# Rapport d’exécution

## Option 1 - % des attributs publics, % des attributs protégés et % des attributs privés dans l’enceinte d’une classe.

L’analyseur traite la syntaxe de base de Java pour la déclaration d’attributs de classe.

Un attribut a un spécificateur d’accès, un type, un nom et une attribution de valeur.

L’analyseur couvre la grammaire suivante :

<attribut> ::= <access> <type> **lowercamel** “**=**” <valeur> “**;**”

<access> ::= {**public** | **private** | **protected**}

<type> ::= **lowercamel** | **uppercamel**

<valeur> ::= <newObject> | **lowercamel** | <number>

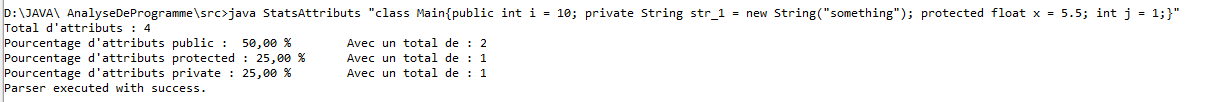
<newObject> ::= **new** **uppercamel** “**(**” <params> “**)**”

<params> ::= <valeur> {“**,**” <valeur>}

<number> ::= **reel** | **entier**

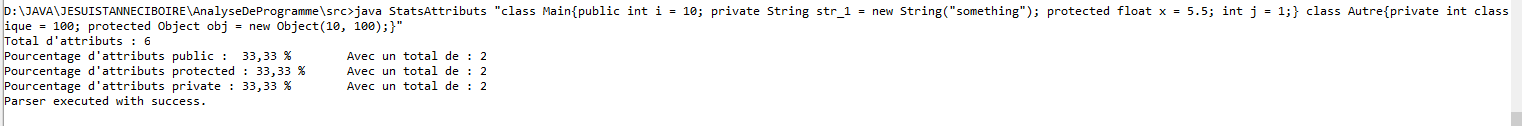
Ici nous avons un exemple d’exécution avec 4 attributs différents.

Note : un attribut sans spécificateur d’accès est considéré public.

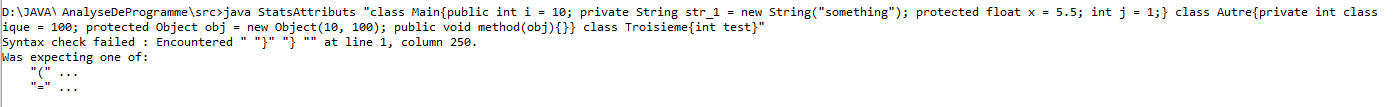


Ici nous avons un deuxième exemple d’exécution avec une deuxième classe.

Note : on ajoute deux attributs au calcul initial soit un privé et un protégé.

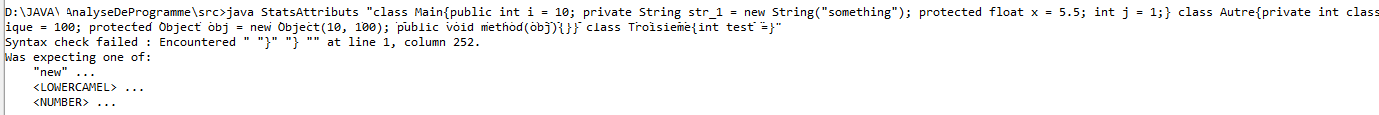


Ici nous avons un exemple d’erreur possible soit une mauvaise déclaration d’attribut. L’analyseur indique qu’il accepte une parenthèse (une déclaration de méthode) ou un égal (une déclaration d’attribut) et que la syntaxe actuelle est erronée.



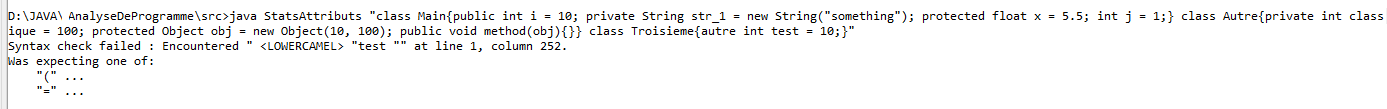
Si nous ajoutons un égal, le comportement attendu serait d’attendre une valeur.

Note : la grammaire indique qu’on attend un nombre, un nouvel objet ou un nom de variable.

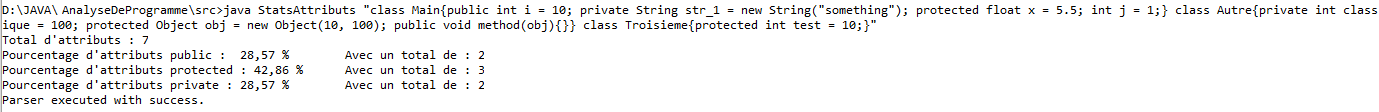


Si le spécificateur d’accès est erroné, l’analyseur lève une erreur.

Note : l’analyseur nous indique que test devrait être une parenthèse ou un égal parce que autre n’est pas un spécificateur existant.



Si nous corrigeons le spécificateur en erreur, nous avons un nouveau calcul d’attribut avec une analyse à succès.



## Option 2 -