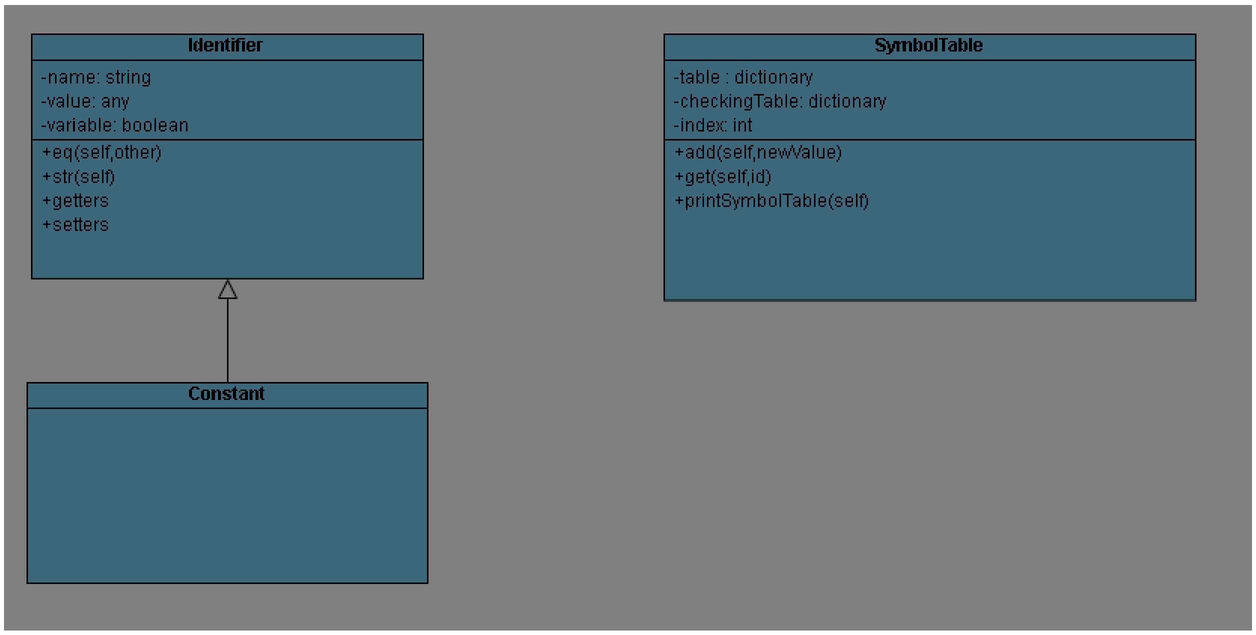
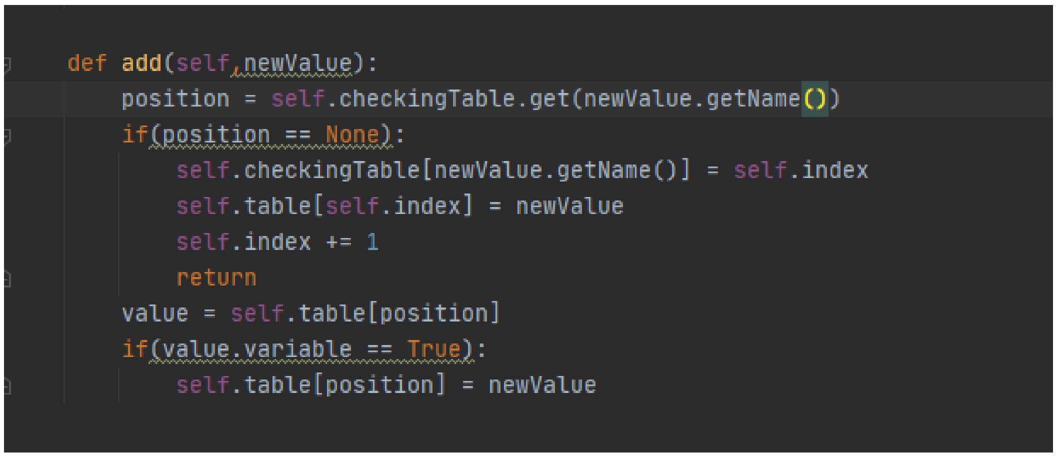
Lab 2:

Github link : https://github.com/PopCristianGabriel/LFTC/tree/master



The class Identifier will hold all types of identifiers in the toy language. I've created also a class Constant which inherits from Identifier, but the field "variable" will be False by default, instead of true, so we can not change the value once it's been created.



```
def add(self,newValue):
    position = self.checkingTable.get(newValue.getName())
    if(position == None):
        self.checkingTable[newValue.getName()] = self.index
        self.table[self.index] = newValue
        self.index += 1
        return
    value = self.table[position]
    if(value.variable == True):
        self.table[position] = newValue
```

This is the add function from the SymbolTable. There are 2 things going on here: the checkingTable and the table itself. The table dictionary will have a key-value pair of type : {id : Identifier | Constant}, and is our Symbol Table. The dictionary checkingTable will hold a key-value pair of type : {name(as a string) : position (as an int)}. This is necessary for checking for duplicates, because , if we were to look for duplicates, we would need to iterate through every value of the symbol table to find it, so instead of that we just look it up in the checking table to get it’s position. Instead of O(n) for searching we have 2 * O(1).