Lab 4

Darius - Daniel Calugar

Github repository: https://github.com/darius-calugar/flcd-lab-4

Language Specification

Alphabet:

- uppercase letters (A-Z)
- lowercase letters (a-z)
- decimal digits (0-9)
- quote character "
- minus character -

Identifiers:

- identifier ::= letter{letter|digit}
- letter ::= "a"|...|"z"|"A"|...|"Z"
- digit ::= "0"|...|"9"

Constants:

- constNum ::= -digit{digit}|digit{digit}
- constString ::= "\""{number|letter|" "|”-”}"\""

Reserved Words:

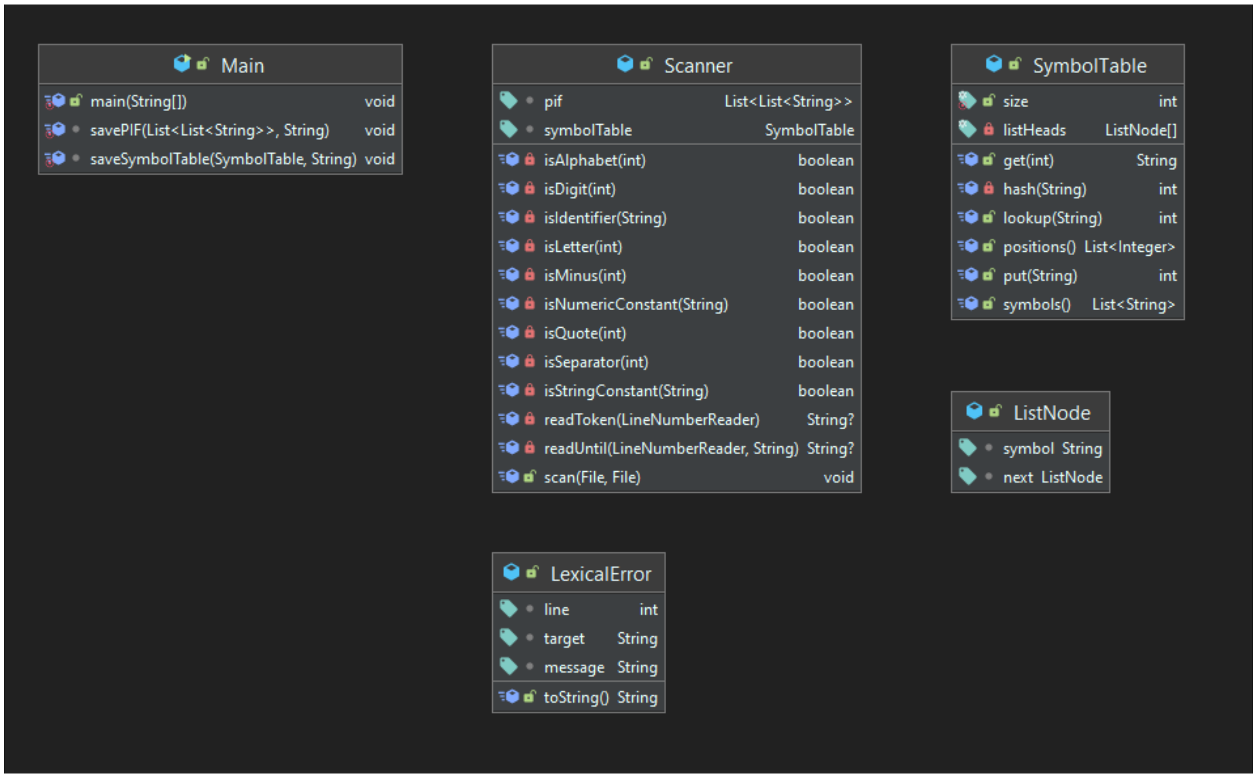
- begin, end, stop, integer, string, list, define, of, at, becomes, plus, minus, times, divided, mod, smaller, larger, is, read, print, if, then, else, each, from, to

Syntax:

- **program** ::= compStmt
- **type** ::= "integer"|"string"|userType|arrayType
- **arrayType** ::= type "list"
- **userType** ::= identifier
- **var** ::= identifier ["of" var] | var "at" constNum
- **expression** ::= numExpression|constString
- **numExpression** ::= expression operator expression|var|constNum
- **condition** ::= expression relation expression
- **operator** ::= "plus"|"minus"|"times"|"divided"|"mod"
- **relation** ::= "smaller"|"larger"|"is"

- **def** ::= "define" userType
- **declStmt** ::= type identifier
- **declListStmt** ::= arrayType identifier
- **stmt** ::= cmpdStmt|assingStmt|ioStmt|ifStmt|declStmt|declListStmt|stopStmt
- **cmpdStmt** ::= "begin" {stmt} "end"
- **assignStmt** ::= var "becomes" expression
- **ioStmt** ::= readStmt|printStmt
- **readStmt** ::= "read" var
- **printStmt** ::= "print" expression
- **ifStmt** ::= "if" condition "then" stmt ["else" stmt]
- **loopStmt** ::= "each" var "from" numExpression "to" numExpression stmt
- **stopStmt** ::= "stop"

Class Structure



```
Main
main(String[])                          void
savePIF(List<List<String>>, String)     void
saveSymbolTable(SymbolTable, String)    void
```

```
Scanner
pif                                     List<List<String>>
symbolTable                             SymbolTable
isAlphabet(int)                         boolean
isDigit(int)                            boolean
isIdentifier(String)                    boolean
isLetter(int)                           boolean
isMinus(int)                            boolean
isNumericConstant(String)               boolean
isQuote(int)                            boolean
isSeparator(int)                        boolean
isStringConstant(String)                boolean
readToken(LineNumberReader)             String?
readUntil(LineNumberReader, String)     String?
scan(File, File)                        void
```

```
SymbolTable
size                 int
listHeads            ListNode[]
get(int)             String
hash(String)         int
lookup(String)       int
positions()          List<Integer>
put(String)          int
symbols()            List<String>
```

```
ListNode
symbol   String
next     ListNode
```

```
LexicalError
line        int
target      String
message     String
toString()  String
```

Main class

- Initializes the **Scanner**, **SymbolTable**, and the **PIF** data structure, then it begins scanning the files “input/**$fileName**.verba”.
- After the files are scanned, the program prints the first **LexicalError** found.
- If no errors are found, the appropriate output files will be created inside “output/”.

static void **savePIF**(List<List<String>> pif**,** String fileName)

- Post:
  - Saves the PIF into a new file at “output/**$fileName**_pif.out”.
  - Throws **IOException** when an error is thrown by the file writer.

static void **saveSymbolTable**(SymbolTable symbolTable, String fileName)

- Post:
  - Saves the symbol table into a new file at “output/**$fileName**_st.out”.
  - Throws **IOException** when an error is thrown by the writer

Scanner class

public void **scan**(File inputFile, File reservedWordsFile)

- Analyses each token from the input file and checks if it is either a reserved word, an identifier, or constant.
- Populates the PIF and ST with the appropriate tokens.
- Pre:
    - Both files should exist and have read permissions
    - The reserved words file must only contain the reserved tokens separated by newline
- Post:
    - Throws **IOException** if there was an error reading the files
    - Throws **LexicalError** if the input file is lexically incorrect

private String **readToken**(LineNumberReader reader, String separators)

- Reads a token from the reader by calling **readUntil**.
- If the token is **-**, it is concatenated with the following **readUntil** result. If the resulting token is not a numeric constant, **LexicalError** is thrown
- If the token is **“**, it is concatenated with the following **readUntil** result (with **“** as the only separator). If the resulting token is not a string constant, **LexicalError** is thrown
- Post:
    - Returns the read token
    - Throws **IOException** if there was an error reading the files
    - Throws **LexicalError** if the input file is lexically incorrect

private String **readUntil**(LineNumberReader reader, String separators)

- Reads a character by character from the reader until a separator is reached.
- If the separator is not a global language separator **([\n\r])**, it is also included in the resulting string.
- Post:
    - Returns the read string
    - Throws **IOException** if there was an error reading the files
    - Throws **LexicalError** if the input file is lexically incorrect

private boolean **isIdentifier**(String token)

- Returns whether a token is an identifier **([a-zA-Z][a-zA-Z0-9]*)**

private boolean **isNumericConstant**(String token)

- Returns whether a token is a numeric constant **(-?[0-9])**

private boolean **isStringConstant**(String token)

- Returns whether a token is a numeric constant **(“[a-zA-Z0-9\- \r\n]*”)**

private boolean **isAlphabet**(int character)

- Returns whether a character is part of the language alphabet **([a-zA-Z0-9”\- \r\n])**

private boolean **isLetter**(int character)

- Returns whether a character is a letter**([a-zA-Z])**

private boolean **isDigit**(int character)

- Returns whether a character is part of the language alphabet **([0-9])**

private boolean **isSeparator**(int character)

- Returns whether a character is part of the language alphabet **([0-9])**

private boolean **isQuote**(int character)

- Returns whether a character is a quote

private boolean **isMinus**(int character)

- Returns whether a character is a minus

SymbolTable class

- Collisions are resolved using linked lists
- Hashtable size is constant
- Position inside the symbol table is: **p = listIndex * size + hash(token)**

public int **put**(String token)

- Stores a token inside the symbol table
- Average Complexity: **O(α)**
- Pre:
    - Token is not already stored inside the symbol table
- Post:
    - Result is the position of the added token

String **get**(int position):

- Average Complexity: **O(α)**
- Post:
    - Result is the token from the given position if it is found
    - Result is **null** if no token is found at that position

int **lookup**(String token):

- Average Complexity: **O(α)**
- Post:
    - Result is the position inside the symbol table of a given token
    - Result is **-1** if the token is not found

Test Cases

Input file contents:

```
begin
  define mytype
  begin
     integer n1
     integer n2
  end

  mytype obj
  integer k
  n1 of obj becomes 1
  n2 of obj becomes 1
  read k
  if k smaller 1
  begin
     print "Error this will not work"
     stop
  end
  if k smaller 3
  begin
     print 1
     stop
  end
  k becomes k minus 2
  each i from 1 to k
  begin
     integer new
     new becomes n1 of obj plus n2 of obj
     if i is k
     begin
        print new
        stop
     end
     n1 of obj becomes n2 of obj
     n2 of obj becomes new
  end
end
```

PIF output file:

```
begin, -1
define, -1
IDENTIFIER, 6
begin, -1
integer, -1
IDENTIFIER, 9
integer, -1
IDENTIFIER, 0
end, -1
IDENTIFIER, 6
IDENTIFIER, 1
integer, -1
IDENTIFIER, 7
IDENTIFIER, 9
of, -1
IDENTIFIER, 1
becomes, -1
CONSTANT, 19
IDENTIFIER, 0
of, -1
IDENTIFIER, 1
becomes, -1
CONSTANT, 19
read, -1
IDENTIFIER, 7
if, -1
IDENTIFIER, 7
smaller, -1
CONSTANT, 19
begin, -1
print, -1
CONSTANT, 4
stop, -1
end, -1
if, -1
IDENTIFIER, 7
smaller, -1
CONSTANT, 11
begin, -1
print, -1
CONSTANT, 19
stop, -1
end, -1
```

```
IDENTIFIER, 7
becomes, -1
IDENTIFIER, 7
minus, -1
CONSTANT, 10
each, -1
IDENTIFIER, 5
from, -1
CONSTANT, 19
to, -1
IDENTIFIER, 7
begin, -1
integer, -1
IDENTIFIER, 16
IDENTIFIER, 16
becomes, -1
IDENTIFIER, 9
of, -1
IDENTIFIER, 1
plus, -1
IDENTIFIER, 0
of, -1
IDENTIFIER, 1
if, -1
IDENTIFIER, 5
is, -1
IDENTIFIER, 7
begin, -1
print, -1
IDENTIFIER, 16
stop, -1
end, -1
IDENTIFIER, 9
of, -1
IDENTIFIER, 1
becomes, -1
IDENTIFIER, 0
of, -1
IDENTIFIER, 1
IDENTIFIER, 0
of, -1
IDENTIFIER, 1
becomes, -1
IDENTIFIER, 16
```

end, -1
end, -1

ST output file:

```
0, n2, hash=0, index=0
10, 2, hash=0, index=1
1, obj, hash=1, index=0
11, 3, hash=1, index=1
4, "Error this will not work", hash=4, index=0
5, i, hash=5, index=0
6, mytype, hash=6, index=0
16, new, hash=6, index=1
7, k, hash=7, index=0
9, n1, hash=9, index=0
19, 1, hash=9, index=1
```