Team Project

SASL COMPILER

Kevin Jeuter

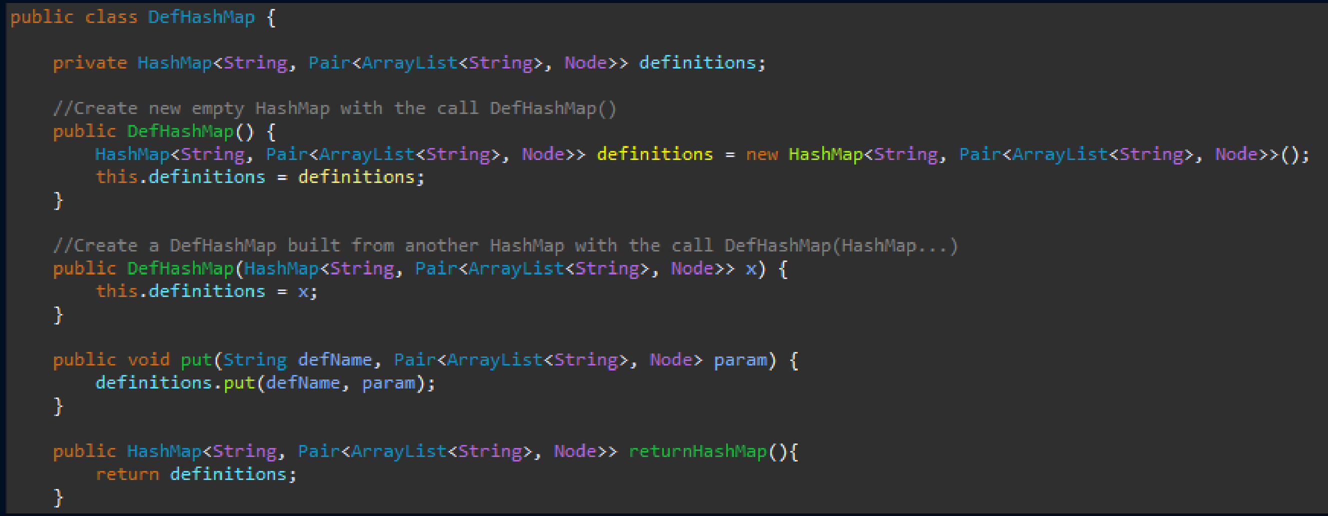
Challenges

- Alone, partner left early on
- Not everything was implemented due to lack of time
  - „Where“ not implemented
  - square bracket notation not implemented
  - No optimization
- Virtual Machine differs from the handout
  - Not as powerful and fast as it could be
  - Can still run programs reliably and relatively fast

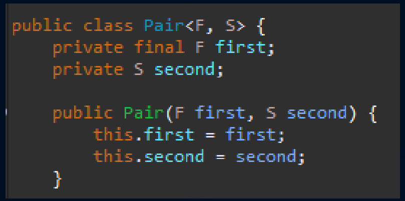
ME WAITING FOR MY CODE TO BUILD TO SEE HOW MANY ERRORS I WILL GET

DefHashMap and pair

- Definition Nodes of 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 contains the expression
- “DefHashMap” used to improve readability
- Pairs were created for the definition nodes



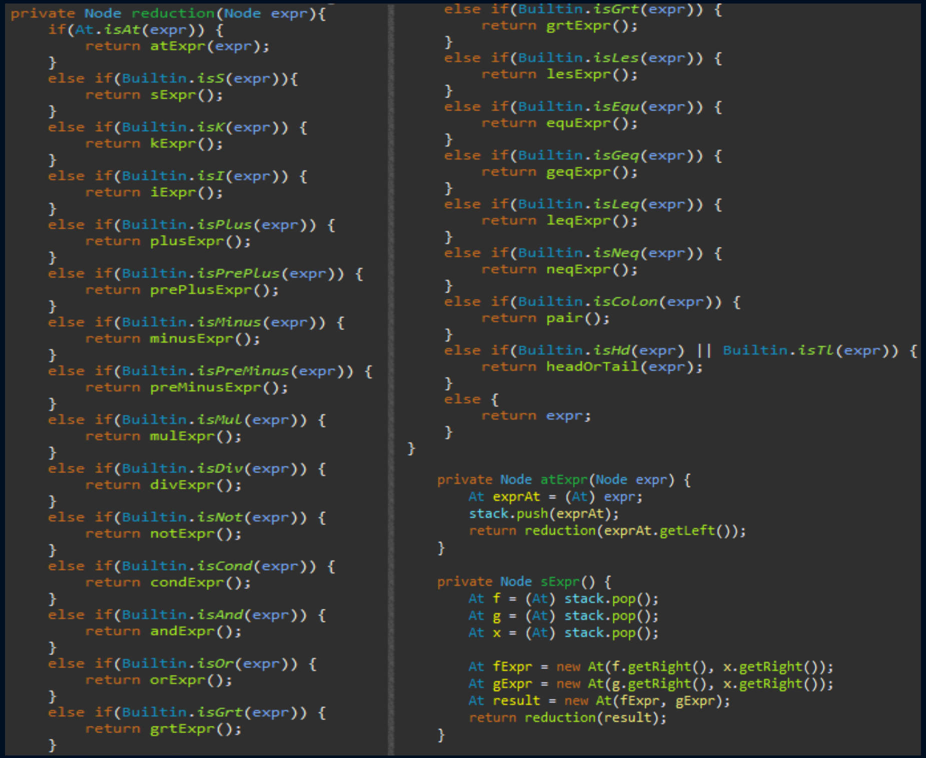
```
public class DefHashMap {

    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;
    }
```

Virtual machine

- Each method calls reduction recursively, to reduce the program completely
- Stack to beginning is empty, fills through „atExpr" until it finds another expression
- Pairs (lists) don't get reduced in the reduction method, but in the print method
- Lists not ending in „nil" will not be accepted, as [...] Notation for lists is not implemented

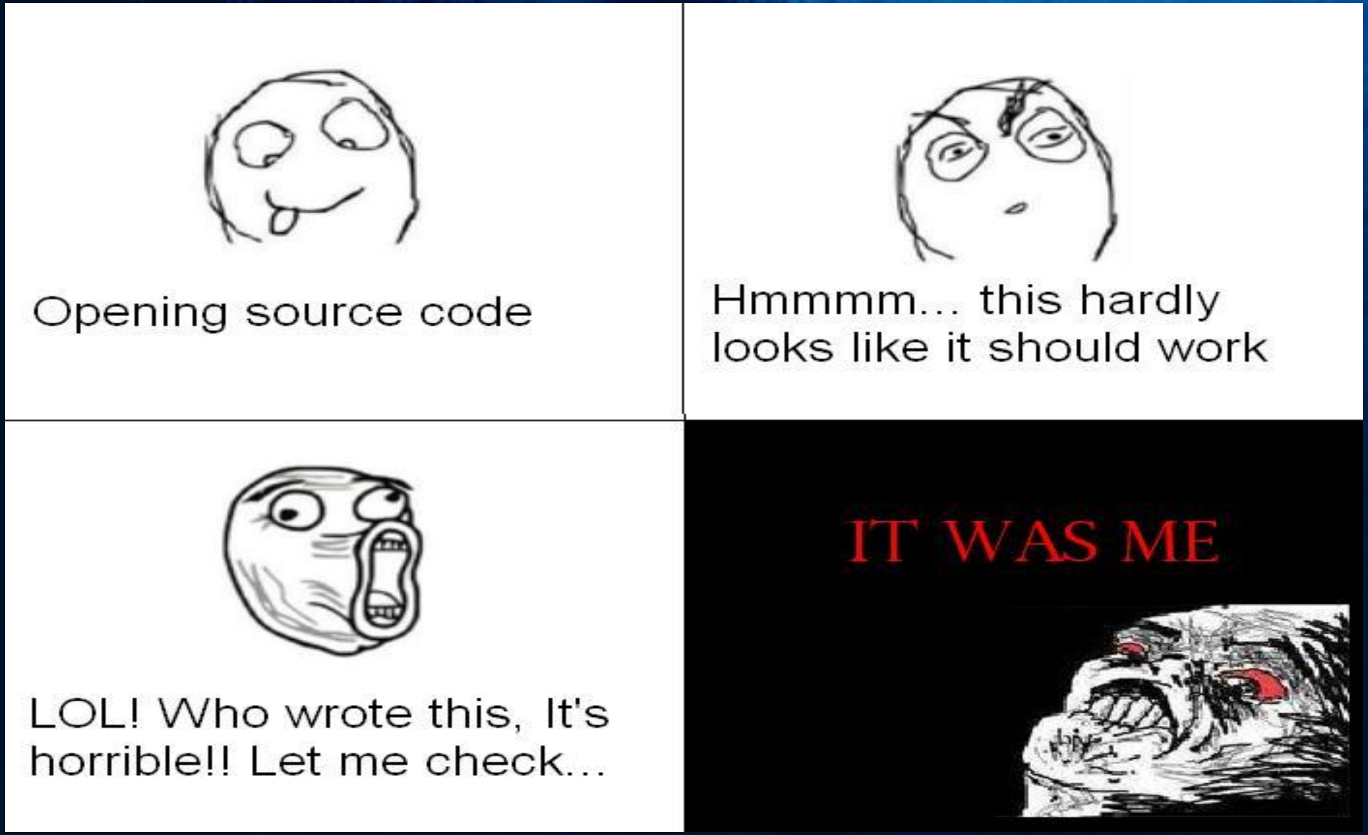


```java
private Node reduction(Node expr){
    if(At.isAt(expr)) {
        return atExpr(expr);
    }
    else if(Builtin.isS(expr)){
        return sExpr();
    }
    else if(Builtin.isK(expr)) {
        return kExpr();
    }
    else if(Builtin.isI(expr)) {
        return iExpr();
    }
    else if(Builtin.isPlus(expr)) {
        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

THANK YOU


Opening source code
Hmmmm... this hardly looks like it should work
LOL! Who wrote this, It's horrible!! Let me check...
IT WAS ME