ΧΡΗΣΤΟΣ ΠΑΝΑΓΙΩΤΗΣ ΠΑΛΑΜΙΔΑΣ
1115201800140

Αρχεία:

- SymbolTable.java
- MyFirstVisitor.java
- MySecondVisitor.java

SymbolTable.java:

Το αρχείο αποτελεί τον ορισμό του Symbol Table, το οποίο δημιουργεί ο “First Visitor” όπως θα αναλυθεί παρακάτω. Αποτελείται από 1 Linked Hash Map που αφορά τη δήλωση των μεταβλητών στη main, στις κλάσεις αλλα και στις μεθόδους των κλάσεων (varDecl), από 1 Linked Hash Map που αφορά τη δήλωση των μεθόδων (methodDecl) (έχει υλοποιηθεί ξεχωριστή δομή για αυτό, θα αναλυθεί παρακάτω), από 1 Linked Hash Map που αφορά τις κλάσεις και τους γονείς των κλάσεων (classOrder) καθώς και ενα String στο οποίο αποθηκεύεται το όνομα της Main κλάσης (MainName). Ο λόγος που επέλεξα Linked Hash Map είναι γιατί μπορούσα να βάλω ένα Key που ορίζω εγώ και είναι μοναδικό κάθε φορά (αν δεν ήταν θα υπάρξει error) και έχει ένα Value που κρατά πληροφορία για αυτό.

Το varDecl αποθηκεύει πληροφορία ως εξής:

- (varName,className) = Type

Το varName είναι το όνομα της μεταβλητής, το className είναι η κλάση-scope μέσα στην οποία βρίσκεται η μεταβλητή και το Type είναι ο τύπος της. Αν υπάρξει διπλότυπο κλειδί τότε υπάρχει error.

Το methodDecl αποθηκεύει πληροφορία ως εξής:

- (methodName,className) = MethodSymTable
- MethodSymTable = Type , argList, varList (Κλάση)
- argList = (varName,methodName) = Type (Linked Hash Map)
- varList = (varName, methodName) = Type (Linked Hash Map)

Το MethodSymTable είναι μια κλάση που υλοποιήθηκε για να μπορεί να αποθηκεύσει όλη την απαραίτητη πληροφορία για κάθε μέθοδο. Αυτή είναι ο τύπος της (Type), οι μεταβλητές που ορίζονται μέσα στο argument list (argList) και οι μεταβλητές που ορίζονται μέσα στο σώμα της (varList). Το argList και VarList έχουν την ίδια δομή με το varDecl.

Το classOrder αποθηκεύει πληροφορία ως εξής:

- class = PossibleMotherClasses
- PossibleMotherClasses ::= Null
  SingleMotherClass
  MotherClass(-MoreMotherClasses)*

Το παραπάνω σημαίνει ότι μια κλάση μπορεί να μην είναι inherited (Null), μπορεί να κάνει extend μια κλάση που δεν είναι inherited (SingleMotherClass) ή μπορεί να κάνει extend μια κλάση που με τη σειρά της κάνει επίσης extend κάποια άλλη κ.ο.κ. Για την τελευταία περίπτωση, όλη η ακολουθία των extended κλάσεων αποθηκεύεται σε 1 String με delimiter την παύλα (-).

MyFirstVisitor.java:
Ο First Visitor ουσιαστικά δημιουργεί και αρχικοποιεί όλο το Symbol Table. Κάνει type checking, ελέγχει αν υπάρχουν variable & class redefinitions. Σε οποιοδήποτε error, “πετάει” exception για typecheck error.

MySecondVisitor.java:
Ο Second Visitor κάνει το semantic analysis. Κατά την έναρξη του βέβαια (στη visit της MainClass) καλείται μια συνάρτηση η οποία κάνει ένα τελικό typecheck με σκοπό να πιάσει προβλήματα όπως ότι δεν υπάρχει ο τύπος κάποιας μεταβλητής, size ή type missmatch μεταξύ shadow συναρτήσεων κ.α.
Κατά το semantic analysis, αν υπάρχει κάποιο error “πετάει” exception για semantic error.\

Σημειώσεις:

- Κυρίως για προσωπική βοήθεια στο debugging, όταν υπάρχει κάποιο error και κάνω throw new Exception, συνήθως εμφανίζω και ένα αναγνωριστικό μήνυμα για το είδος του προβλήματος.
- Στην αρχή, ολοκλήρωσα να γράφω τον κώδικα με το σκεπτικό ότι δε γίνεται να υπάρχει το εξης:
    - class A
    - class B extends A
    - class C extends B

  καθώς νόμιζα ότι αυτό δεν είναι single inheritance. Τελικά το έφτιαξα και έτρεξα αρκετά test files και όλα δουλεύουν σωστά. Ενδέχεται όμως να έχω παραλείψει κάτι καθώς είναι κώδικας που πρόσθεσα στο τέλος.
- Έχω τροποποιήσει ελάχιστα το makefile, ώστε στο “make all” να κάνει ‘clean’ πριν κάνει το compile καθώς αντιμετώπισα κάποια προβλήματα.
- Στο Second Visitor, στη visit της MainClass (γραμμή 41) υπάρχει η εντολή “symbolTable.prinstST()“ η οποία εκτυπώνει το πως διαμόρφωσε ο First Visitor το Symbol Table. Μπορείτε να την κάνετε comment out ώστε να μην εμφανίζει τίποτα.
- Θα συμπεριλάβω ένα φάκελο με όλα τα αρχεία που έκανα το testing.

How to run:

- make all
- java Main *filename*.java