# Compilateur – Projet SQL

**Documentation du projet**

# 

# Dates

Date du document : 08 janvier 2016  
Début projet: 13 novembre 2015  
Remis du projet et présentation: 15 janvier 2016

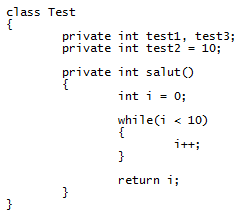
# À propos

Etudiants : Di Stasio Leonardo & Assunçao Jeshon  
Professeur : Tièche François

# Introduction

En programmation, il arrive fréquemment que l’on désire créer des classes afin de faire un lien entre notre application et une base de données. Cependant, il est nécessaire de créer « à la main » chaque table ainsi que ses attributs dans la base de données, ce qui peut évidemment induire des erreurs.

Projet SQL est donc un compilateur qui va analyser une classe (C++, JAVA, etc.) afin de générer le code SQL à exécuter sur la base de données. Il sera donc toujours possible de modifier le code SQL avant de l’exécuter afin qu’il corresponde bien aux critères de l’utilisateur (taille d’un varchar, etc.).



# 

Figure 1.2 : Code SQL généré pour cette classe

# 

Figure 1. : Exemple d'une classe Java

# Conception général

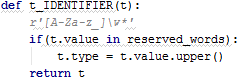
# Analyse lexical

**Lexèmes définis** :

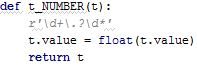
* ‘IDENTIFIER’
* ‘NUMBER’

Le premier lexème est réservé pour détecter les variables, nom de classe et de fonction, le deuxième pour détecter les nombres.

**Traitement du lexème « IDENTIFIER » :**



**Traitement du lexème « NUMBER » :**



**Mots réservés :**

* Private, Public, Protected
* class
* int, Integer, float, Float, string, String, char, Character, double, Double
* while, for
* extends, implements
* print, void, return, break, continue
* if, switch, else

On réserve donc tous les mots qui ont une signification (en Java).

**Caractères spéciaux :**

* ();={},!<>"\'/\*+-

Nous réservons ici tous les caractères pouvant servir à une opération mathématique, une fin de ligne, une inversion (point d’exclamation) ainsi que les caractères « englobant » (parenthèses, accolades et chevrons).

**Caractères à ignorer :**

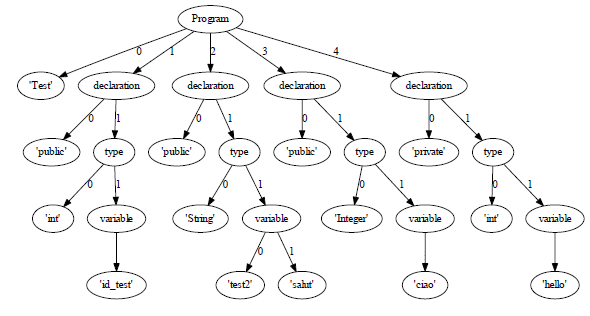
* « Espace »
* \t

# C:\HEARC\Compilateur\Projet\Documentation\PRN-NE-CA2-1409-CAN_0032_001-rotated-page-001.jpgAnalyse syntaxique

# Analyse sémantique

Pour l’analyse sémantique il nous fallait « coudre » l’arbre abstrait. Pour cela il nous suffisait de récupérer les variables de la class Java ainsi que le nom de la class.

L’arbre obtenu donnait quelque chose de ce genre :



Le nœud « Program » à pour fils le nom de classe (dans l’exemple ci-dessus il s’agit de « Test ») ainsi que tous les nœuds « Declaration » qui identifie des variables.

Ces « Declaration » sont spécifiées par la visibilité (public, protected ou private) ainsi que d’un nœud « Type » qui précise le type de la variable.

Comme expliqué le nœud « Type » contient le type de variable qui a été déclaré (int, string, etc.) ainsi qu’un nœud « Variable ».

Dans « Variable » il est possible de retrouver un ou plusieurs noms de variables. S’il y en a plusieurs c’est qu’elles ont été déclarées de cette façon :

**public** String test2, salut;

On a donc la possibilité de concaténer les déclarations de variables avec des virgules.

Ceci est tout pour la partie sémantique. Il faut juste retenir que les variables, leur type, leur visibilité et le nom de classe sont cousus dans l’arbre sémantique.

# Partie arrière

# Conclusion

# Références