## TP 2 : Analyseur syntaxique

- On va continuer développer notre analyseur syntaxique dans le projet « mips1Proj ». Pour cela :

### 1- Programmer toutes les procédures syntaxiques.

- On doit pouvoir gérer les erreurs syntaxiques :
    - Créer la classe « **ErreurSyntaxique »** qui hérite de la classe **ErreurCompilation**.

- On doit déclarer les codes des erreurs syntaxiques dans **CodesErr** :
    - Au code d'erreur **PROGRAM_ERR** est associé le message « Mot clé program attendu ! »
    - Au code d'erreur **ID_ERR** est associé le message « Identificateur attendu ! »
    - Au code d'erreur **PVIR_ERR** est associé le message « Symbole ; attendu ! »
    - …

- Créer une nouvelle classe « **Parser** » dans laquelle :
    - vous déclarez l'attribut scanner de type **Scanner**.
    - vous générez les getters et setters.
    - Vous définissez un constructeur avec un argument représentant le nom du fichier à analyser.
    - Vous définissez la méthode **testeAccept**.
    - Vous définissez les procédures syntaxiques : **program**, **block**, **consts**, **vars**, **insts**, **inst**, **affec**, **si**, **tantque**, **ecrire**, **lire**, **cond**, **expr**, **term**, **fact**.

### 2- Tester votre analyseur syntaxique.

- Pour tester l'analyseur, on définit une méthode **main** dans la classe « **Parser** ». Dans le main, on crée un objet de type Parser en pr´cisant le fichier contenant le programme à analyser, puis on initialise la table des mots clés, après on lit le premier caractère. Ensuite on appelle l'analyseur lexical, juste après on appelle l'axiome. Si après l'exécution de l'axiome, on atteint le EOF alors l'analyse syntaxique a réussi sinon on lance une erreur de syntaxe dont le code est EOF_ERR.