TP Compilation - HEPIAL

Ottavio Buonomo & Jean-Daniel Kuenzi

Introduction

- TP du cours de Techniques de compilation
- Création d’un langage de programmation
- Groupe de 2
- Utilisation de Jflex, CUP et Jasmin
    - Avec Java

- Qu’est ce que HEPIAL ?
- Analyse lexicale
- Analyse syntaxique
- Création de la TDS
- Analyse sémantique
- Génération du code
- Démo
- Conclusion

Qu’est ce que HEPIAL ?

- Langage de programmation
- Composé de
  - Variables / Constantes
    - Types
      - Entier
      - Booléens
  - Opérations
    - Arithmétique
    - Binaire
    - Unaire
  - Boucles
  - Lecture / Écriture sur la console

```
programme demo1
entier n;
entier result;
debutprg
    ecrire "Nombre a mettre au carré: ";
    lire n;
    result = n*n;
    ecrire "Résultat:";
    ecrire result;
finprg
```

Compilation de HEPIAL

Analyse lexicale

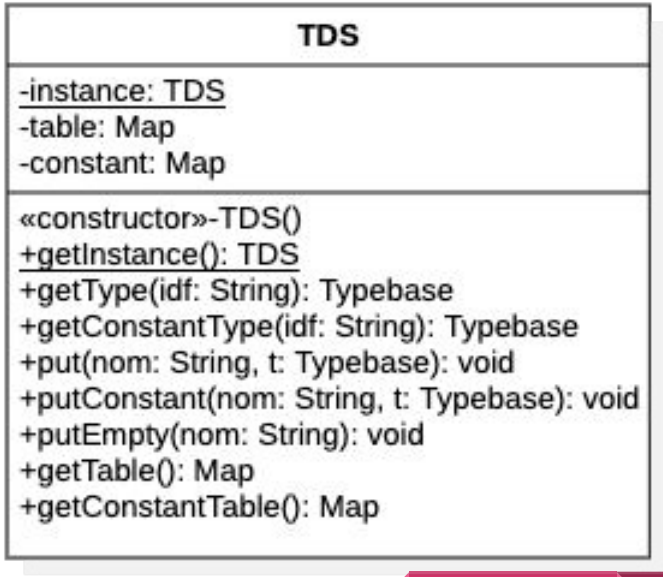
- Utilisation de Jflex + CUP
- Permet de reconnaître les mots du langage
- Transcrire les mots en symboles
- Fichier “.flex” fourni

Analyse syntaxique

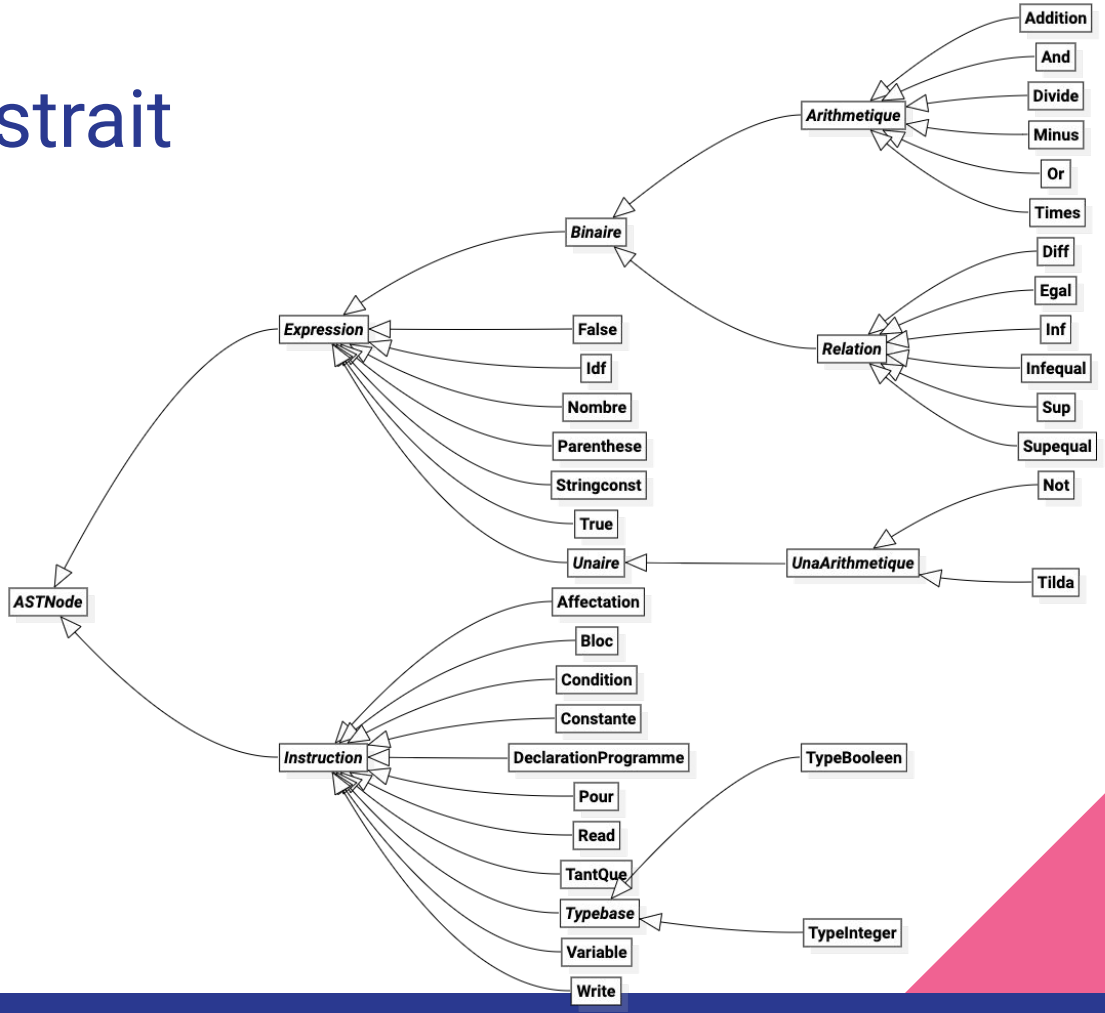
- Utilisation de CUP
- Permet de vérifier que la syntaxe du code soit correcte
  - Ex. : si x < 10 alors ✔️
  - Ex. : si x < alors 10 ❌
- Stockage des déclarations dans la TDS
- Complétion du fichier “hepial.cup”
- Création de l’arbre abstrait

Création de la table des symboles

- Utilisation CUP + Java
- Contient les variables déclarées
- HashMap<String, Typebase>
- 2 tables
  - Variables
  - Constantes
- Singleton
- Partagé au sein du compilateur



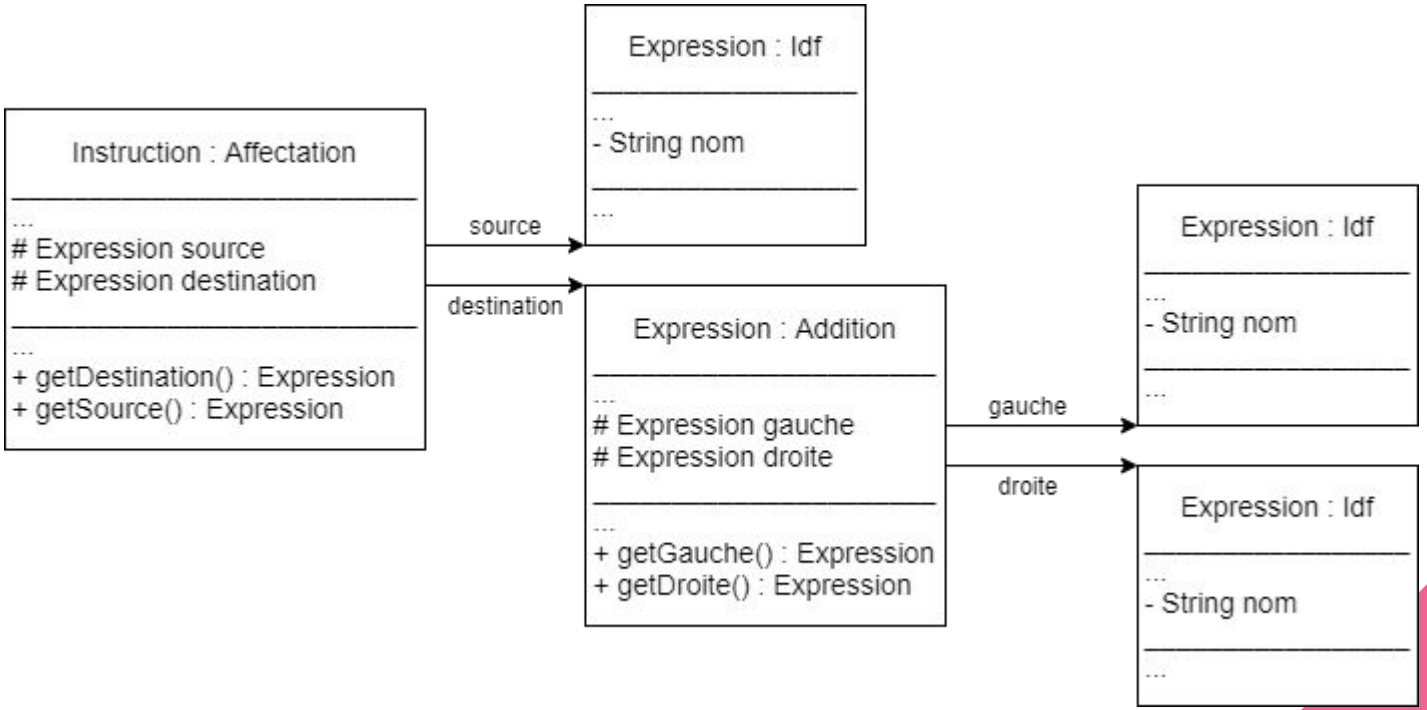
Arbre abstrait



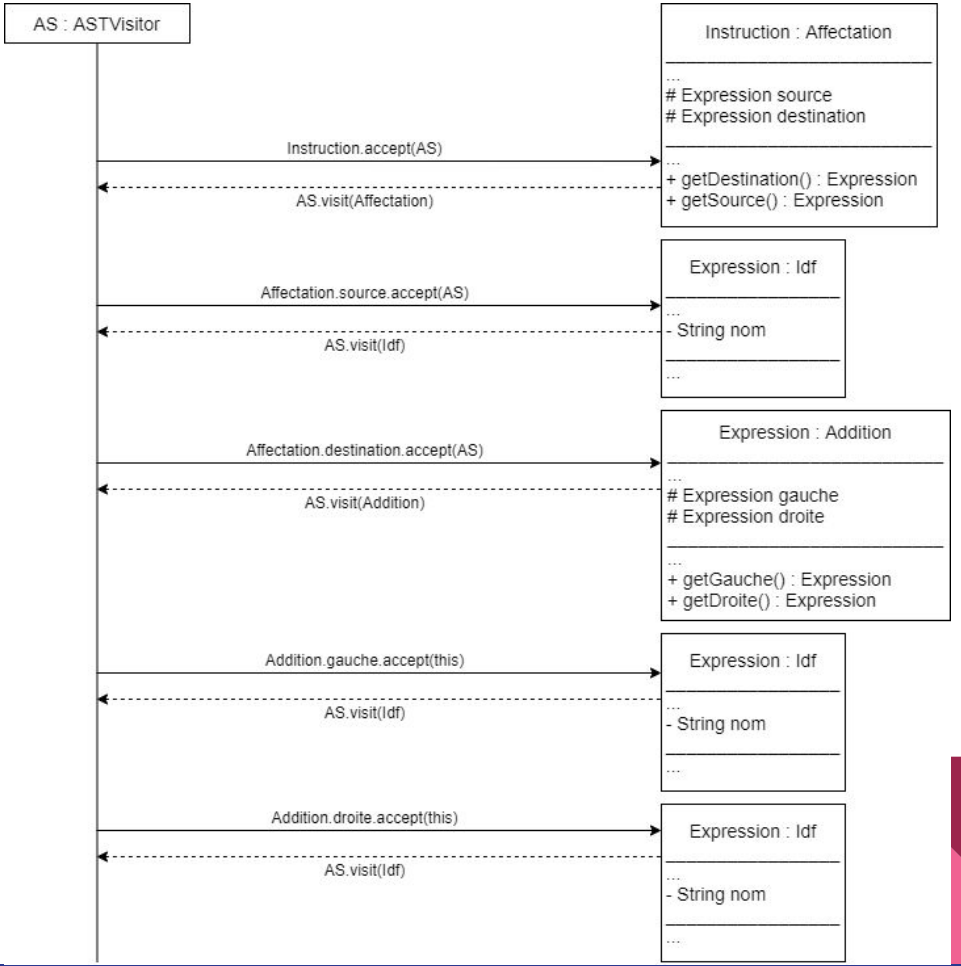
Analyse sémantique

- Parcours des noeuds de l’arbre (Visiteur)
- Vérification de types en fonction de l’expression
- Vérification de déclaration
- Utilisation de la table des symboles
- AnalyseSemantique.java

Analyse sémantique



Analyse sémantique



Génération du byte-code

- Assembleur Jasmin
  - Génération byte-code pour la JVM
- Système de pile (stack)
- Parcours les noeuds de l’arbre abstrait
- Écriture du code Jasmin dans un fichier “monJasmin.j”
- Passage du code assembleur à la JVM
- Lancement du programme

```
class public demo
.super java/lang/Object
.method public static main([Ljava/lang/String;)V
.limit stack 20000
.limit locals 100
.var 0 is i I
.var 1 is b I
.var 2 is c I
.var 3 is times I
.var 4 is d Z
.var 5 is MACONSTANTE I
ldc 1000
istore 5
var 6 is MACONSTANTE2 Z
```

Génération du byte-code

- Opération avec un label
- Label unique
- Permet de se déplacer dans le code
- Un compteur unique
- Méthode getCpt()

Démo

Conclusion

- Créer un langage de programmation
- Appliquer les étapes de la compilation
  - Analyse lexicale
  - Analyse syntaxique
  - Analyse sémantique
- Application des notions du cours de TCP
  - Jflex / CUP
- Difficultés
  - Prise en main compliquée

Questions ?