# Compilateur – Projet SQL

**Documentation du projet**

# 

# Dates

Date du document : 11 janvier 2016  
Début projet: 13 novembre 2015  
Remis du projet : 15 janvier 2016

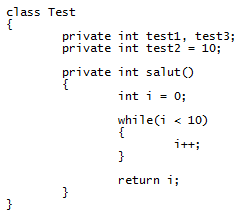
# À propos

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

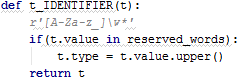
# Analyse lexical

**Lexèmes définis** :

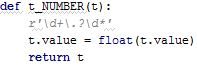
* ‘IDENTIFIER’
* ‘NUMBER’

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, float, String, char, 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).

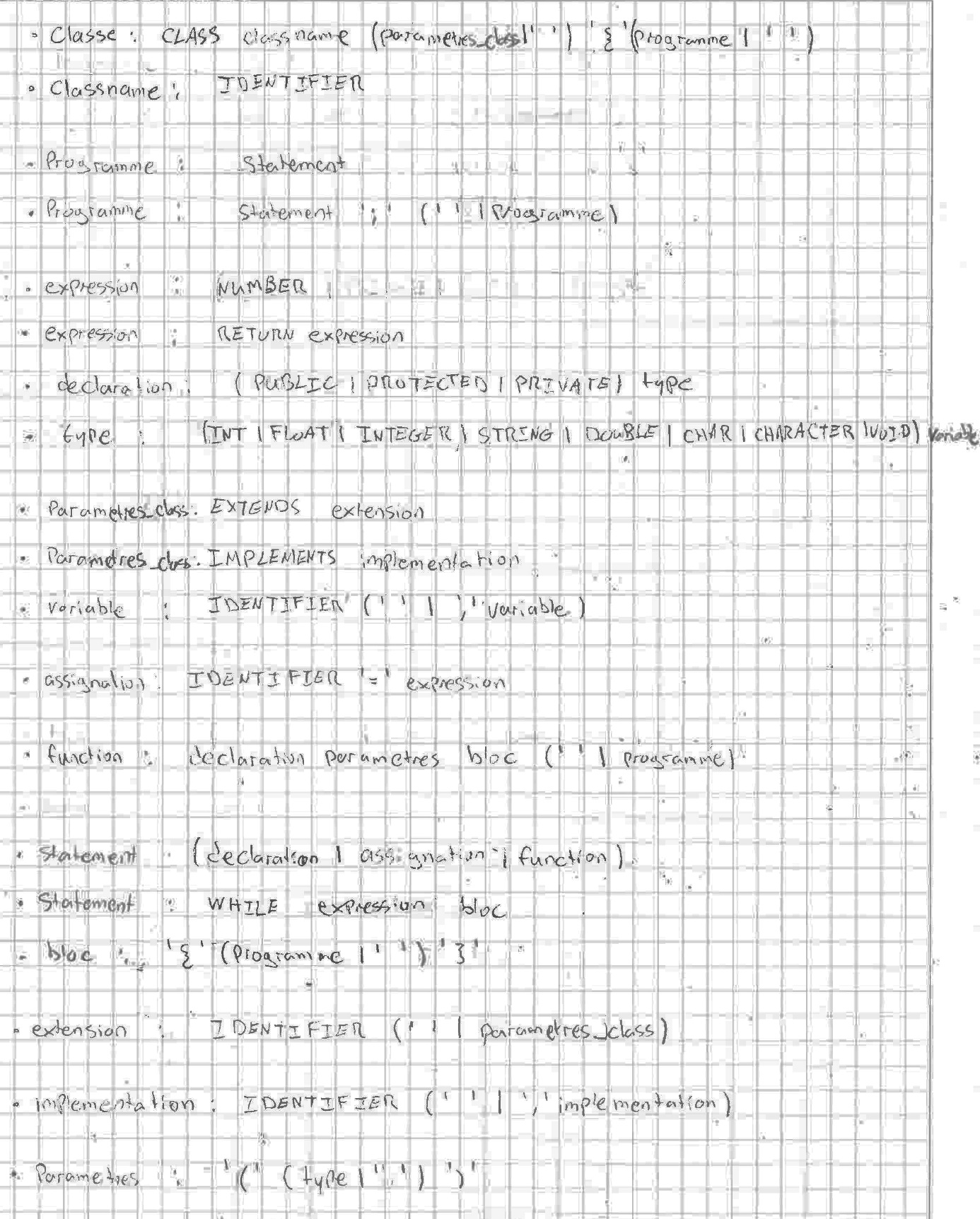
**Caractères à ignorer :**

* « Espace »
* \t

# Analyse syntaxique

Quasiment toute la syntaxe de Java est gérée dans notre compilateur. C’est-à-dire que celui-ci est en mesure de reconnaître une boucle, une déclaration de variable, un extends, etc. Cependant, uniquement les informations nécessaires à la génération de notre code SQL sont récupérées pour former l’arbre, c’est-à-dire : variable et nom de classe.

Voici notre grammaire :



# Analyse sémantique

# Partie arrière

A partir de l’arbre généré par l’analyse sémantique, nous pouvons désormais générer le code SQL. Pour cela, nous traitons 6 types de nœuds (uniquement les nœuds nécessaires) qui sont :

* « ProgramNode » pour le nom du fichier sql ainsi que le nom de la table à créer
* « DelcarationNode » pour toute déclaration
* « TypeNode » pour récupérer le type d’une variable
* « VariableNode » pour construire le code SQL à partir du nom de la variable, du type et de sa taille
* « TokenNode » pour récupérer un token quelconque
* « AssignNode » pour gérer l’assignation de variables

# Conclusion

# Références