Intégration des nombres complexes et des Unums en Java avec COJAC

PV de la séance du 9 juin 2020 (9h30 - 10h40) via Teams

Présents: Frédéric Bapst, Cédric Tâche

Ordre du jour

Les points suivants ont été abordés durant la séance:

1. Validation du PV du 2 juin 2021.
2. Cahier des charges v1.0
3. Rapport v0.2
4. Fonctionnement de COJAC
5. Planification

PV

1. Le PV du 2 juin était complet.

Cahier des charges

1. Certaines phrases peuvent prêter à confusion.
2. Certaines tâches peuvent être précisées.
3. D'autres objectifs secondaires peuvent être ajoutés:
   - Documentation de COJAC: rédiger de la documentation en anglais ou faire une vidéo.
   - Comparaison des approches wrappers et behaviours (qui sont décrites plus loin.)
   - Ajout d'un CI pour vérifier les tests et compiler le JAR.
4. D'autres améliorations à la base de code existante seraient aussi le bienvenu.
5. Un jalon pour la démonstration du premier prototype sera ajouté dans la planification.
6. Les démonstrations seront spécifiées avant la conception des fonctionnalités.

Intégration avec COJAC

1. COJAC est un agent JAVA qui modifie les classes lorsqu'elles sont chargées par la JVM comme indiqué sur la figure 1.
2. COJAC a deux modes de fonctionnement pour changer le comportent des nombres à virgule flottante. Soit le comportement des doubles peut être interprêté différemment (aussi appelé behaviour), soit les doubles peuvent être remplacés par des wrappers.
3. La réinterprétation des doubles peut se voir dans les classes *BehaviourClassVisitor* et *BehaviourMethodVisitor* ainsi que dans le paquet *models/behaviours*.

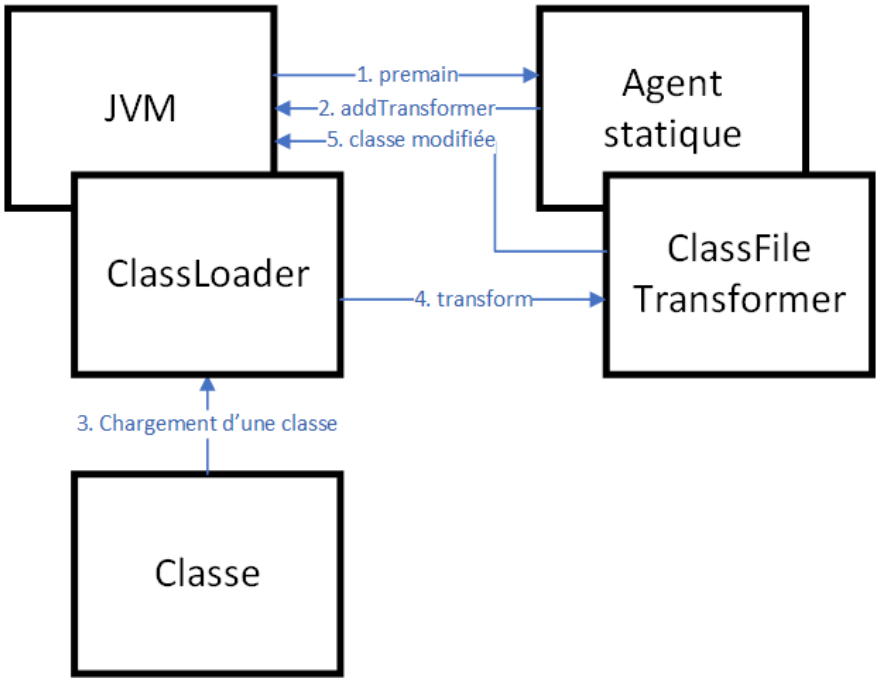

Figure 1: Principe d'un agent JAVA statique

4. Le remplacement des doubles par un wrapper peut se voir dans les classes *FloatReplaceClassVisitor* et *FloatReplaceMethodVisitor* ainsi que dans le paquet *models/wrappers*. Les wrappers *WrapperBasic*, *WrapperBigDecimal* et *WrapperInterval* sont de bons exemples.

5. Les wrappers sont un ensemble de classes hiérarchisées avec un wrapper racine commun.

6. COJAC a une option **-W** pour indiquer le type de wrapper à utiliser en remplacement des doubles. Elle permet de tester les wrappers avant d'implémenter l'option spécifique.

Erreur COJAC dans les démonstrations

1. L'erreur apparaissant pendant l'exécution des démonstrations avec une version de Java supérieure à 8 est due à une nouvelle instruction *invokedynamic* ajoutée dans Java 8 pour supporter les lambdas. Cette instruction a ensuite été utilisée pour d'autres usages dans les versions plus récentes et ces nouvelles utilisations ne sont actuellement pas supportées par COJAC.

Autres problèmes avec COJAC

1. M. Tâche a remarqué que du code dans COJAC ne respecte pas la documentation de la méthode *transform* qui est disponible au lien suivant: `https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/ClassFileTransformer.html`. La documentation dit que le tableau retourné doit soit être **null**, soit être **un nouveau tableau de bytes**.. A plusieurs endroits, COJAC retourne directement le tableau de bytes d'entrée. Un TODO sera rajouté dans le code, mais la modification ne sera pas forcément faite pendant ce projet.

2. Le code et l'architecture de COJAC sont très mal documentés. M. Bapst regardera si des anciens rapports d'étudiants contiennent plus d'informations.

Librairie de modification du Bytecode JAVA

1. La librairie utilisée pour modifier le Bytecode des classes dans le projet se nomme ASM. Cette librairie utilise abondamment le pattern Visitor.
2. Byte Buddy est une librairie de haut niveau pour générer et modifier des classes depuis un agent Java. Comme ce projet ne nécessitera pas de modifier le Bytecode directement car du code Java est déjà disponible, cette librairie ne sera sûrement pas utilisée. Cependant, c'est une bonne idée de le mentionner dans le rapport, car c'est potentiellement une alternative à ASM.

Planification

1. Le projet a un retard d'un jour par rapport à la planification dû à des problèmes avec COJAC et des incompréhensions sur son fonctionnement.

Décision

1. La nouvelle version du cahier des charges sera rendue cette semaine dès que possible.
2. Le patch de M. Bapst pour corriger l'exécution des démonstrations sera testé avec toutes les démonstrations. Si le résultat est concluant, le patch sera appliqué, sinon la compilation des démonstrations (et des applications utilisateurs) continueront à se faire avec Java 8.

Prochaines tâches

1. Finir le cahier des charges.
2. Finir l'analyse de COJAC.
3. Faire l'analyse des nombres complexes.
4. Rédiger et coder les spécifications des démonstrations.
5. Commencer la conception de l'intégration des nombres complexes.

Prochaines échéances

Responsable: M. Bapst

Description	**Date limite**
Chercher les rapports des anciens projets liés à COJAC	16.06.2021

Responsable: M. Tâche

Description	**Date limite**
Finir le cahier des charges	11.06.2021

Prochaine séance: Le mercredi 16 juin 2020 à 9h30