID LEFT_PAREN parameter parameterList RIGHT_PAREN

predicateList   ->  COMMA predicate predicateList | lambda
parameterList   ->  COMMA parameter parameterList | lambda
stringList      ->  COMMA STRING stringList | lambda
idList          ->  COMMA ID idList | lambda
parameter       ->  STRING | ID
```

This production rule represents the code wtihin the function schemeList(). For example, this one will have no calls to __match() and two function calls (scheme() and schemeList()) as well as a check in self.FOLLOW['schemeList'] to take care of the lambda production

All UPPER_CASE words are terminals. These termianls will be passed in as a paramter in a call to __match()

All lowerCase words are nonterminals. On the right side, they represent function calls

Functions in Parser

Datalog Program Data Structure

The following diagram shows the structure of the DatalogProgram classes. The variables that are shown are the variables that you will need for each class (you might add more if needed). After your Parser class is completed, you will go back and add logic to create each of these classes as you parse. Notice how each piece of the input relates to each class, and consider how that will relate to nonterminals and terminals in the datalog grammar.