Ecole Ingénieurs 2000 - Université de Marne-la-Vallée
Filière Informatique et Réseaux
2ème année

Documentation

******

IRCompiler

******

Compilateur d'un petit langage de redéfinition d'opérateurs.

Compte rendu de : **Tom MIETTE, Sébastien MOURET**

Fait le : **25 janvier 2008**

Sommaire

I. Introduction

IRCompiler est une petite application développée en Java qui est capable de compiler un petit langage de programmation personnalisé. Brièvement, ce langage est capable de manipuler les classes de l'API Java standard, d'effectuer les opérations basiques, de manipuler des boucles et des sauts conditionnels, ainsi que de redéfinir l'usage des opérateurs par défaut.

II. Ce que sait faire IRCompiler

Par rapport au sujet d'origine, notre compilateur est en mesure de gérer entièrement les niveaux 1 et 2, ainsi qu'une partie du niveau 3.

En effet, les deux types de boucles « foreach », l'utilisation et la concaténation des strings ainsi que l'appel des méthodes statiques « plus() », « minus() », etc. des objets sont tous gérés par le compilateur. Quelques fichiers de tests permettent de mettre en évidence ces points (référez-vous au paragraphe ci-dessous).

III. Ce que ne sait faire IRCompiler

Compte tenu de ce qui est déclaré dans la grammaire du language, il y quelques expressions que IRCompiler ne sait complètement gérer. Voici une liste de ces limites :

- la multiple affectation n'est pas permise "a = b = c". En effet, l'affectation doit se faire en une instruction "isolée" afin qu'elle soit interprétée. Ainsi, les utilisations, telles que « if a = b » ou « print a = b » lèveront des erreurs de parsing.

- dans les boucles « foreach », le compilateur vérifie bien que le type à itérer est Iterable. Cependant, il n'y a pas de vérification du cast entre le type déclaré et le type réel de l'objet Iterable pendant la phase sémantique de compilation. Cette vérification se fait simplement en bytecode (opération CHECKCAST) et lèvera une erreur qu'au moment de l'exécution du fichier binaire.

- le boxing/unboxing des types primitifs n'est pas géré.

- les morceaux de code « unreacheble » ne sont pas détectés dans les fonctions. De plus, le compilateur ne vérifie pas entièrement l'absence d'une instruction "return". IRCompiler se contente de détecter une instruction "return" dans le corps de la fonction (à n'importe quel endroit) pour valider cette vérification.

- la libération des registres n'est pas entièrement prise en compte. En effet, les registres sont alloués selon les différents contextes (fonction, et méthode main principale). Cependant, les registres alloués à une variable ne sont libérés qu'au moment où le contexte

courant est dépilé (c'est à dire à la fin de la fonction, ou à la fin du fichier). Ainsi, si l'on utilise plus d'une quarantaine de variables dans un même contexte, il n'y aura plus de registres disponibles et ceci entraînera une erreur d'allocation.

IV. Les tests

Le répertoire « test » du projet contient plusieurs fichiers exemples qui mettent chacun en évidence un point particulier du mécanisme de compilation. Référez-vous aux commentaires dans ces fichiers afin de voir ce qu'ils démontrent.

Pour lancer correctement ces tests, il faut les compiler avec le jar exécutable du projet comme le montre la commande suivante :

```
$ java - jar ${user_directory}/bin/IRCompiler.jar <input file> <output file> <print code>
```

Assurez-vous ensuite de lancer le fichier .class généré en vous plaçant dans le même répertoire où ce dernier se trouve afin que la méthode « main() » se lance correctement. Il faut noter également que ces fichiers utilisent des classes personnalisées prédéfinies (Point, Complexe, etc.).