Lab2 FLCD - Stefan Guliciuc

Link github: https://github.com/stefan99x/FLCD

Filename: Node.py

Class Node:

```
class Node:
    def __init__(self, identifier =""):
        self.identifier = identifier
        self.position = 0
        self.left = None
        self.right = None
```

Legend:

Identifier = represents the Token
Position = as the name implies represents the position in the BinarySearchTree
Left = left node
Right = right node

Filename: NodeOperations.py
Class: NodeOperations

```
class NodeOperations:
    @staticmethod
    def search(root: Node, identifier):
        if root is None:
            return None
        if root.identifier == identifier:
            return root
        if root.identifier > identifier:
            return NodeOperations.search(root.left, identifier)
        if root.identifier < identifier:
            return NodeOperations.search(root.right, identifier)
        return None
```

Description: searches in the binary search tree a node by it’s identifier.

```
    @staticmethod
    def insert(root: Node, node: Node):
        if root is None:
            root = node
        else:
            if root.identifier < node.identifier:
                if root.right is None:
                    root.right = node
                else:
                    NodeOperations.insert(root.right, node)
            else:
                if root.left is None:
                    root.left = node
                else:
                    NodeOperations.insert(root.left, node)
```

Description: As the name implies inserts in our tree a new node.
Params: root => starting node of our Tree
node = > the node to be added.

```
    @staticmethod
    def inOrder(root: Node):
        if root:
            NodeOperations.inOrder(root.left)
            print("#" + root.identifier + "=>" + str(root.position))
            NodeOperations.inOrder(root.right)
```

Description: Prints all the nodes in our tree
Params: root => starting node of our Tree

Filename: BinarySearchTree.py
Class: BinarySearchTree

```
def printBinarySearchTree(self):
    NodeOperations.inOrder(self.root)
```

Description: Prints our binary search tree with our method defined in NodeOperations.py

```
def search(self, identifier):
    return NodeOperations.search(self.root, identifier)
```

Description: Call the search method defined in NodeOperations.py
Params: identifier = the token by we search

```
def insert(self, identifier):
    newNode = Node(identifier)
    if self.root is None:
        self.root = newNode
        newNode.position = self.currentPosition
        self.currentPosition += 1
    node = self.search(identifier)
    if node:
        return node.position
    NodeOperations.insert(self.root, newNode)
    newNode.position = self.currentPosition
    self.currentPosition += 1
```

Description: Inserts a new node in our BinarySearchTree only if that node does not already exists
Params: identifier = the token of our new Node

Filename: SymbolTable.py
Class: SymbolTable

```
def insert(self, identifier):
   return self.tree.insert(identifier)
```

Description: Call the insert method defined in BinarySearchTree.py
Params: identifier = the token of our new Node

```
def print(self):
   self.tree.printBinarySearchTree()
```

Description: Call the print method defined in BinarySearchTree.py
Params: identifier = the token of our new Node