
**Computer Engineering Department**

# Fundamentals of Compiler Design

## Assignment 4

Ali Sedaghi
97521378

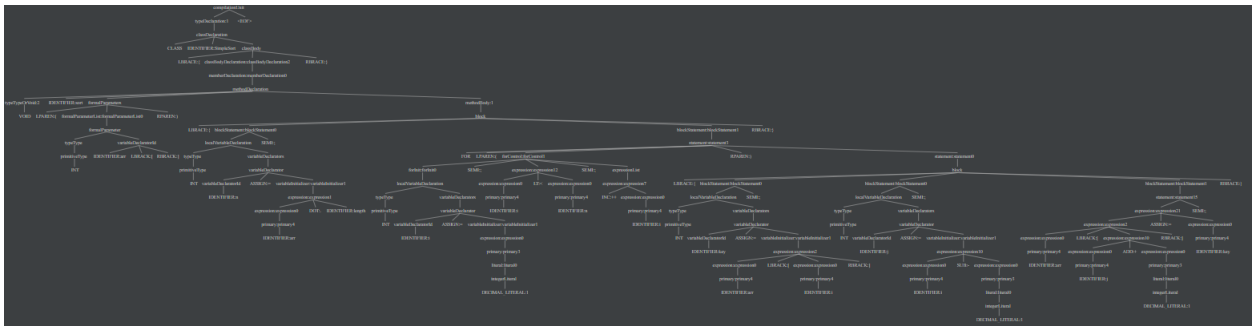
# Table of contents

# Theoretical Questions

## Q1- Parse Tree

تصویر با کیفیت درخت تجزیه مربوط به این سوال درون مسیر زیر وجود دارد:

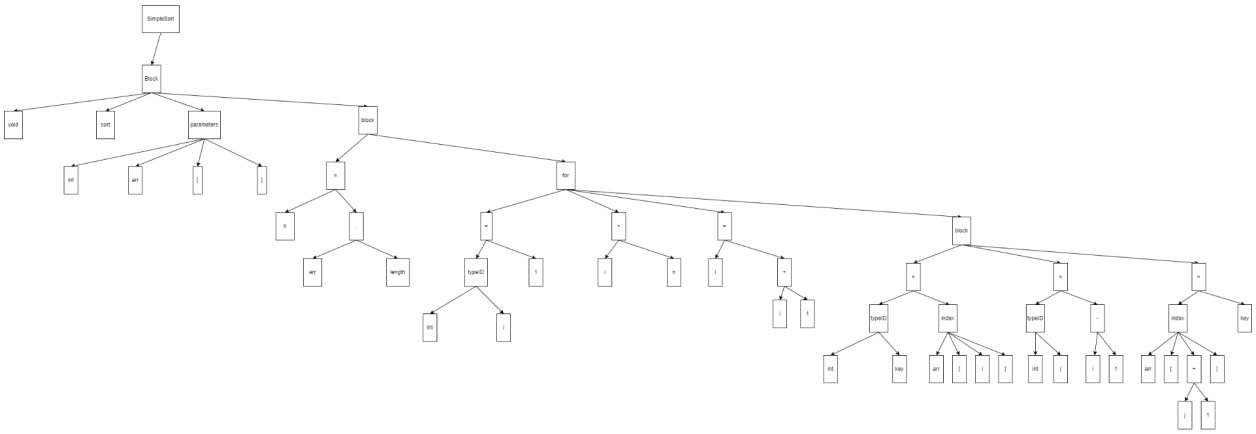
HW4 => docs => T-Q1.svg



## Q2- AST

تصویر با کیفیت درخت خلاصه نحوی مربوط به این سوال درون مسیر زیر وجود دارد:

HW4 => docs => T-Q2.svg




## Q3- 3 Address Code

```
Label SimpleSort .
Label SimpleSort_sort .

        T1 = sizeoff(arr);
        T1 = T1 / 4
```

```
LOOP:
        if !(i < n ) goto END
        key = arr[i]
        j = i - 1
        T1 = j + 1
        arr[T1] = key
        i = i + 1
END:
        goto END
```

# Practical Questions

## Q1- LL(K)

شرط نداشتن Null برقرار است. اما در این گرامر پدیده Left Recursion بسیار وجود دارد. همچنین در مجموعه First بسیاری از Rule های این گرامر عناصر مشترک وجود دارد. پس این گرامر LL1 نیست. پس می‌توان نتیجه گرفت این گرامر LLK نیز نیست.

مثالی از اشتراک در مجموعه First:

```
Type ::= "int" "[" "]"
       | "boolean"
       | "int"
       | Identifier
```

برای Type مشاهده می‌کنیم دو Transition شروع آن با int است.

## Q2- Grammar

گرامرهای انتلر این زبان مینی جاوا درون مسیرهای زیر وجود دارد:

HW4 => Q2 => MiniJavaLexer.g4
HW4 => Q2 => MiniJavaParser.g4

## Q3- AST

گرامرهای انتلر مربوط به درخت خلاصه نحوی زبان مینی جاوا درون مسیرهای زیر وجود دارد:

HW4 => Q3 => MiniJavaLexer.g4
HW4 => Q3 => MiniJavaParser.g4

همچنین فایل‌های زیر مسئول ترسیم درخت خلاصه نحوی (AST) و به دست آوردن کد سه آدرسی (TAC) هستند:

HW4 => AST.py
HW4 => MiniJavaListener.py
HW4 => main.py

```
parser grammar MiniJavaParser;


options { tokenVocab=MiniJavaLexer; }

@parser::members
{
    from AST import AST
    self.ast = AST()
}

program returns [node=None]
    : {$node = self.ast.make_node('program',None,None)}
    m=mainClass ( c=classDeclaration )* EOF
    {self.ast.add_child($node,$m.node)}
    {self.ast.add_child($node,$c.node)}
    {self.ast.print_tree($node)}
    ;

mainClass returns [node=None]
    : 'class' i1=identifier '{' 'public' 'static' 'void' 'main'
    {self.ast.add_brother($i1.node,$i2.node)}
    {self.ast.add_brother($i2.node,$s.node)}
    {$node = self.ast.make_node('main-class',$i1.node,None)}
    ;
```

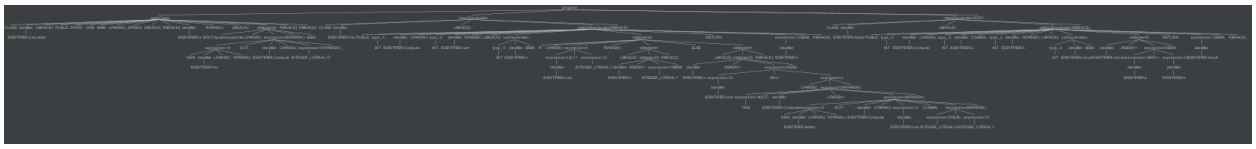
```
import queue


class AST:
    def __init__(self):
        self.root = None
        self.current = None
        self.q = queue.Queue()

    def make_node(self, value, child, brother):
        tree_node = Node(value, child, brother)
        if self.root is None:
            self.root = tree_node
            self.current = tree_node
        return tree_node

    def add_child(self, node, new_child):
        if node.child is None:
            node.child = new_child
        else:
            self.current = node.child
            while self.current.brother is not None:
                self.current = self.current.brother
            self.current.brother = new_child
        self.current = new_child

    def add_brother(self, node, new_brother):
        if node.brother is None:
            node.brother = new_brother
```

## Q4- Results

خروجی درخت تجزیه گرامر مینی جاوا برای کد داده شده به صورت زیر است:

تصویر با کیفیت آن در مسیر زیر موجود است:

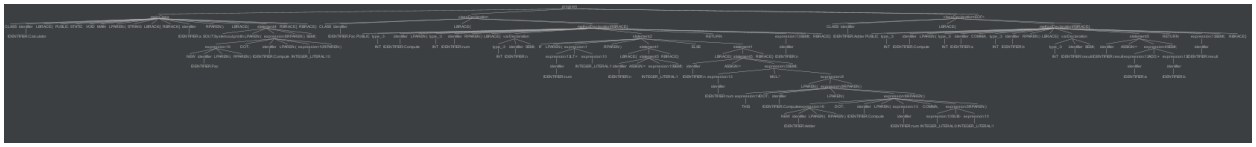
HW4 => docs => P-Q2.svg



خروجی درخت خلاصه نحوی گرامر مینی جاوا برای کد داده شده به صورت زیر است:

تصویر با کیفیت آن در مسیر زیر موجود است:

HW4 => docs => P-Q3.svg



## Q5- BONUS PART TAC

همچنین با پیاده سازی گرامر ویژه برای تولید کدهای سه آدرسه در انتلر که گرامرهای آن در پوشه‌های زیر موجود است، کد 3 آدرسه را نیز تولید کردیم.

HW4 => TAC => MiniJavaLexer.g4
HW4 => TAC => MiniJavaParser.g4

```
parser grammar MiniJavaParser;


options { tokenVocab=MiniJavaLexer; }

@parser::members
{
temp_counter = 0
goto_counter = 0
def create_goto(self):
    self.goto_counter += 1
    return 'L' + str(self.goto_counter)
def create_temp(self):
    self.temp_counter += 1
    return 'T' + str(self.temp_counter)
def remove_temp(self):
    self.temp_counter -= 1
def get_temp(self):
    return 'T' + str(self.temp_counter)
}

program
    : {print('\nTAC generated for statements:')} mainClass ( classDeclaration )* EOF
    ;

mainClass
    : 'class' identifier '{' 'public' 'static' 'void' 'main' '(' 'String' '[' ']' identifier
    {print($s.value_attr)}
    ;
```

```
Reading file ...

TAC generated for statements:

        if (num < 1) goto L1
        T1=
        n = num * T1
        goto L2
L1:     n = 1
L2:
        result = a + b

Process finished with exit code 0
```