



ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS



Μεταγλωττιστές

2024 – 2025

Β' Φάση

3120009 Ανδρίτσος Γεώργιος

3190209 Φατσέα Ανθήπη

3210188 Σταμούλος Χρήστος

3210191 Σχοινάκη Μαρία

Ανάλυση Σημασιολογικών Ελέγχων

Χρησιμοποιήσαμε 2 διαφορετικούς visitor.

- Τον tableFillVisitor.java(1ος visitor)
- Visitor.java(2ος visitor).

Χρησιμοποιήσαμε και δύο πίνακες συμβόλων.

- methodTable - Για τις συναρτήσεις μας
- variableTable - Για τις μεταβλητές μας.

Στον 1ο visitor κάνουμε το γέμισμα του methodTable καθώς η σειρά δήλωσης της . Στον 2ο visitor κάνουμε όλους τους ελ

1. Περίπτωση ελέγχων χρήση μη δηλωμένης μεταβλητής

Χρήση μη δηλωμένης μεταβλητής.

Ο παραπάνω έλεγχος πραγματοποιείται στον 2ο Visitor (Visitor.java).

Η κατασκευή του ελέγχου αυτού βασίζεται στα Ins και outs των παρακάτω:

AssignPlusStatement, AStatFunctionOrStatement, AAssignMinusStatement,

AAssignMultStatement, AAssignDivStatement, AAssignEqStatement.

Μέσα στις παραπάνω μεθόδους οι οποίες κληρονομούνται από τον depth first adapter, αλλάζουν κατάσταση σε boolean μεταβλητές οι οποίες αποφασίζουν για το αν θα πρέπει να γίνει έλεγχος της μεταβλητής που χρησιμοποιείται την παρούσα στιγμή. Ο έλεγχος

επιτυγχάνεται στην caseAIdentifier η οποία σε συνεργασία με τις παραπάνω συναντήσεις αναζητά και ελέγχει την μεταβλητή στον πίνακα συμβόλων.

Ο πίνακας συμβόλων των μεταβλητών γεμίζει από την caseAAssignment η οποία καθορίζει πότε η μεταβλητή πρέπει να αποθηκευτεί στον πίνακα.

Ακολουθεί το παρακάτω παράδειγμα

```
1  def add(x,y):  
2  |  return x + y  
3  print k
```

```
*** Commencing Checks! ***  
*** No errors found in 1st Visit. Proceeding . . . ***  
!!! Error in line:3: Variable 'k' used before declaration.  
*** All visitors completed. ***  
*** Found 1 errors on visitor2 ***
```

Το πρόγραμμα εμφανισε μήνυμα λάθους διότι το κ δεν έχει δηλωθεί

2. Κλήση μη δηλωμένης συνάρτησης

Περιγραφή προβλήματος:

Όταν καλείται μια συνάρτηση που να μην έχει δηλωθεί.

Τρόπος επίλυσης:

Ο συγκεκριμένος έλεγχος γίνεται στον 2ο visitor (Visitor). Συγκεκριμένα κάνουμε Override την `caseAFunctionCall(AFunctionCall node)` και αναζητούμε την συνάρτηση στον `methodTable` που είχε γεμίσει στον 1ο visitor (`tableFillVisitor`).

```
def add(x,y=2):  
    return x + y  
  
print add(1,1,1)
```

```
*** Commencing Checks! ***  
  
*** No errors found in 1st Visit. Proceeding . . . ***  
  
!!!! Error in line 3: Function Call add doesn't match any defined function  
Check function definition add in line 1  
  
*** All visitors completed. ***  
  
*** Found 1 errors on visitor2 ***
```

3. Λάθος ορισμός ορισμάτων σε κλήση συνάρτησης

Περιγραφή προβλήματος:

Αυτό έχει άμεση σχέση με το προηγούμενο θέμα. Ουσιαστικά το να καλείται κάποια συνάρτηση που έχει δηλωθεί αλλά δεν έχουν περαστεί σωστά τα ορίσματά της.

Τρόπος επίλυσης:

Αντίστοιχα ελέγχεται στον 2ο visitor (Visitor) στην `caseAFunctionCall(AFunctionCall node)` παράλληλα με την αναζήτηση της δηλωμένης συνάρτησης. Αφού βρούμε τις πιθανές συναρτήσεις βάσει του ονόματος ελέγχουμε να δούμε και τον πλήθος των μεταβλητών.

```
def add(x,y=2):  
    return x + y  
  
print add(1)
```

```
*** Commencing Checks! ***  
  
*** No errors found in 1st Visit. Proceeding . . . ***  
  
*** All visitors completed. ***  
  
*** No errors were found ***
```

4. Χρήση μεταβλητής διαφορετικού τύπου

Περιγραφή προβλήματος:

Όταν γίνεται χρήση διαφορετικών τύπων δεδομένων (π.χ. αριθμός με συμβολοσειρά) σε αριθμητικές πράξεις, προκαλείται σφάλμα τύπου.

Τρόπος επίλυσης:

Για την ανίχνευση αυτού του προβλήματος, εφαρμόζεται έλεγχος τύπων δεδομένων κατά την ανάλυση των αριθμητικών εκφράσεων στις μεθόδους **caseAAdditionExpression**, **caseASubtractionExpression**. Αν εντοπιστεί προσπάθεια χρήσης μη συμβατών τύπων δεδομένων, παράγεται σφάλμα και ενημερώνεται ο χρήστης για το σφάλμα τύπου που προέκυψε.

Παράδειγμα Χρήσης:

```
1 def add(x,y):
2     return x + y
3 k="hello world"
4 print add(2,k)
```

```
*** Commencing Checks! ***
```

```
*** No errors found in 1st Visit. Proceeding . . . ***
```

```
!!! Error in line 4: Mismatched types in addition expression. Left side is Number and right side is String
```

```
*** All visitors completed. ***
```

```
*** Found 1 errors on visitor2 ***
```

5. Αριθμητικές πράξεις στις οποίες χρησιμοποιείται ως τελεστές το: None

Περιγραφή προβλήματος:

Το None δεν μπορεί να χρησιμοποιηθεί σε αριθμητικές πράξεις.

Τρόπος επίλυσης:

Διατήρηση πληροφοριών για τις αριθμητικές εκφράσεις μέσω Stack.

Χρησιμοποιείται μια στοίβα (*Stack<String> operatorStack*) για την παρακολούθηση των τελεστών και των τελεστών μέσα σε αριθμητικές εκφράσεις. Όταν το None εντοπίζεται μέσα σε μια αριθμητική έκφραση, ελέγχεται αν η στοίβα περιέχει κάποιο τελεστή και εμφανίζεται το κατάλληλο μήνυμα σφάλματος.

Έλεγχος μέσω της μεθόδου caseANoneValue.

Στη μέθοδο αυτή ανιχνεύεται η παρουσία του None μέσα στο AST και αναλύεται το πλαίσιο χρήσης του. Αν η χρήση γίνεται μέσα σε αριθμητική έκφραση, εκτυπώνεται μήνυμα σφάλματος.

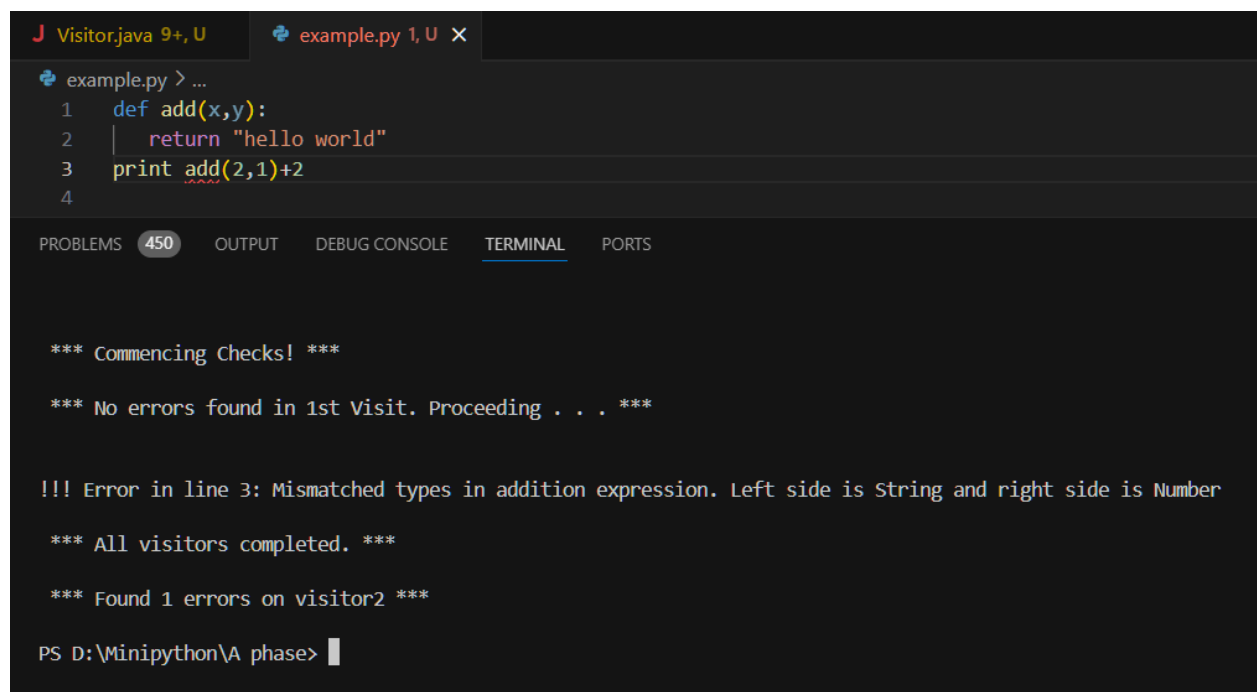
Παράδειγμα Χρήσης:

```
1  x = None
2  y = 1 ** None
```

6. Περίπτωση λάθους τρόπου χρήσης συνάρτησης.

Συνηθισμένο λάθος στη χρήση μεθόδων προκύπτει όταν η επιστρεφόμενη τιμή μιας συνάρτησης δεν είναι του αναμενόμενου τύπου, οδηγώντας σε σφάλματα κατά την

εκτέλεση. Για παράδειγμα, αν μια συνάρτηση υποτίθεται ότι επιστρέφει αριθμητική τιμή αλλά επιστρέφει μια συμβολοσειρά, τότε η χρήση της σε αριθμητικές πράξεις θα προκαλέσει σφάλμα τύπου. Στο παρακάτω παράδειγμα:



```
J Visitor.java 9+, U  example.py 1, U X
example.py > ...
1  def add(x,y):
2      return "hello world"
3  print add(2,1)+2
4
PROBLEMS 450  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

*** Commencing Checks! ***

*** No errors found in 1st Visit. Proceeding . . . ***

!!! Error in line 3: Mismatched types in addition expression. Left side is String and right side is Number

*** All visitors completed. ***

*** Found 1 errors on visitor2 ***

PS D:\Minipython\A phase> 
```

Η `caseAFunctionCall` είναι υπεύθυνη για τον έλεγχο της ορθότητας μιας κλήσης συνάρτησης στη γλώσσα `miniPython`. Συγκεκριμένα, ο έλεγχος ξεκινά με την ανάκτηση της αντίστοιχης δήλωσης της συνάρτησης από τον πίνακα `methodTable`, χρησιμοποιώντας το `matchedKey`. Στη συνέχεια, δημιουργείται ένας χάρτης (`HashMap<String, String> defArgsCallArgs`) που αντιστοιχίζει τα ορίσματα της συνάρτησης στις τιμές που περνιούνται κατά την κλήση. Λαμβάνεται η λίστα των ορισμάτων, που έχουν δηλωθεί στη συνάρτηση (`defArgs`), και διατηρείται μια ξεχωριστή λίστα με τα ονόματά τους (`defArgsList`). Παράλληλα, ελέγχονται αν υπάρχουν

προκαθορισμένες τιμές για τα ορίσματα και αποθηκεύονται στον χάρτη `defaultValues`, προσδιορίζοντας τον τύπο τους (`Number`, `String` ή `None`). Στη συνέχεια, αν υπάρχουν επιπλέον ορίσματα με προκαθορισμένες τιμές, αυτά προστίθενται στον ίδιο χάρτη.

Αν η λίστα των ορισμάτων κλήσης (`arglist`) δεν είναι κενή, τότε γίνεται αντιστοίχιση των ορισμάτων της συνάρτησης με τα πραγματικά ορίσματα που δόθηκαν κατά την κλήση. Χρησιμοποιείται ένας `iterator` για να διασχίσει παράλληλα τις λίστες των δηλωμένων και των δοθέντων ορισμάτων, και κάθε όρισμα της κλήσης αναλύεται ώστε να προσδιοριστεί ο τύπος του μέσω της μεθόδου `determineExpressionType`. Ο χάρτης `defArgsCallArgs` ενημερώνεται με τους τύπους αυτών των τιμών και αποθηκεύεται στο αντικείμενο `function`. Τέλος, η συνάρτηση προστίθεται στη στοίβα `stackFunctions`, ώστε να μπορεί να αναλυθεί περαιτέρω η επιστρεφόμενη τιμή της και να διασφαλιστεί ότι η χρήση της είναι ορθή.

7. Επανάληψη δήλωσης συνάρτησης με τον ίδιο αριθμό ορισμάτων

Επανάληψη δήλωσης συνάντησης με τον ίδιο αριθμό ορισμάτων.

Οι παραπάνω έλεγχοι πραγματοποιούνται στον `1o Visitor` (`tableFillvisitor`) μέσω της συνάρτησης `caseAFunction`. Η μέθοδος ψάχνει στον πίνακα συμβόλων των συναρτήσεων αν υπάρχει ξανά συνάρτηση με το ίδιο όνομα. Αν την εντοπίσει τότε ξεκινάει τους ελέγχους μεταξύ της μεθόδου που εισήλθε σαν όρισμα και αυτής που έκανε εξαγωγή από τον πίνακα συμβόλων.

Το όνομα συνάρτησης στον πίνακα συμβόλων είναι κωδικοποιημένο με τέτοιο τρόπο ώστε να περιέχει το πλήθος των non default παραμέτρων + το όνομα της συνάρτησης.

Για παράδειγμα έστω η συνάρτηση fib με 3 non default παραμέτρους στον πίνακα συμβόλων θα αποθηκευτεί ως 3fib. Η αναζήτηση και η αποθήκευση στον πίνακα συμβόλων των συναρτήσεων γίνεται με τον παραπάνω τρόπο.

Η αναζήτηση υλοποιείται στην isOnMethodTable. Στο σημείο αυτό το πρόγραμμα ελέγχει την συνάρτηση που λαμβάνεται σαν όρισμα και δημιουργεί την αντίστοιχη εκδοχή τής με την παρούσα κωδικοποίηση και την αναζητά στον πίνακα συμβόλων.

Ακολουθεί το παράδειγμα:

```
Visitor.java 9+, U  example.py U X
example.py > ...
1  def add(x,y):
2      print("ok")
3
4  def add(x,y,z=1):
5      print("ok")
6

PROBLEMS 449 OUTPUT DEBUG CONSOLE TERMINAL PORTS

*** Commencing Checks! ***

=====FAIL=====

!!! Error in Line 4:  Function add is already defined
=====

*** 1 errors found in 1st Visit. Exiting . . .

PS D:\Minipython\A phase> 
```

Ο compiler εμφανίζει error γιατί και οι 2 συναρτήσεις έχουν την δυνατότητα να κληθούν με τον ίδιο τρόπο πχ add(1,2).