Compilers - Homework 1

Βασίλειος Πουλόπουλος - 1115201600141

Compila on, Execu on και Clean

Υπάρχει **makefile** το οποίο κάνει compile με την εντολή `make compile` και διαγράφει τα **.class** αρχεία με την εντολή `make clean`.

Για το **execu on** του προγράμματος αφού κάνουμε **make** τρέχουμε την εντολή:

```
java Main tesf-file-1 test-file-2 ... test-file-n
```

Κλάσεις αποθήκευσης των δεδομένων

Έχω υλοποιήσει τις εξής τρεις κλάσεις για να είναι οργανομένη η υλοποίησή μου.

VarClass

Η VarClass, περιέχει τα στοιχεία για τις μεταβλητές των μεθόδων, τα ορίσματά τους, αλλά και τα fields των κλάσεων. Για κάθε ένα από αυτά αποθηκεύω το όνομά του, τον τύπο του και το μέγεθός του. Το μέγεθος υπολογίζεται από το type. Αν είναι ακέραιος αριθμός έχει μέγεθος 4, αν είναι boolean εχει 1, ενώ αν είναι πίνακας ακεραίων έχει 8. Ακολουθεί η υλοποίηση:

```
public class VarClass {
    public String name;
    public String type;
    public int size;

    public VarClass(String name, String type){
        this.name = name;
        this.type = type;
        if (type == "boolean"){
            this.size = 1;
        }
```

```
            else if (type == "int"){
                this.size = 4;
            }
            else this.size = 8;
        }
    }
```

MethodClass

Η MethodClass, περιέχει τα στοιχεία των μεθόδων, ονομα και τύπο, ένα ArrayListVarClass> με τα arguments και ένα LinkedHashMap<String, VarClass> με τις μεταβλητές. Ο λόγος που επέλεξα ArrayList, αντί για Map, για τα arguments, ήταν γιατί για να γίνει check κάθε φορά που καλείται κάποια συνάρτηση για να δω αν είχε δωθεί σωστοί τύποι των arguments στην κλήση της συνάρτησης, έπρεπε να τσεκάρω ότι έχουν δωθεί δεδομένα με σωστούς τύπους. Γι' αυτό το λόγο, έφτιαχνα ένα ακόμα ArrayList με τα args της κλήσης και με ένα loop σύγκρινα τα δύο ArrayLists πολύ εύκολα. Ακολουθεί η υλοποίηση:

```
public class MethodClass {
    public String name;
    public String type;
    public ArrayList<VarClass> args;
    public LinkedHashMap<String, VarClass> vars;

    public MethodClass(String name, String type){
        this.name = name;
        this.type = type;
        this.args = new ArrayList<VarClass>();
        this.vars = new LinkedHashMap<String, VarClass>();
    }
}
```

ClassInfo

Η τελευταία κλάση είναι η **ClassInfo**, στην οποία αποθηκεύω όνομα και τύπο της κλάσης, σε ένα `LinkedHashMap<String, MethodClass>` τις μεθόδους της κλάσης, σε ένα `LinkedHashMap<String,VarClass>` τα fields της και δύο `LinkedHashMap<String,Integer>` για τα offsets των πεδίων και των μεθόδων αντίστοιχα.

```
public class ClassInfo {

    public String name;
    public String parent;
    public LinkedHashMap<String, MethodClass> methods;
    public LinkedHashMap<String,VarClass> fields;
    public LinkedHashMap<String,Integer> fieldOffsets;
    public LinkedHashMap<String,Integer> methodOffsets;


    public ClassInfo(String name, String parent) {
        this.name = name;
        this.parent = parent;
        this.methods = new LinkedHashMap<String, MethodClass>();
        this.fields = new LinkedHashMap<String, VarClass>();
    }
}
```

Visitors

Υλοποίησα δύο visitors. τον **DeclVisitor** και τον **TypeVisitor**

DeclVisitor

Ο πρώτος visitor γεμίζει ένα `LinkedHashMap<String, ClassInfo>` το οποίο περιέχει όλα τα declara ons.
Τα fields και ο constructor της κλάσης του DeclVisitor είναι τα παρακάτω

```
public LinkedHashMap<String,ClassInfo> classDeclarations;
    public String className;
    public String methodName;

    public DeclVisitor(){
        this.classDeclarations = new LinkedHashMap<String,ClassInfo>();
        className = null;
        methodName = null;

    }
```

Το classDeclara ons είναι το map που ανέφερα απο πάνω. Το className και το methodName, είναι τα ονόματα της κλάσης και της μεθόδου στις οποίες βρίσκεται

κάθε στιγμή ο visitor. Αν δεν βρίσκεται σε κάποια κλάση ή/και μέθοδο, τότε οι τιμές των μεταβλητών είναι null.
Πριν εισέλθει κάτι καινούργιο στο Map, πάντα γίνεται check, αν υπάρχει ήδη και αν υπάρχει, γίνεται throw excep on, με το αντίστοιχο μήνυμα. Αν πχ στην ίδια κλάση εμφανιστούν δύο κλάσεις με το ίδιο όνομα, όταν η δεύτερη μέθοδος, πάει να μπει στο Map, θα γίνει έλεγχος, θα φανεί ότι υπάρχει ήδη μέθοδος με αυτό το όνομα και θα τερματίσει το πρόγραμμα.

TypeVisitor

Ο δεύτερος visitor λαμβάνει το `LinkedHashMap<String, ClassInfo> classDeclarations` που φτιάχνεται στον προηγούμενο Visitor και το χρησιμοποιεί για να γίνουν όλα τα υπόλοιπα checks.
Τα fields και ο constructor της κλάσης του TypeVisitor είναι τα παρακάτω:

```
public LinkedHashMap<String,ClassInfo> classDeclarations;
public ArrayList<VarClass> argList;
public String className;
public String methodName;

public TypeVisitor(LinkedHashMap<String, ClassInfo> classDeclarations){
    this.classDeclarations = classDeclarations;
    this.className = null;
    this.methodName = null;
}
```

Το `argList` είναι η λίστα με την οποία γεμίζω τα args κάθε κλήσης μεθόδου για να γίνει το checking των τύπων που ανέφερα στην ενότητα VarClass.