Scanner

Goian Tudor – 933/2

https://github.com/VaruTudor/Formal-Languages-and-Compiler-Design

Requirement - Implement the Symbol Table (ST) as the specified data structure, with the corresponding operations

Implementation

- I decided to use a Hash Table as the data structure for the Symbol Table. The main advantage of hash tables over other data structures is speed . The access time of an element is on average O(1), therefore lookup could be performed very fast.
- For each entry in the ST a Deque (which under the hood is a doubly-linked list) will be used. It ensures that there will be no conflicts – when elements hash to the same position, they will be part of the same deque.
- I use modular hashing (the hash function is simply h(k) = k mod m for some m - size. The value k is an integer hash code generated from the key. If m is a) where for strings I add the ASCII (American Standard Code for Information Interchange) codes of each composing character.
- Supported operations
    - add(key)

        input: key – the token (string)

        output: getPosition(key)
    - remove(key)

        input: key - the token (string)

        output: -

- contains(key)

  input: key - the token (string)

  output: true/false
- getPosition(key)

  input: key - the token (string)

  output: (listPosition, listIndex) – listPosition is the position in the outer list and listIndex is the position in the current deque

Requirement - Implement a scanner (lexical analyzer): Implement the scanning algorithm and use ST from lab 2 for the symbol table.

Implementation

- Supported operations
  - getStringToken(line, index)

    input: line – a line read from the file (string), index – the current position

    output: (token(string), index(int))
  - isPartOfOperator(char)

    input: char – a character (string)

    output: true/false
  - getOperatorToken(line, index)

    input: line – a line read from the file (string), index – the current position

    output: (token(string), index(int))
  - tokenize(line)

    input: line – a line read from the file (string)

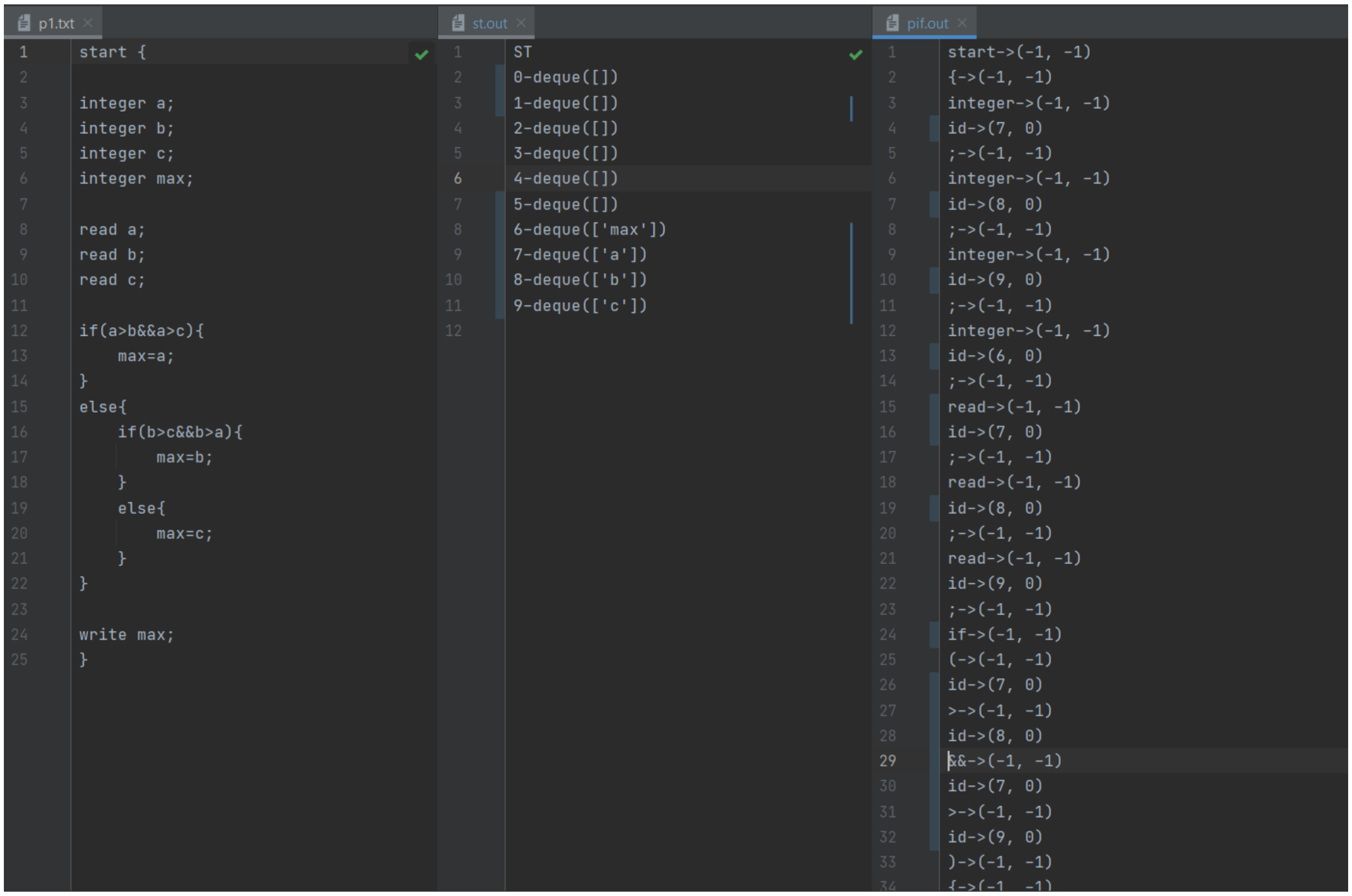
    output: tokens(list(string)) – the tokens identified in the given line

- Tokenizing

  For a line, character by character, check if the current one is an operator, separator, string or belongs in the ST (meaning it is constant or identifier) .

- Scanning

  Line by line performs tokenizing, adds each identifier or constant to the ST and the keywords, separators and operators /w (-1,-1). For the constants the code will be “const” and for identifiers “id”



p1.txt

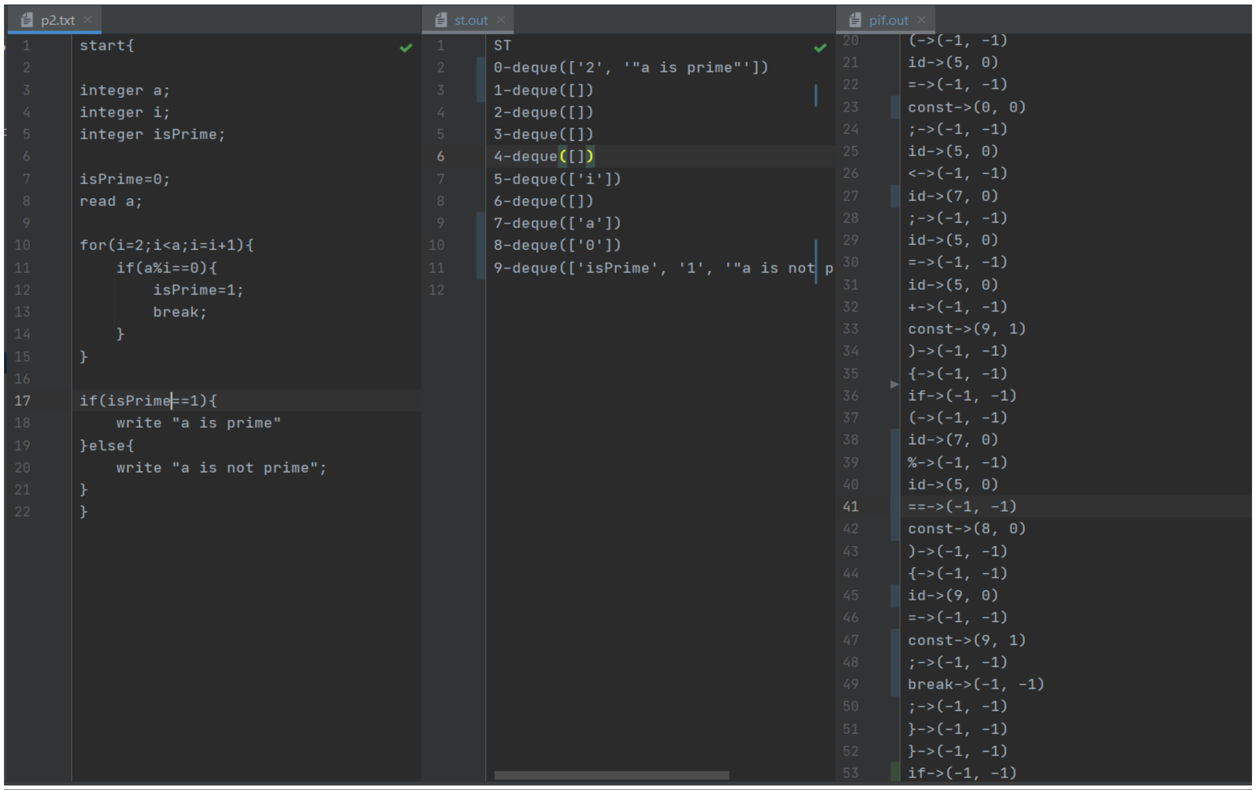
```
start {

integer a;
integer b;
integer c;
integer max;

read a;
read b;
read c;

if(a>b&&a>c){
    max=a;
}
else{
    if(b>c&&b>a){
        max=b;
    }
    else{
        max=c;
    }
}

write max;
}
```

st.out

```
ST
0-deque([])
1-deque([])
2-deque([])
3-deque([])
4-deque([])
5-deque([])
6-deque(['max'])
7-deque(['a'])
8-deque(['b'])
9-deque(['c'])
```

pif.out

```
start->(-1, -1)
{->(-1, -1)
integer->(-1, -1)
id->(7, 0)
;->(-1, -1)
integer->(-1, -1)
id->(8, 0)
;->(-1, -1)
integer->(-1, -1)
id->(9, 0)
;->(-1, -1)
integer->(-1, -1)
id->(6, 0)
;->(-1, -1)
read->(-1, -1)
id->(7, 0)
;->(-1, -1)
read->(-1, -1)
id->(8, 0)
;->(-1, -1)
read->(-1, -1)
id->(9, 0)
;->(-1, -1)
if->(-1, -1)
(->(-1, -1)
id->(7, 0)
>->(-1, -1)
id->(8, 0)
&&->(-1, -1)
id->(7, 0)
>->(-1, -1)
id->(9, 0)
)->(-1, -1)
{->(-1, -1)
```



p2.txt

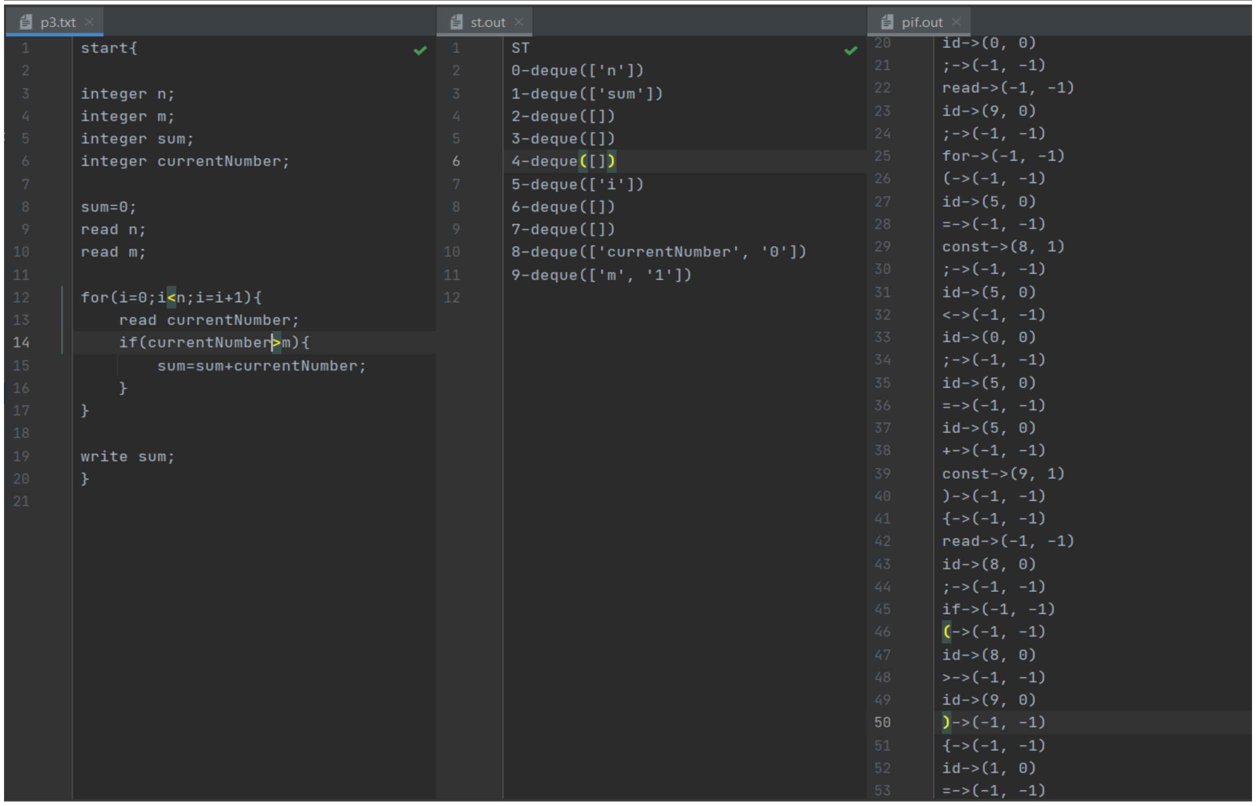
```
start{

integer a;
integer i;
integer isPrime;

isPrime=0;
read a;

for(i=2;i<a;i=i+1){
    if(a%i==0){
        isPrime=1;
        break;
    }
}

if(isPrime==1){
    write "a is prime"
}else{
    write "a is not prime";
}
}
```

st.out

```
ST
0-deque(['2', '"a is prime"'])
1-deque([])
2-deque([])
3-deque([])
4-deque([])
5-deque(['i'])
6-deque([])
7-deque(['a'])
8-deque(['0'])
9-deque(['isPrime', '1', '"a is not p
```

pif.out

```
(->(-1, -1)
id->(5, 0)
=->(-1, -1)
const->(0, 0)
;->(-1, -1)
id->(5, 0)
<->(-1, -1)
id->(7, 0)
;->(-1, -1)
id->(5, 0)
=->(-1, -1)
id->(5, 0)
+->(-1, -1)
const->(9, 1)
)->(-1, -1)
{->(-1, -1)
if->(-1, -1)
(->(-1, -1)
id->(7, 0)
%->(-1, -1)
id->(5, 0)
==->(-1, -1)
const->(8, 0)
)->(-1, -1)
{->(-1, -1)
id->(9, 0)
=->(-1, -1)
const->(9, 1)
;->(-1, -1)
break->(-1, -1)
;->(-1, -1)
}->(-1, -1)
}->(-1, -1)
if->(-1, -1)
```



p3.txt

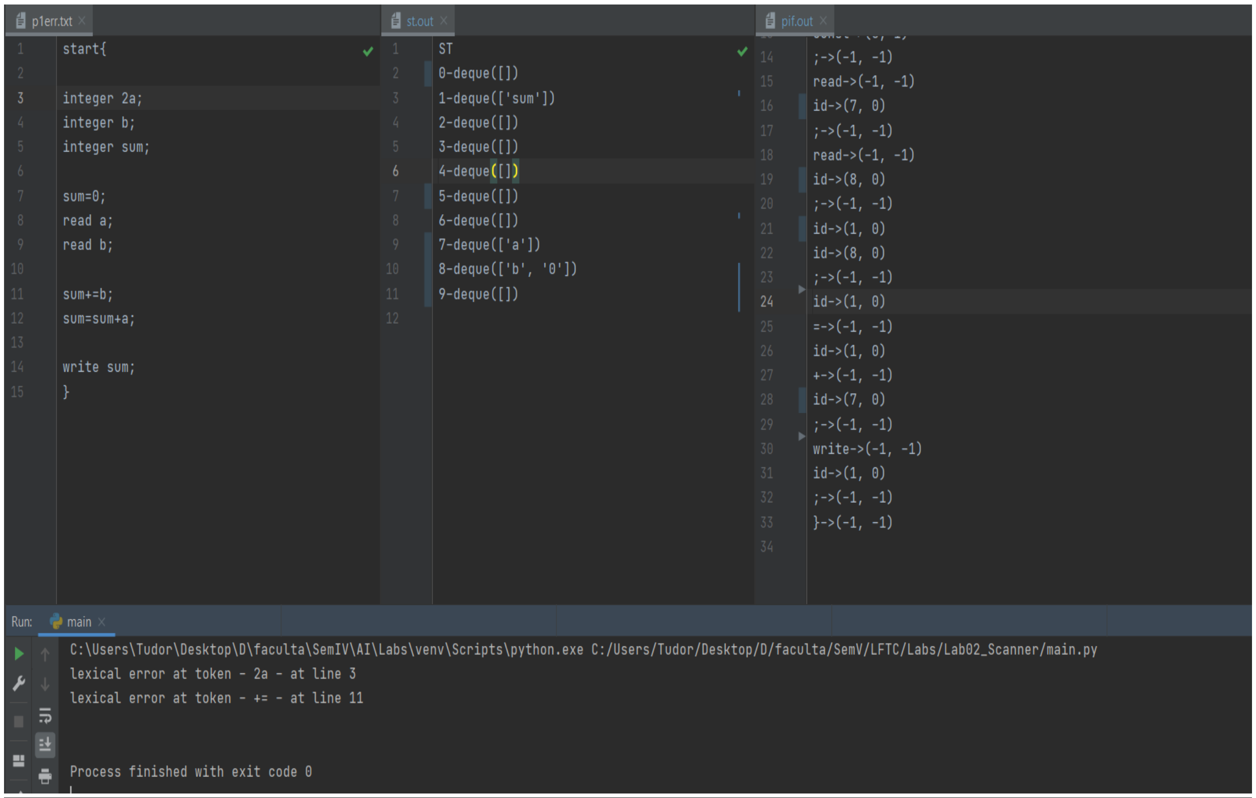
```
start{

integer n;
integer m;
integer sum;
integer currentNumber;

sum=0;
read n;
read m;

for(i=0;i<n;i=i+1){
    read currentNumber;
    if(currentNumber>m){
        sum=sum+currentNumber;
    }
}

write sum;
}
```

st.out

```
ST
0-deque(['n'])
1-deque(['sum'])
2-deque([])
3-deque([])
4-deque([])
5-deque(['i'])
6-deque([])
7-deque([])
8-deque(['currentNumber', '0'])
9-deque(['m', '1'])
```

pif.out

```
id->(0, 0)
;->(-1, -1)
read->(-1, -1)
id->(9, 0)
;->(-1, -1)
for->(-1, -1)
(->(-1, -1)
id->(5, 0)
=->(-1, -1)
const->(8, 1)
;->(-1, -1)
id->(5, 0)
<->(-1, -1)
id->(0, 0)
;->(-1, -1)
id->(5, 0)
=->(-1, -1)
id->(5, 0)
+->(-1, -1)
const->(9, 1)
)->(-1, -1)
{->(-1, -1)
read->(-1, -1)
id->(8, 0)
;->(-1, -1)
if->(-1, -1)
(->(-1, -1)
id->(8, 0)
>->(-1, -1)
id->(9, 0)
)->(-1, -1)
{->(-1, -1)
id->(1, 0)
=->(-1, -1)
```

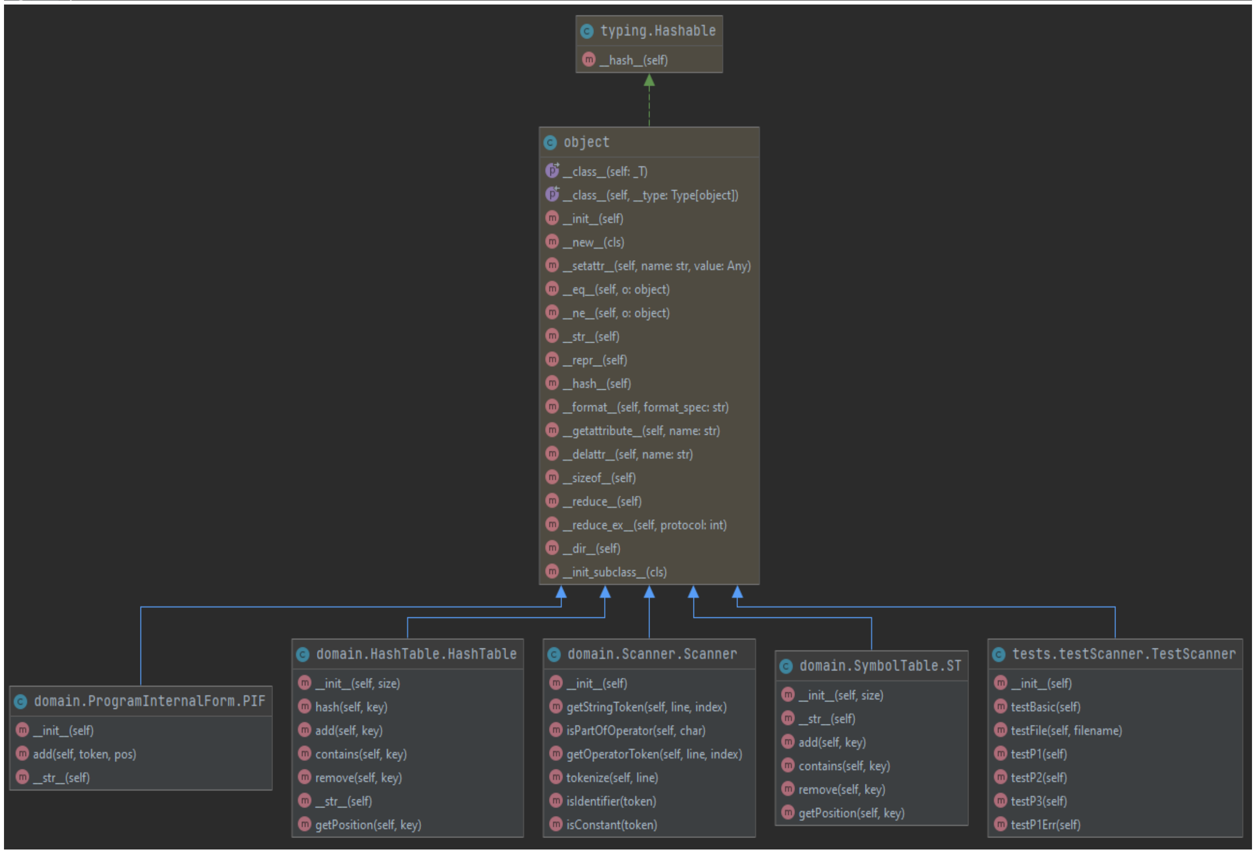


```
start{

integer 2a;
integer b;
integer sum;

sum=0;
read a;
read b;

sum+=b;
sum=sum+a;

write sum;
}
```

```
ST
0-deque([])
1-deque(['sum'])
2-deque([])
3-deque([])
4-deque([])
5-deque([])
6-deque([])
7-deque(['a'])
8-deque(['b', '0'])
9-deque([])
```

```
;->(-1, -1)
read->(-1, -1)
id->(7, 0)
;->(-1, -1)
read->(-1, -1)
id->(8, 0)
;->(-1, -1)
id->(1, 0)
id->(8, 0)
;->(-1, -1)
id->(1, 0)
=->(-1, -1)
id->(1, 0)
+->(-1, -1)
id->(7, 0)
;->(-1, -1)
write->(-1, -1)
id->(1, 0)
;->(-1, -1)
}->(-1, -1)
```

```
C:\Users\Tudor\Desktop\D\faculta\SemIV\AI\Labs\venv\Scripts\python.exe C:/Users/Tudor/Desktop/D/faculta/SemV/LFTC/Labs/Lab02_Scanner/main.py
lexical error at token - 2a - at line 3
lexical error at token - += - at line 11


Process finished with exit code 0
```



typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
domain.ProgramInternalForm.PIF
__init__(self)
add(self, token, pos)
__str__(self)
domain.HashTable.HashTable
__init__(self, size)
hash(self, key)
add(self, key)
contains(self, key)
remove(self, key)
__str__(self)
getPosition(self, key)
domain.Scanner.Scanner
__init__(self)
getStringToken(self, line, index)
isPartOfOperator(self, char)
getOperatorToken(self, line, index)
tokenize(self, line)
isIdentifier(token)
isConstant(token)
domain.SymbolTable.ST
__init__(self, size)
__str__(self)
add(self, key)
contains(self, key)
remove(self, key)
getPosition(self, key)
tests.testScanner.TestScanner
__init__(self)
testBasic(self)
testFile(self, filename)
testP1(self)
testP2(self)
testP3(self)
testP1Err(self)