**Component Design:**

**Main**

main (int argc, const char \*argv[])

Create a scanner

Create a token

Have scanner start reading the file

Check for <identifiers>

Return 0

**Print**

Print (char source\_name[], char date[])

Set the filename to source\_name

get current time

set the page number to zero

printLine (char line[])

increment the line count

if line\_count> page height Print header

print the string argument

PrintPageHeader ()

Print header (page number, source file name , current date)

PrintToken (Token \*token)

Increment the line count

Default -> print token

printTreeRecursive (Identifier \*identifier)

set LineNumberList

get TokenString

send identifier to binary tree

printTree (Identifier \*identifier)

set proper labels and spaces

print identifier tree

**Scanner**

Scanner (FILE \*source\_file, char source\_name[], char date[], Print printer)

Src\_file= source\_file

Copy (src\_name, source name)

Copy (todays\_date, date)

Initialize char table to identify what type of char we are looking at

Initialize Line numer=0

Source line [0] = ‘\0’

getSourceLine(char source\_buffer)

create source buffer

create false Boolean

get a line from the filestream

if line received then true

return Boolean

getToken()

initialize a character code variable

skip past all the blanks

examine ch for LETTER, DIGIT, QUOTE, EOF, or SPECIAL

call appropriate function depending on ch

return new\_token

getChar(char souce\_buffer[])

set a temp char to EOF

if at the end of line ->return null character

else return the char at the index

skipBanks (char source\_buffer[])

skip past the blanks

return pointer to the first non blank character

skipComments (char source\_buffer[])

skip past the comments

return pointer to the first non blank character

watch for the EOF character

getWord (char \*str, char \*token\_ptr. Token \*tok)

Extract the word

Downshift the word, to make it lower case

Check if the word is a reserved word

If is not a reserved word its an identifier

Set token to identifier

getNumber (char \*str, char \*token\_ptr, Token \*tok)

extract number and convert it to a literal number

check if real or float

temp string number

set the token type to NUMBER

getString (char \*str, char \*token\_ptr, Token \*tok)

Initialize a temporary string

Whie char ch is not a ‘\’’

Read more characters

Append characters to temp string

Set the setType to STRING\_LIT

Set the setCode to STRING

getSpecial (char \*str, char \*token\_ptr, Token \*tok)

initialize the temp string

check for character operators (:,<,>,.,|)

read next character

if (= or .) -> append both to temp string

else -> append first character to the temp string

downshiftWord (char word[])

make all characters in the incoming word lower case

isReservedWord (char \*str, Token \*tok)

Scan the token table for reserved words

If it is a reserved word -> set the token code member ->return True

Else -> return False

getLineNumer ()

return line\_number

**Token**

Token ()

Initialize variables for binary search tree (lines, left, righ)

setCode (TokenCode newCode)

set newCode

getCode ()

Return token code member

setTokenString (string s)

setTokenString = s

getTokenString (string s)

return Token code member

**LineNumberList**

LineNumberList ()

Set val to 0

Set next to NULL

setLineNumberList (int num)

Set lineNumber =num

getLineNumberList ()

return LineNumber

setNextLineNumberList (LineNumberList \*next)

Set nextLineNumber =next

getNextLineNumberList ()

return nextLineNumber

**Identifier**

Identifier (string val)

Set list = NULL

Set literal equal to val

tokenString is equal to val

setLeftChild (Identifier \*tok)

set leftChild = tok

getLeftChild()

return leftChild

setRightChild (Identifier \*tok)

set RightChild = tok

getRightChild()

return rightChild

getTokenString()

return literal

addToLineNumberList (LineNumberList \*listItem)

if tmp is NULL list is listItem

else tmp= next line numner

setNextLineNumber to tmp

getLineNumberList()

return list

**IdentifierBinaryTree**

IdentifierBinaryTree()

Set value of treeRoot = NULL

depthFirstDeleteTree(Identifier \*tok)

if tok->getLeftChild not NULL

getLeftChild

if tok->getRightChild not NULL

getRighChild

delete tok

setTreeRoot(Identifier \*root)

Set value of treeRoot = root

getTreeRoot()

return treeRoot

addIdentifier2(Identifier\* &head, Identifier\* tok, int lineNum)

set false boolean called success

if head= NULL

head= tok

add head ToLineNumberList

set success to true

getTokenString and set leftChild, rightChild or head appropriately

**Integer**

Integer()

Set value to “INTEGER”

Return int

**Real**

Real()

Set value to “REAL”

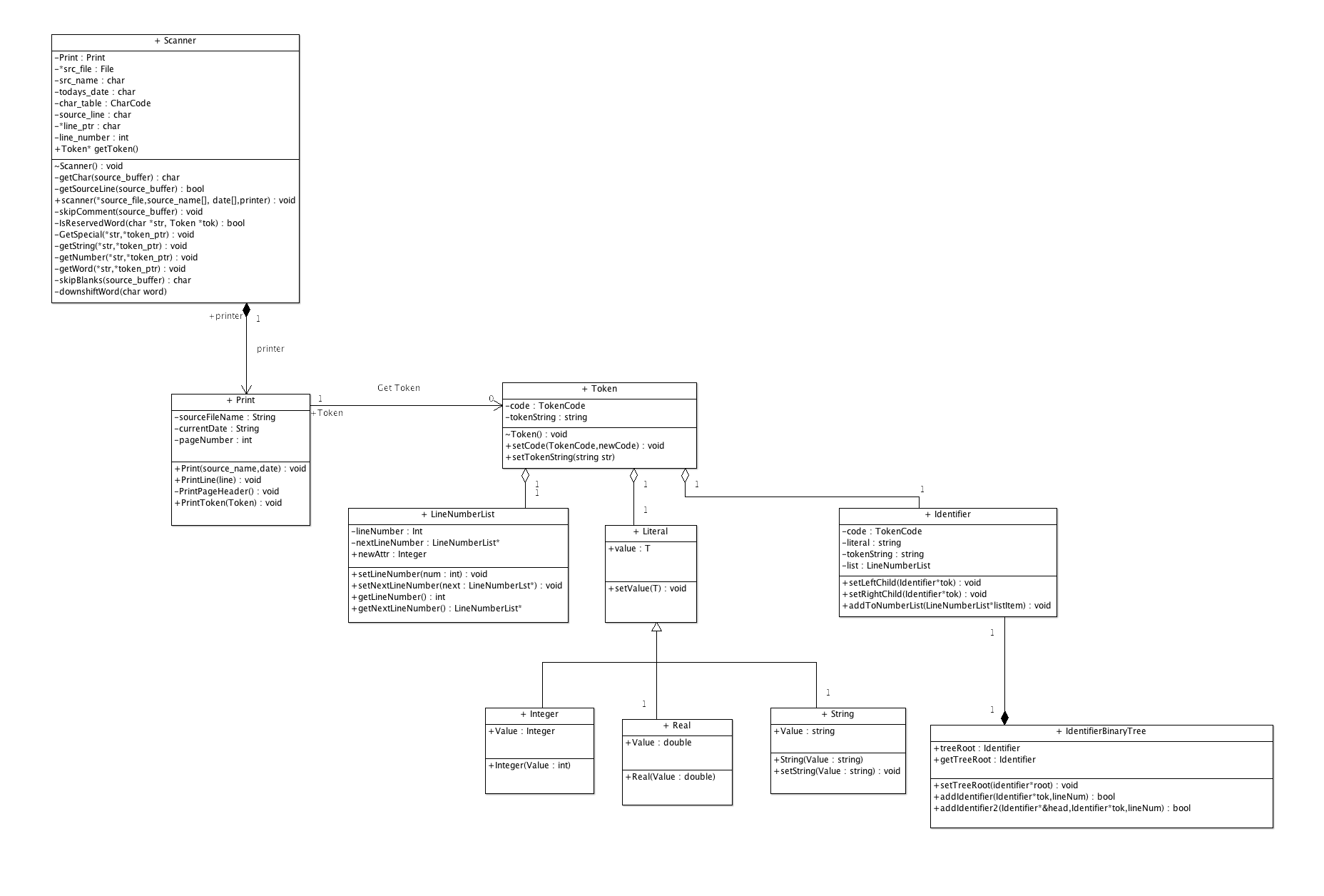
Return double

**String**

setString(string value)

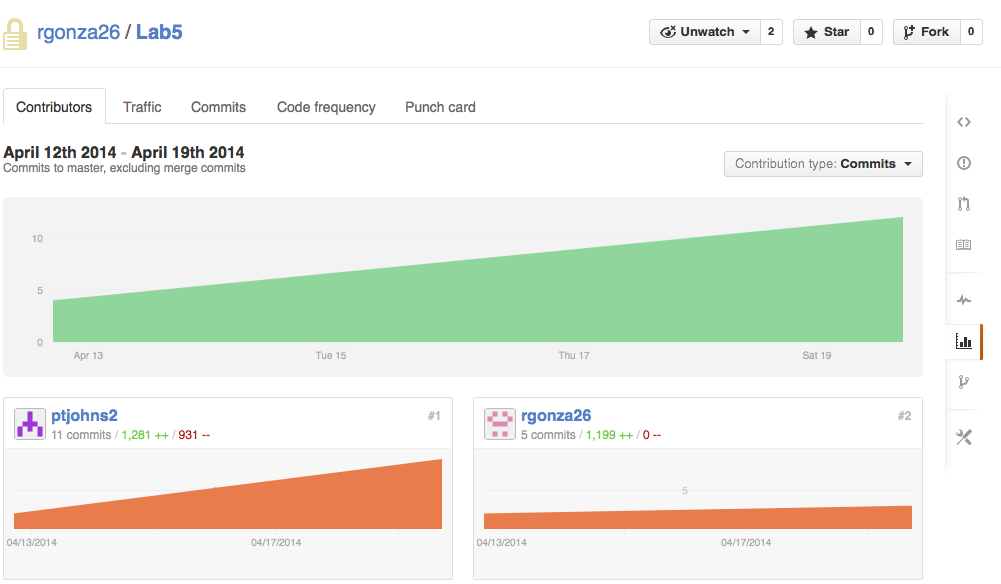
Set value to “String”

Return value

**UML DIAGRAM**

**Team Contribution Table**

|  |  |  |
| --- | --- | --- |
| Name | Login | Score (0= No contribution 2= good contribution) |
| Peter Johnson | ptjohns2 | 2 |
| Roberto Gonzalez | rgonza26 | 2 |
|  |  |  |
| URL to Repository: |  | https://github.com/rgonza26/Lab5 |



**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| main | argv |  |  |  |
| Test 1 | V | Program Produces Expected Results | Expected result | |
| Test 2 | I | Application Crashes | Expected result | |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| print | sourceFileName |  |  |  |
| Test 3 | V | File Name matches valid flie name | Expected result | |
| Test 4 | I | File Name does not match valid file name | Expected result | |
| printLine | line |  |  |  |
| Test 5 | V | prints line | Expected result | |
| Test 6 | I | prints nothing | Expected result | |
| printPageHeader |  |  |  |  |
| Test 7 |  | prints header | Expected result | |
| printToken | Token |  |  |  |
| Test 8 | V (4) | Prints the integer literal | Expected result | |
| Test 9 | V (3.1) | Prints the Float literal | Expected result | |
| Test 10 | V ("test") | Prints the STRING literal | Expected result | |
| Test 11 | V ("PROGRAM") | Prints the Token | Expected result | |
| Test 12 | I | Nothing Prints NO\_Token type | Expected result | |
| printTreeRecursive | Identifier |  |  |  |
| Test | V | Identifier gets added to leftChild, RightChild | Expected Result | |
| Test | I | tree error |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| Scanner | Tested during Main |  |  |  |
|  |  |  |  | |
| getSourceLine | Tested during Main |  |  | |
|  |  |  |  |  |
| getToken | Tested during Main |  |  | |
|  |  |  |  | |
| getChar | source\_buffer |  |  |  |
| Test 13 | V ("test) | returns 't' | Expected result | |
| Test 14 | I "" | returns" | Expected result | |
| skipBlanks | source\_buffer |  |  | |
| Test 15 | V ( two spaces) | returns 2 | Expected result | |
| Test 16 | I (spaces here" | returns 1 | Expected result | |
| skipComment | source\_buffer |  |  | |
| Test 17 | V "this is a {comment}" | Removes Comment from source line | Expected result | |
| Test 18 | I "this is a comment" | Prints the line | Expected result | |
| getWord | ch |  |  | |
| Test 19 | V 'a' | Tested during print | Expected result | |
| Test 20 | I '{' | Empty token string | Expected result | |
| getNumber | ch |  |  | |
| Test 21 | V '5' | Tested during print | Expected result | |
| Test 22 | I '{' | Empty token string | Expected result | |
| getString | ch |  |  | |
| Test 23 | V | Tested during print | Expected result | |
| Test 24 | I '{' | Empty token string | Expected result | |
| getSpecial | str |  |  | |
| Test 25 | V "program" | returns 1 | Expected result | |
| Test 26 | I 'nothing" | returns false | Expected result | |
| downshitfWord | char |  |  | |
| Test 27 | V 'OK' | ok | Expected result | |
| Test 28 | I '{' | Empty token string | Expected result | |
| isREservedWord | ch |  |  | |
| Test 29 | V"DO" | Tested during print | Expected result | |
| Test 30 | I '{' | Empty token string | Expected result | |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| Token | Tested during Main |  |  |  |
|  |  |  |  | |
| setCode | Tested during Main |  |  | |
| getcode | TokenCode |  |  |  |
| Test 31 | V | new code is set | Expected result | |
| Test 32 | I | error no code set | Expected result | |
| setTokenString | Tested during Main |  |  | |
| getTokenString | String |  |  |  |
| Test 33 | V | val is set to token string | Expected result | |
| Test 34 | I | Val is not set error | Expected result | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| LineNumberList |  |  |  |  |
|  |  |  |  | |
| setLineNumber | int |  |  | |
| Test 35 | V | lineNumber = num | Expected result | |
| Test 36 | I | invalid | Expected result | |
| setNextLineNumber | LineNumberList |  |  |  |
| Test 37 | V | nextLineNumber = next | Expected result | |
| Test 38 | I | Invalid | Expected result | |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| Identifier |  |  |  |  |
|  |  |  |  | |
| setLeftChild | Identifier |  |  | |
| Test 39 | V | this leftChild | Expected result | |
| Test 40 | I | Error | Expected result | |
| setRightChild | Identifier |  |  | |
| Test 41 | V | set RightChild | Expected result | |
| Test 42 | I | Error | Expected result | |
| addToLineNumberList | LineNumberList |  |  | |
| Test 43 | V | get NextLineNumber | Expected result | |
| Test 44 | I | Error | Expected result | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| IdentifierBinaryTree |  |  |  |  |
|  |  |  |  | |
| depthFirstDeleteTree | identifier |  |  | |
| Test 45 | V | getLeftChild or getRightChild | Expected result | |
| Test 46 | I | invalid | Expected result | |
| setTreeRoot | Identifier |  |  |  |
| Test 47 | V | treeRoot=root | Expected result | |
| Test 48 | I | Invalid | Expected result | |
|  |  |  |  |  |
|  |  |  |  |  |
| Function | Parameter (V->Valid, I ->Invalid) | Expected Result | Acutal Result | |
| Integer | int |  |  |  |
|  |  |  |  | |
| Real | double |  |  | |
|  |  |  |  | |
|  |  |  |  | |
| String | String |  |  |  |
|  |  |  |  | |
|  |  |  |  | |