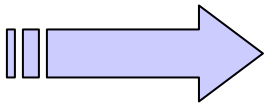
Διάσχιση AST δένδρου

Από τη γραμματική στις τάξεις (I)

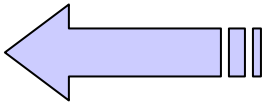
- Για κάθε τερματικό και μη-τερματικό σύμβολο της γραμματικής, ο SableCC φτιάχνει την αντίστοιχη Java τάξη στον κατάλογο <package_name>/node.
- Tokens

```
boolean = 'boolean';
assign = '=' ;
identifier = letter(letter|digit|'_')*;
number = digit+;
```



TBoolean.java
TAssign.java
TIdentifier.java
TNumber.java

extends **Token.java**



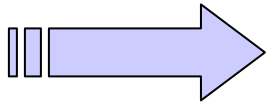
Χρήσιμες Μέθοδοι:

- toString()
- getLine()
- getPos()

Από τη γραμματική στις τάξεις (II)

- Κανόνες AST (για κάθε alternative)

```
exp = {and}     [l]:exp [r]:exp
    | {plus}    [l]:exp [r]:exp
    | ...
    | {number} number
    | {id} identifier;
```



AAndExp.java
APlusExp.java
ANumberExp.java
AIdExp.java

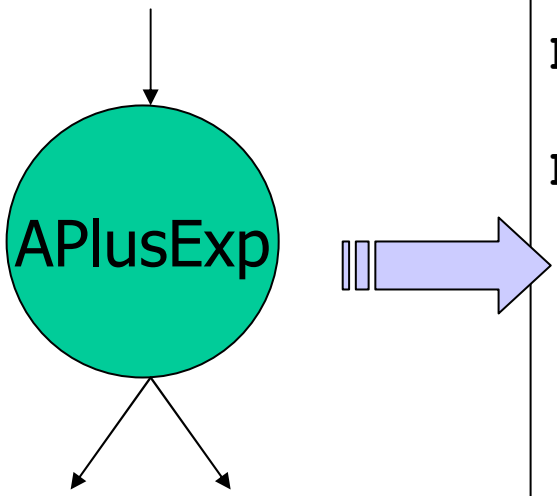
extends **Pexp[.java]**

`program = main_class;`  AProgram[.java] extends PProgram[.java]

- Μέθοδοι Τάξεων, π.χ. APlusExp.java:
  - public PExp getL() { … }
  - public PExp getR() { … }
- AIdExp.java:
  - public TIdentifier getIdentifier() { … }

Επισκέπτες κόμβων AST δένδρου

- Στον κατάλογο <package_name>/analysis
  - DepthFirstAdapter extends AnalysisAdapter




```
...
public void inAPlusExp (APlusExp node){...}

public void outAPlusExp (APlusExp node){...}

public void caseAPlusExp (APlusExp node){
  inAPlusExp (node);
  if(node.getL() != null){
    node.getL().apply(this);
  }
  if(node.getR() != null){
    node.getR().apply(this);
  }
  outAPlusExp (node);
}
...
```

Κόμβοι AST δένδρου

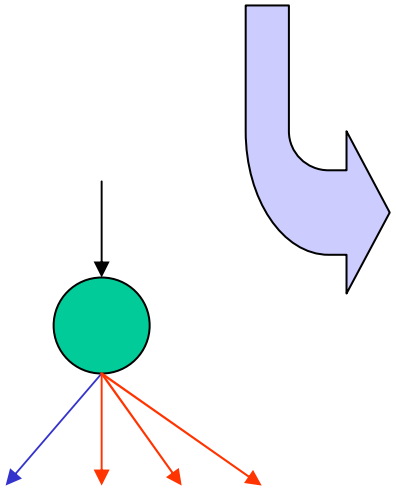
- Αρχικός κόμβος: Start

```
program = main_class;
```

```
public void caseStart(Start node) {
  inStart(node);
  node.getPProgram().apply(this);
  node.getEOF().apply(this);
  outStart(node);
}
```

- Διάσχιση Λίστας

```
main_class  = identifier statement* ;
```



```
public void caseAMainClass(AMainClass node) {
  inAMainClass(node);
  if(node.getIdentifier() != null) {
    node.getIdentifier().apply(this);
  }
  {
    Object temp[] = node.getStatement().toArray();
    for(int i = 0; i < temp.length; i++) {
      ((PStatement) temp[i]).apply(this);
    }
  }
  outAMainClass(node);
}
```

Κατασκευή δικού σας επισκέπτη

- YourVisitor.java
  - extends DepthFirstAdapter
- Μπορείτε να κάνετε override όποιες μεθόδους του DepthFirstAdapter θέλετε:
  - στις μεθόδους **inXxxx** και **outXxxx** γράφετε κατευθείαν τον κώδικά σας
  - στις μεθόδους **caseXxxx** θα πρέπει να αντιγράψετε τον κώδικά τους από το αρχείο DepthFirstAdapter.java και να προσθέσετε όπου θέλετε τον κώδικά σας, μιας και στις μεθόδους αυτές περιλαμβάνεται και ο κώδικας διάσχισης των υποδένδρων
- Οι μέθοδοι caseXxxx γίνονται override όταν χρειάζεται να κάνουμε κάτι ανάμεσα στις επισκέψεις των υποδένδρων του κόμβου

Compiler.java

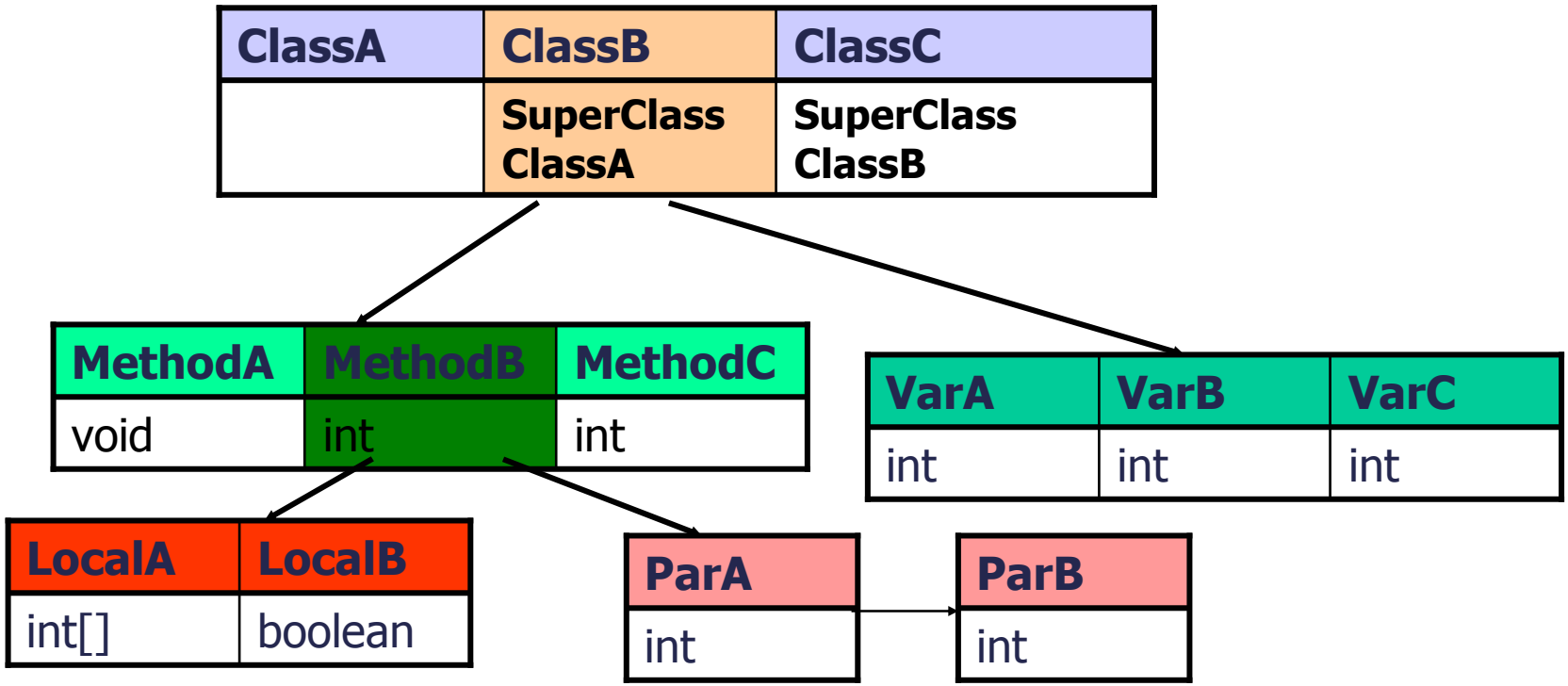
```
import java.io.*;
import package.lexer.Lexer;
import package.parser.Parser;
import package.node.Start;

public class Parser {
  public static void main(String[] args)  {
    try    {
      Parser parser = new Parser(
                      new Lexer(
                      new PushbackReader(
                      new FileReader(args[0].toString()),1024)));
      Start ast = parser.parse();

      ast.apply(new YourVisitor());

    } catch (Exception e) {
      System.err.println(e);
    }
  }
}
```

Πίνακας Συμβόλων

Java Hashtables

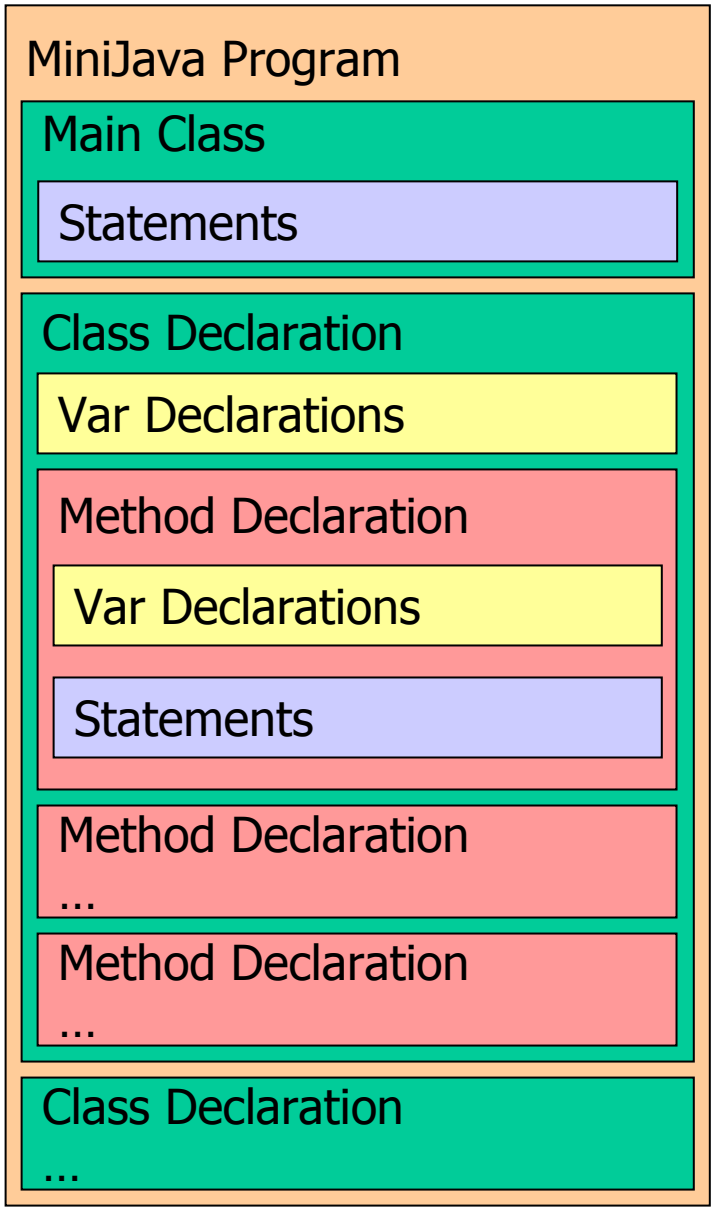
- Πακέτο java.util.Hashtable
- Δομή τύπου «λεξικού»
- Παρέχουν αντιστοίχιση μεταξύ κλειδιών και τιμών
- Μέθοδοι:
  - Κατασκευαστής: `new Hashtable();`
  - `boolean containsKey (Object key)`
  - `boolean containsValue (Object value)`
  - `Object get (Object key)`
  - `Object put (Object key, Object value)`
  - `Object remove (Object key)`
  - `boolean isEmpty()`
  - `void clear()`
  - `void putAll(Hashtable t)`
  - `Object clone()`
  - `Enumeration keys()`
  - `Collection values()`

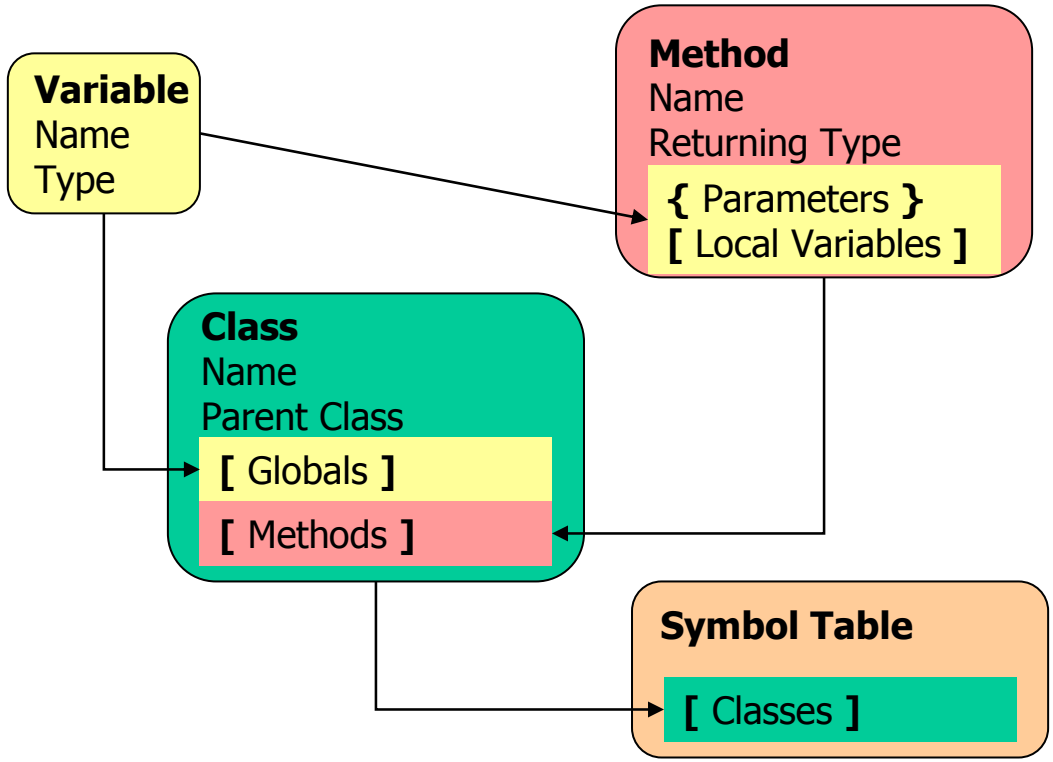
Κατασκευή Πίνακα Συμβόλων



- Ξεχωριστή δομή δεδομένων για τις τάξεις, τις μεθόδους και τις μεταβλητές
- Πρέπει να φυλάσσονται όλες οι απαραίτητες πληροφορίες για κάθε στοιχείο (όνομα, τύπος, κλπ...)

Δομή Πίνακα Συμβόλων





Συμβουλές:

- Σε κάθε τάξη ορίστε τις κατάλληλες μεθόδους.
- Κάποιες από αυτές μπορούν να πραγματοποιούν μέρος των ελέγχων.

Γέμισμα Πίνακα Συμβόλων

- Χρειάζεται μια διάσχιση του δένδρου, για να γεμίσετε τον Πίνακα Συμβόλων
- Ταυτόχρονα, μπορείτε να κάνετε κάποιους ελέγχους αναφορικά με τα στοιχεία που εισάγετε, π.χ.:
  - δήλωση μεταβλητής με υπάρχον όνομα
  - δήλωση μεθόδου με υπάρχον όνομα
  - δήλωση τάξης με υπάρχον όνομα
  - …
- Ίσως χρειαστεί να τηρείτε και να ενημερώνετε και κάποιες global μεταβλητές για να ξέρετε μέσα σε ποια τάξη και ποια μέθοδο είστε κάθε στιγμή

Παράδειγμα γεμίσματος Πίνακα Συμβόλων

```
program      = main_class simple_class* ;

main_class   = [clname]:identifier [arg]:identifier statement ;

simple_class  = identifier statement ;

...
```

```
public class MyAdapter extends DepthFirstAdapter {
    private Vector symtable;

    MyAdapter(Vector table) { this.symtable = table;}

    public void inAMainClass(AMainClass node) {
        String ClassName = node.getClname().toString().trim();
        symtable.add(ClassName);
    }

    public void inASimpleClass(ASimpleClass node) {
        String ClassName = node.getIdentifier().toString().trim();

        int line = ((TIdentifier) node.getIdentifier()).getLine();
        int pos  = ((TIdentifier) node.getIdentifier()).getPos();

        if (symtable.contains(ClassName)) {
            System.out.println("[" + line + "," + pos +"]:  Class " + ClassName +
                               " is already defined");
        } else {
              symtable.add(ClassName);
          }
    }
}
```

in και out Hashtables

- Στην τάξη minijava/analysis/AnalysisAdapter.java, έχουν οριστεί 2 global hashtables που μπορούν να χρησιμοποιηθούν για αποθήκευση προσωρινών πληροφοριών (π.χ. επιστρεφόμενοι τύποι πράξεων, …)
- Παρεχόμενες μέθοδοι:
  - setOut(Object key, Object value) ⇒ out.put (key, value)
  - getOut(Object key) ⇒ out.get (key)
  - setIn(Object key, Object value) ⇒ in.put (key, value)
  - getIn(Object key) ⇒ in.get (key)
  - setOut(Object key, null) ⇒ out.remove (key)
  - setIn(Object key, null) ⇒ in.remove (key)
- Ως κλειδιά μπορούν να χρησιμοποιηθούν οι κόμβοι του AST δένδρου

Δείτε επίσης σχετικό word document στο e-class [set_get_out.doc]

Παράδειγμα χρήσης out Hashtable

```
exp = {plus}    [l]:exp [r]:exp
    | {number} number
    | {id} identifier;
```

```
public void outAVardecl(AVardecl node) {
 String variable = node.getIdentifier().toString().trim();
 int line = ((TIdentifier) node.getIdentifier()).getLine();
 int pos  = ((TIdentifier) node.getIdentifier()).getPos();
 PType type;
 if (symtable.containsKey(variable))
  System.out.println("[" + line + ", " + pos + "]: Variable " + variable + " defined!");
 else {
  type = node.getType(); // String type = ((ACustomType) node.getType()).getIdentifier().toString().trim();
  symtable.put(variable,type); }
}
public void outANumberExp(ANumberExp node) {
 setOut(node, new AIntegerType()); // setOut(node, "int");
}
public void outAIdExp(AIdExp node) {
 PType tmp = (PType) symtable.get(node.getIdentifier().toString().trim()); // String tmp = ...
 setOut(node, tmp);
}
public void outAPlusExp(APlusExp node) {
 PType left  = (PType) getOut(node.getL()); // String left = getOut(node.getL());
 PType right = (PType) getOut(node.getR()); // String right = getOut(node.getR());
 if ( !(left instanceof AIntegerType) || !(right instanceof AIntegerType) )
  System.out.println("Addition (+) should be used with integer variables!" );
 else {
  setOut(node, getOut(node.getL()) );
  setOut(node.getL(), null);
  setOut(node.getR(), null); }
}
```

Σημασιολογική Ανάλυση

Κατάλογος σημασιολογικών λαθών (I)

Στην **1η σάρωση** του AST (γέμισμα Πίνακα Συμβόλων), μπορούν να γίνουν οι εξής έλεγχοι (παράλληλα ή στο τέλος):

1. Επανάληψη δήλωσης
   - Τάξης
   - Μεθόδου (στην ίδια τάξη)
   - Μεταβλητής (στην ίδια τάξη/μέθοδο)
   - Παραμέτρου (στην ίδια μέθοδο) [***μαζί με μεταβλητές***]
2. Ανύπαρκτη υπερτάξη
3. Κυκλική κληρονομικότητα τάξεων (***απλή περίπτωση***)
4. Κυκλική κληρονομικότητα τάξεων (***δύσκολη περίπτωση***)

Κατάλογος σημασιολογικών λαθών (II)

Στη **2η σάρωση** του AST θα γίνουν οι εξής έλεγχοι:

5. Μη δηλωμένος τύπος μεταβλητής ή επιστροφής μεθόδου
6. Μη δηλωμένη μεταβλητή / ανύπαρκτη μέθοδος
7. Ασυμβατότητα ***πρωταρχικών τύπων*** σε πράξεις και εκχωρήσεις
8. Λανθασμένοι τύποι σε εκφράσεις με πίνακες
9. Λανθασμένοι τύποι εκφράσεων σε while, if, do-while, for και println
10. Δημιουργία αντικειμένου ανύπαρκτης τάξης
11. Λάθος αριθμός παραμέτρων στις κλήσεις μεθόδων
12. Λανθασμένοι ***πρωταρχικοί τύποι*** μεταβλητών/παραμέτρων στις κλήσεις μεθόδων
13. Λανθασμένο override μεθόδου – ***πρωταρχικοί τύποι*** (δεν επιτρέπεται overload)
14. Λανθασμένο override μεθόδου – ***μη πρωταρχικοί τύποι*** (δεν επιτρέπεται overload)
15. Ασυμβατότητα ***μη πρωταρχικών τύπων*** σε εκχωρήσεις και πέρασμα παραμέτρων
16. Πιθανότητα μη αρχικοποίησης μεταβλητής

Σημασιολογικά λάθη

- Τα μηνύματα λάθους πρέπει να είναι:
  - κατανοητά για τον εκάστοτε προγραμματιστή
  - διευκρινιστικά (γραμμή/θέση που εντοπίστηκε το λάθος, όπου είναι δυνατό)
- Όταν μια σάρωση του δένδρου τελειώνει και έχουν βρεθεί λάθη, δεν πρέπει να προχωράει η μεταγλώττιση στην επόμενη σάρωση
- Συνολικός αριθμός λαθών στο τέλος της μεταγλώττισης
- Αν βρείτε και άλλου τύπου σημασιολογικά λάθη που για να ελέγξετε, μπορείτε να τα ενσωματώσετε…

- **Όσα περισσότερα, τόσο καλύτερα**