Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

www.di.uoa.gr

HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

Άσκηση 2

Μεταγλωττιστές

Εαρινό Εξάμηνο 2019-2020

Παντελιάδη Ευφροσύνη

- Αριθμός Μητρώου: 1115201400301
- Email: sdi1400301@di.uoa.gr

Εισαγωγή

Στα πλαίσια του δεύτερου project κληθήκαμε να υλοποιήσουμε ένα μεταγλωτιστή για τη γλώσσα *MiniJava*, που αποτελεί ένα υποσύνολο της *Java*. Για τον σκοπό αυτό μας δόθηκαν η γραμματική της γλώσσας, τα εργαλεία *Java Tree Builder* και *JavaCC*.

Η υλοποίηση της εργασίας οργανώνεται στα παρακάτω αρχεία:

- Main.java,
- SymbolTable.java,
- Info.java
- ClassInfo.java,
- MethodInfo.java,
- FieldInfo.java
- StatementInfo.java,
- ClassChecker.java,
- MethodChecker.java και
- StatementChecker.java

Επιπλέον έχει δημιουργηθεί ένα *Makefile*, ώστε να διευκολυνθεί η διαδικασία της εκτέλεσης.

Δομές Δεδομένων

SymbolTable:

Στο αρχείο *SymbolTable.java* βρίσκεται η δομή που υλοποιεί το Symbol Table. Τα πιο βασικά στοιχεία που συναντάμε σε αυτήν είναι η λίστα `List<String> classes`, που περιέχει τα ονόματα των κλάσεων, και το Hash Map `HashMap<String, ClassInfo> classMap`, το οποίο χρησιμοποιώντας το όνομα μιας κλάσης μας οδηγεί στην οντότητα *ClassInfo* που την αναπαριστά.

Κάθε φορά που “επισκεπτόμαστε” τη δήλωση κάποιας κλάσης ελέγχουμε αν το όνομά της χρησιμοποιείται ήδη. Αν όχι, προσθέτουμε το όνομά της στη λίστα με τα ονόματα των κλάσεων, δημιουργούμε ένα νέο αντικείμενο της κλάσης *ClassInfo* και δημιουργούμε μια συσχέτιση για εκείνη στο HashMap.

Επιπλέον εδώ υπάρχει και ο κώδικας που υπολογίζει τα offsets για τα πεδία και τους δείκτες.

Info:

Η δομή αυτή αποτελεί τη base class για όλες τις δομές που αναπαριστούν κλάσεις, μεθόδους και statements. Περιέχει μόνο δύο πεδία: ένα όνομα και το offset.

ClassInfo:

Η συγκεκριμένη δομή εντοπίζεται στο αρχείο *ClassInfo.java* και αποτελεί εκείνη που χρησιμοποιείται για την αναπαράσταση μιας κλάσης. Τα πιο βασικά πεδία που περιέχει είναι η λίστα `List<FieldInfo> fields`, που περιέχει όλα τα πεδία της κλάσης, η λίστα `List<String> methods`, που περιέχει τα ονόματα των μεθόδων της, καθώς και το Hash Map `HashMap<String, MethodInfo> methodMap`, που συσχετίζει το όνομα μιας μεθόδου της με την οντότητα *Method Info* που την αναπαριστά. Επιπλέον περιέχει και μια αναφορά `ClassInfo parent`, που δείχνει στη γονική της κλάση (εφόσον έχει).

MethodInfo:

Η δομή που αναπαριστά μια μέθοδο βρίσκεται στο αρχείο *MethodInfo.java*. Τα βασικότερα στοιχεία που περιέχει είναι η λίστα `List<FieldInfo> variables`, που περιέχει τις τοπικές μεταβλητές της μεθόδου, η λίστα `List<FieldInfo> arguments`, η οποία αποτελεί υποσύνολο της προηγούμενης και περιέχει όλες εκείνες τις μεταβλητές που είναι ορίσματα, και η αναφορά `ClassInfo owner`, που δείχνει στην κλάση που περιέχει τη συγκεκριμένη μέθοδο.

FieldInfo:

Στο αρχείο *FieldInfo.java* βρίσκεται η δομή που αναπαριστά τα πεδία και γενικότερα τις μεταβλητές. Αποτελείται μονάχα από τον τύπο δεδομένων της μεταβλητής (`String type`) και μια μεταβλητή που δείχνει αν έχει αρχικοποιηθεί η τιμή της (`boolean initialized`).

StatementInfo:

Η δομή που βρίσκεται στο αρχείο *StatementInfo.java* είναι κυρίως βοηθητική για τον έλεγχο του σημασιολογικού ελέγχου και ουσιαστικά δεν αντιπροσωπεύει ένα statement.

Visitors

Οι visitors εντοπίζονται στα αρχεία *ClassChecker.java*, *MethodChecker.java* και *StatementChecker.java*. Το όνομα καθενός από αυτά φανερώνει το επίπεδο στο οποίο γίνεται ο έλεγχος (κλάσεων, μεθόδων και statements, αντίστοιχα).

ClassChecker:

Ο συγκεκριμένος visitor αναλαμβάνει τον έλεγχο σε επίπεδο κλάσεων. Αυτό σημαίνει πως κάθε φορά που δηλώνεται μια κλάση ελέγχει αν το όνομά της χρησιμοποιείται ήδη από κάποια άλλη. Μόνο στην περίπτωση που απάντηση είναι αρνητική συνεχίζει τον έλεγχο, ο οποίος θα γίνει για τα πεδία και της μεθόδους της κλάσης.

Όσον αφορά τις μεθόδους ο *ClassChecker*, αποθηκεύει και ελέγχει μονάχα τις ονομασίες τους.

MethodChecker:

Πρόκειται για τον visitor που πραγματοποιεί τον έλεγχο σε επίπεδο μεθόδων. Ο κώδικας που τον υλοποιεί βρίσκεται στο αρχείο *MethodChecker.java*. Ελέγχει τα ορίσματα και τις τοπικές μεταβλητές των μεθόδων, την περίπτωση πολυμορφισμού και επιπλέον αποθηκεύει τα απαραίτητα στοιχεία στις διαθέσιμες δομές.

StatementChecker:

Αποτελεί τον visitor που υλοποιεί το κυριότερο κομμάτι του σημασιολογικού ελέγχου. Μερικά παραδείγματα της λειτουργίας είναι ο έλεγχος του αριθμού και του είδους των ορισμάτων σε μια μέθοδο, ο έλεγχος συμβατότητας των στοιχείων σε ένα statement (π.χ. σε ένα assignment statement βγάζει error αν σε μία μεταβλητή int προσπαθήσουμε να αναθέσουμε μια τιμή boolean) κ.ο.κ.

Για τον έλεγχο του initialization αξιοποιεί τη βοηθητική κλάση *StatementInfo*. Συγκεκριμένα, έπειτα από παραδείγματα που έγιναν σε πραγματικό compiler, θεωρούμε όλα τα πεδία των κλάσεων και τα ορίσματα μεθόδων ως initialized, ενώ τις τοπικές μεταβλητές των μεθόδων ως uninitialized.

Αποτελέσματα:

Η εκτέλεση του προγράμματος θα τερματίσει είτε στο πρώτο σφάλμα που θα συναντήσει είτε στο τέλος, αφού δε θα έχει εντοπίσει κανένα σφάλμα. Στην πρώτη περίπτωση, θα εμφανιστεί στο τερματικό ένα μήνυμα που ενημερώνει το χρήστη για το σφάλμα το οποίο βρέθηκε. Στη δεύτερη περίπτωση, εκτυπώνεται πληροφορία για τις κλάσεις και τις συναρτήσεις που αποτελούν το πρόγραμμα, μαζί με τιμές για τα offsets που κληθήκαμε να υπολογίσουμε.

(*Σημείωση: Για συναρτήσεις που δε χρειάζεται να υπολογίσουμε το offset, γιατί η δήλωσή τους έχει προηγηθεί σε γονική κλάση, η τιμή του offset εκτυπώνεται ως -1.)