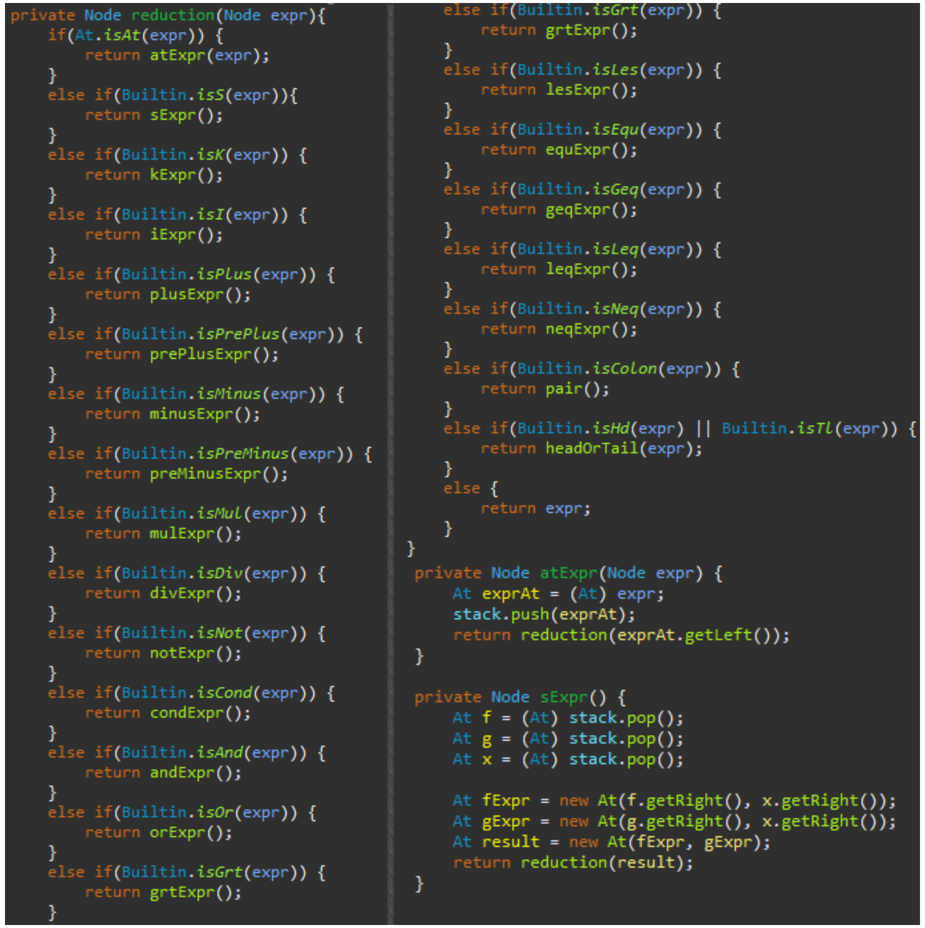
TEAMPROJECT: SASL-COMPILER

TEAM MEMBER: KEVIN JEUTER

31.07.2020

VIRTUAL MACHINE



```java
private Node reduction(Node expr){
    if(At.isAt(expr)) {
        return atExpr(expr);
    }
    else if(Builtin.isS(expr)){
        return sExpr();
    }
    else if(Builtin.isK(expr)) {
        return kExpr();
    }
    else if(Builtin.isI(expr)) {
        return iExpr();
    }
    else if(Builtin.isPlus(expr)) {
        return plusExpr();
    }
    else if(Builtin.isPrePlus(expr)) {
        return prePlusExpr();
    }
    else if(Builtin.isMinus(expr)) {
        return minusExpr();
    }
    else if(Builtin.isPreMinus(expr)) {
        return preMinusExpr();
    }
    else if(Builtin.isMul(expr)) {
        return mulExpr();
    }
    else if(Builtin.isDiv(expr)) {
        return divExpr();
    }
    else if(Builtin.isNot(expr)) {
        return notExpr();
    }
    else if(Builtin.isCond(expr)) {
        return condExpr();
    }
    else if(Builtin.isAnd(expr)) {
        return andExpr();
    }
    else if(Builtin.isOr(expr)) {
        return orExpr();
    }
    else if(Builtin.isGrt(expr)) {
        return grtExpr();
    }
```

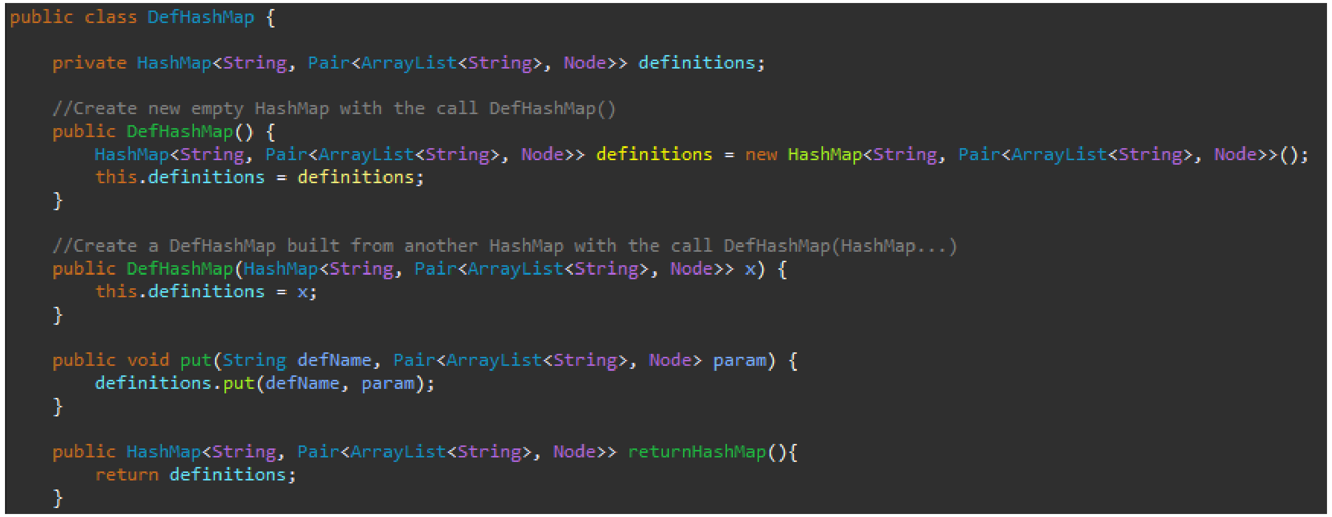
```java
    else if(Builtin.isGrt(expr)) {
        return grtExpr();
    }
    else if(Builtin.isLes(expr)) {
        return lesExpr();
    }
    else if(Builtin.isEqu(expr)) {
        return equExpr();
    }
    else if(Builtin.isGeq(expr)) {
        return geqExpr();
    }
    else if(Builtin.isLeq(expr)) {
        return leqExpr();
    }
    else if(Builtin.isNeq(expr)) {
        return neqExpr();
    }
    else if(Builtin.isColon(expr)) {
        return pair();
    }
    else if(Builtin.isHd(expr) || Builtin.isTL(expr)) {
        return headOrTail(expr);
    }
    else {
        return expr;
    }
}
 private Node atExpr(Node expr) {
     At exprAt = (At) expr;
     stack.push(exprAt);
     return reduction(exprAt.getLeft());
 }

 private Node sExpr() {
     At f = (At) stack.pop();
     At g = (At) stack.pop();
     At x = (At) stack.pop();

     At fExpr = new At(f.getRight(), x.getRight());
     At gExpr = new At(g.getRight(), x.getRight());
     At result = new At(fExpr, gExpr);
     return reduction(result);
 }
```

Short code snippet of my VM

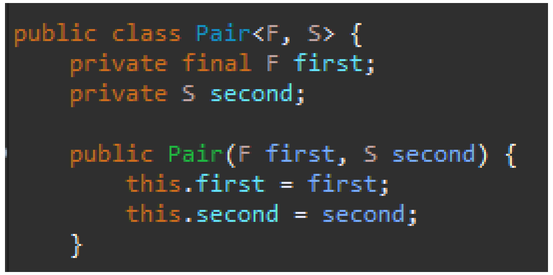
- Each Method calls reduction recursively, to reduce the Program fully.
- Stack to beginning is empty, fills through „atExpr“ until it finds one of the other expressions
- Pairs (Lists) don't get reduced here, but in the print Method
- Lists without „nil“ at the end will not be accepted, because I did not implement […] lists.

DEFHASHMAP AND PAIR

- My Definition Nodes of my AST are created with a left and a right part
  - On the left side I used a HashMap, to bind definition names to their expressions
  - On the right side is the Node, that is to be executed
- For readability purposes I made a class “DefHashMap”, to hide the complexity
- I also needed Pairs, which I had to create myself in Java



```
public class DefHashMap {

    private HashMap<String, Pair<ArrayList<String>, Node>> definitions;

    //Create new empty HashMap with the call DefHashMap()
    public DefHashMap() {
        HashMap<String, Pair<ArrayList<String>, Node>> definitions = new HashMap<String, Pair<ArrayList<String>, Node>>();
        this.definitions = definitions;
    }

    //Create a DefHashMap built from another HashMap with the call DefHashMap(HashMap...)
    public DefHashMap(HashMap<String, Pair<ArrayList<String>, Node>> x) {
        this.definitions = x;
    }

    public void put(String defName, Pair<ArrayList<String>, Node> param) {
        definitions.put(defName, param);
    }

    public HashMap<String, Pair<ArrayList<String>, Node>> returnHashMap(){
        return definitions;
    }
```



```
public class Pair<F, S> {
    private final F first;
    private S second;

    public Pair(F first, S second) {
        this.first = first;
        this.second = second;
    }
```

CHALLENGES AND DIFFERENCES

- Alone, partner left early on.
- I did not implement everything due to time pressure.
  - „Where“ not implemented
  - […] lists not implemented
  - No optimizers implemented
- Virtual Machine differs from the handout.
  - Not as powerful and fast as it could be
  - Can still run programs reliable and fast

CONCLUSION

- No programming experience before Uni
- Never worked independently on a project

Overall it was a very positive, yet time consuming experience. It definitely helped me understand better how to program independently without having a complete set of plans laid out before me.

I would have liked to have a partner to communicate about problems with, since sometimes figuring things out alone without seeing another perspective can be quite challenging.

Luckily, my tutor helped me when I was stuck and couldn‘t figure out what to do.