Master S&T - Mention Informatique
$1^{ère}$ année
2015 – 2016
Compilation

Projet de compilation

Interprétation et compilation en programmation objet

1 Objectifs

1. **Prototyper un compilateur et un interpréteur** afin de concrétiser les concepts des techniques de compilation : analyse lexicale, analyse syntaxique, syntaxe abstraite, contrôle de types, sémantique interprétative (Opérationnelle), compilation, traitement des erreurs ;
2. **Réaliser un travail d'équipe** pour comprendre et résoudre les problèmes d'organisation selon les méthodes de Génie Logiciel : spécification des interfaces entre les différents modules, composition des modules, définition des procédures de test, définition des procédures de recette, réalisation de revues de codes, rédaction de la documentation,... ;
3. **Développer une interface utilisateur** conviviale et cohérente pour l'ensemble de l'application ;
4. **Vérifier sur des exemples** que le compilateur MiniJaja $\longrightarrow$ JajaCode est correct vis à vis des sémantiques interprétatives des deux langages. Pour cette partie, vous devrez appliquer une stratégie de test qui permette de valider le mieux possible le produit.

2 Organisation

Un GROUPE de TRAVAIL est composé d'au moins quatre étudiants du même groupe TP. Le travail à réaliser est présenté sur le schéma 1.

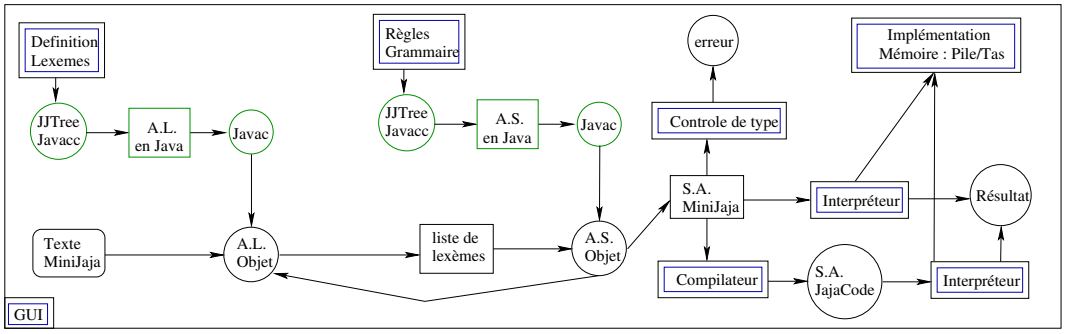

FIGURE 1 – Décomposition du Projet

La démonstration réalisée en fin de projet permettra de montrer que la figure 2 commute bien.

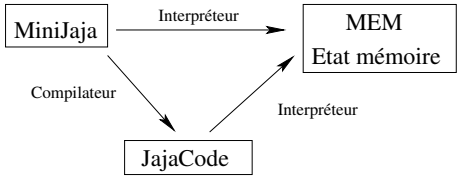

FIGURE 2 – Diagramme de concordance

3 Analyse syntaxique

Il s'agit de décrire l'ensemble des symboles lexicaux, l'ordre de priorité des opérateurs et la grammaire du langage sous forme BNF. Il existe plusieurs techniques qui permettent de générer des analyseurs syntaxiques à partir de la définition de tokens et de règles de grammaire. Les plus utilisés pour Java sont Javacc ou ANTLR. Vous écrirez ainsi un analyseur produisant un arbre de syntaxe abstraite dans un fichier ou directement en mémoire pour que le module de contrôle de types puisse le prendre en entrée.

4 Méthode de travail

Chaque groupe forme une équipe de développement. La première tâche de cette équipe doit être de répartir le travail entre ses différents membres ou sous-groupes. De manières à coordonner vos développements, nous vous conseillons de mettre en place une structure de centralisation de l'information :

- Soit sous la forme de points de synchronisation de l'ensemble de l'équipe (réunions de concertation souvent répétées) ;
- Soit sous le forme d'une personne (responsable de projet) qui assure la centralisation et la diffusion des informations ;
- Soit en combinant savamment les deux options précédentes (il existe de nombreux outils dédiés pour aider dans ces tâches).

D'autre part, nous vous conseillons aussi de bien définir les tâches de chacun, en évitant les recoupement et les lacunes. Pour arriver à ce résultat, vous disposez de la pré-réunion qui vous a permis de réaliser la fiche de projet, qui doit contenir :

- La liste des personnes du groupe ;
- L'énumération des taches (avec estimation de durée, des personnes intervenant) ;
- Organisation dans le temps (planning, diagramme de Gantt, point de synchronisation, itération, sous-ensemble fonctionnel ...) ;
- Les points sur lesquels vous estimez que vous aurez besoin de personnes ou de ressources ;
- Le standard de programmation qui sera adopté pour le développement, le langage et les outils ;
- Les moyens de communication.

Lors de la séance de TP qui suit, nous rencontrerons chaque groupe pour discuter des options choisies et de l'organisation des sous-groupes. Chaque sous-groupe organise son travail en cohérence avec les jalons du projet :

- Chacun analyse et réalise sa partie en accord avec les sous-groupe avec lesquels il interagit ;
- Chacun effectue des tests dits "de chemin d'exécution" ou tests unitaires pour valider sa partie. Ces tests s'effectuent de manière ascendante à partir des modules élémentaires puis par intégration progressive (tests d'intégration) avec une couverture minimale de 70% ;
- Chacun établi des tests dit "boite noire" à partir de la spécification (sans connaître sa réalisation) d'un module réalisé par un autre membre de son groupe ;
- Le responsable de projet organise des revues de réalisation. Pour cela, il procède par échanges des dossiers de réalisation entre les membres de son équipe. Celui qui reçoit un dossier doit en faire une revue critique et faire un rapport de revue destiné au réalisateur. Ce rapport ne doit pas corriger les erreurs, mais seulement les signaler. Ces rapport de revue devront figurer dans le rapport final ;
- Chacun devra utiliser des technique de programmation dîtes défensives.

Note : tous les jeux de test devront être explicités dans le rapport final. Le jeu de test recette sera utilisé pour la démonstration du produit final.

5 Conseils de réalisation

La réalisation des différents modules sera guidée par la structure des syntaxes abstraites des langages. Il est conseillé de définir une classe par sorte d'arbre de syntaxe abstraite. Chacune des classes définissant le langage contiendra, par exemple, pour le langage MiniJaja :

- Un opérateur de création d'un instance de la classe ;
- Une opération de contrôle de type ;
- Une opération d'interprétation ;

— Une opération de compilation ;
— Une opération d'affichage ;
— ...

Cette structuration permet de conserver la structuration modulaire de la spécification donnée dans le support de cours. Une opération sera alors l'implémentation d'une règle. Cette structuration donne une réalisation modulaire facilement modifiable en cas de modification du langage. La suppression d'une structure du langage ou la modification d'une structure existante induira de définir ou de modifier la classe correspondante.

Enfin un choix judicieux dans la hiérarchie des classes permet d'utiliser l'héritage afin que des classes partagent des attributs et des opérations pour éviter la réécriture.

6 Rapport Final

Ce rapport sera constitué de deux document. Ils devront être rendus, en version papier au secrétariat et en version électronique sur la plateforme moodle, le **vendredi 11 décembre avant 12h00**.

Le premier exposera :
— Le sujet ;
— Les contraintes fonctionnelles (propriétés de l'interface homme-machine) que satisfait le logiciel ;
— La décomposition du logiciel et les structures de communication entre les différentes parties ;
— Pour chaque partie, les principaux choix de réalisation et en particulier le traitement des erreurs ;
— Les jeux de tests et leur justification pour chaque partie ;
— Les jeux de tests recette et leur justification ;
— Les résultats sur la batterie de tests fournie ;
— Le compte rendu du travail en équipe (rapports de revue, organisation, avantages, inconvénients...) ;
— Une conclusion.

Le second est intitulé *"Rapport de conduite de projet"*. Il décrit l'organisation du projet (taches, répartition des rôles, découpage en itération, ..), le suivi de projet (suivi entre le prévisionnel et le réalisé), l'utilisation des outils de suivi de projet (SonarQube, Jenkins, Nexus...) et une rétrospective de conduite de projet. Il sera évalué dans le cadre du module Génie Logiciel.

7 Cahier des charges

L'objectif du travail est la réalisation d'un environnement de développement MiniJaja/JajaCode. Pour cela, il faudra effectuer l'analyse syntaxique, l'interprétation et la compilation en JajaCode des programme MiniJaja et l'interprétation des programmes JajaCode obtenus comme résultat de la compilation.

7.1 Besoins fonctionnels

— On veut pouvoir compiler tous les programmes MiniJaja corrects et bien typés. Un programme est correct s'il se décompose d'après la syntaxe concrète décrite dans les notes de cours ;
— On veut pouvoir comparer les résultats de l'interprétations d'un programme MiniJaja et du programme JajaCode obtenu par compilation ;
— On veut pouvoir faire la compilation des programmes édités grâce à un éditeur de texte standard sous différents systèmes d'exploitation (Unix, Windows) ;
— On veut pouvoir observer le JajaCode résultat de la compilation ;
— Pouvoir observer l'interprétation pas à pas des programmes MiniJaja et JajaCode ;
— Pouvoir observer l'exécution d'un programme munis de points d'arrêt.

7.2 Besoins non fonctionnels

— La compilation et l'interpréteur du langage MiniJaja doivent respecter les règles décrites dans les notes de cours ;
— Les messages d'erreurs doivent être les plus clairs possibles. C'est à dire qu'ils doivent localiser le plus précisément possible le lieu de l'erreur relativement au texte source. Ils doivent donner un message le plus explicite possible. Par conséquent, les erreurs doivent être détectées le plus tôt possible ;
— La réalisation des analyseurs lexical et syntaxique doit utiliser Javacc (JJtree) ou équivalant ;

— Les structures de données mémoire la pile et le tas doivent être mises en œuvre le plus efficacement possible. La réalisation des opérations doit être basée sur l'accès direct. Une pile avec des entrées par hachage est la structure conseillée pour MEM et une système de ramasse miette pour le tas. L'utilisation des techniques de hachage permet de satisfaire la contrainte d'accès direct au mieux ;
— La réalisation du logiciel doit conserver la structure des règles des notes de cours. En utilisant, les classes (C++, Java) pour décrire la syntaxe abstraite, il est possible d'attacher à chaque classe d'arbre de syntaxe abstraite des opérations *interpréter, compiler, ...* qui réalisent les règles. La notion d'héritage doit permettre de partager les opérations identiques par le plus grand nombre de types d'arbres possibles. La hiérarchie des classes doit en tenir compte ;
— On doit pouvoir observer les états mémoire de fin d'exécution des programmes afin de les comparer ;
— L'ensemble du projet doit être réalisé dans un même environnement cohérent.

7.3 Rendu intermédiaire

A fin d'avoir un cadencement du travail conforme à une démarche de type Scrum. Vous devez définir un sous-ensemble fonctionnel de votre projet et faire la livraison intermédiaire (*"Release 1"*) associée. La date de cette livraison est fixée durant la semaine du **2 novembre** pendant votre séance de TP de compilation.

7.4 Soutenance

La soutenance se déroulera **jeudi 17 décembre**. C'est aussi la date de rendue de la version finale (*"Release 2"*).

Votre présentation se décompose de la façon suivante : 20 minutes où vous présentez votre projet en abordant les choix effectués (technologie, architecture, cadencement, implémentations...) et en faisant un bilan des points positifs et négatifs rencontrés. Chacun des membres du groupe devra intervenir. Vous aurez ensuite 15 minutes pour nous faire la démonstration de votre logiciel en nous présentant les différentes fonctionnalités implantées à l'aide de vos jeux de tests. Puis 15 minutes sont réservées pour les tests du jury et les questions. Cela nous donne une présentation d'une durée globale de 50 minutes.

Un vidéo projecteur est à votre disposition. Il est **très fortement** conseillé de vérifier le bon fonctionnement de celui avec votre matériel avant la soutenance et non juste au début.

Votre logiciel est à déposer sous moodle au plus tard après la soutenance.