TP2 HMIN306
Analyse Statique

Pour ce TP, vous disposez d'un projet maven pour commencer. Il est disponible sur github: https://github.com/greenpanther93/AnalyseAvecSpoon. Il contient trois classes principales, *JDTMain*, qui vous permettra de réaliser l'exercice 1, *SpoonMain* qui vous permettra de réaliser l'exercice 2 et *GraphVizMain* qui vous permettra de réaliser l'exercice 3. Normalement, ce projet contient toutes les dépendances nécessaires pour compléter ce TP.

Exercice 1 : Découverte de l'AST avec JDT (15-30 min)

JDT, ou Java Development Tooling, est un outil d'Eclipse qui facilite la manipulation d'un code source JAVA. Lorsque vous utilisez Eclipse en tant que développeur, JDT facilite la manipulation de votre projet en analysant le projet et en proposant des raccourcis pour vous faciliter la vie (i.e. navigation entres vos classes, auto-complétion, visualisation du projet dans le workspace, etc.). **Consultez** les tutoriels suivants pour vous familiariser avec l'outil JDT:

- Parseur AST dans JDT : http://www.programcreek.com/2011/01/a-complete-standalone-example-of-astparser/
- Tutoriel AST : https://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST
- Le patron visiteur est un patron fortement utilisé pour parcourir l'ensemble d'un AST. Il repose sur le principe d'une méthode définie par l'utilisateur lorsqu'il rencontre un noeud spécifique. Rappel sur le patron : https://fr.wikipedia.org/wiki/Visiteur_(patron_de_conception
- (optionnel) AST View : http://www.eclipse.org/jdt/ui/astview/index.php

Appliquez le patron getter/setter sur la classe *classAsString* qui est inclus dans le projet github comme string.

Pour rappel, le patron getter/setter sert à faire passer l'accès à un attribut par une méthode qui permettra de vérifier les valeurs d'affectation ou de lecture. Utilisez le patron visiteur sur l'AST pour réaliser cette tâche.

Rendu: Lorsque vous rendez votre TP, veuillez inclure le projet de cet exercice avec le code implémenté dans la classe *JDTMain*.

Exercice 2 : Étude de l'outil Spoon et analyse statique de RestSuite (1h)

Spoon est une bibliothèque open source qui vous permet de transformer et d'analyser un code source Java. Spoon fournit un méta modèle Java complet et détaillé dans lequel tout

élément de programme (classes, méthodes, champs, instructions, expressions, etc.) est accessible à la fois pour la lecture et la modification. Spoon prend un code source en entrée et produit un code source transformé prêt à être compilé. Étudiez l'outil Spoon, disponible sur http://spoon.gforge.inria.fr/. Utilisez Spoon (en utilisant l'ensemble des outils disponibles avec Spoon : filtres, scanners, iterators, queries, templates, patterns, processors, etc...) sur RestSuite pour trouver les informations suivantes:

- Nombre de classes de l'application.
- Nombre de lignes de code de l'application.
- Nombre total de méthodes de l'application.
- Nombre total de packages de l'application.
- Nombre moyen de méthodes par classe.
- Nombre moyen de lignes de code par méthode.
- Nombre moyen d'attributs par classe.
- Les 10% des classes qui possèdent le plus grand nombre de méthodes.
- Les 10% des classes qui possèdent le plus grand nombre d'attributs.
- Les classes qui font partie en même temps des deux catégories précédentes.
- Les classes qui possèdent plus de X méthodes (la valeur de X est donnée).
- Les 10% des méthodes qui possèdent le plus grand nombre de lignes de code (par classe).
- Le nombre maximal de paramètres par rapport à toutes les méthodes de l'application.

Pour commencer, consultez le site principal et la section "Getting Started" et "Querying source code elements". La Javadoc est disponible et vous permettra de répondre à vos questions approfondies : http://spoon.gforge.inria.fr/mvnsites/spoon-core/apidocs/

Rendu: Comme l'exercice précédent, veuillez rendre le projet avec le code principal dans la classe *SpoonMain*, ainsi qu'un fichier .txt contenant vos réponses.

Exercice 3 : Construction du graphe d'appel d'une application (1h30min)

Construisez un graphe de dépendance à partir d'une analyse statique:

- En s'appuyant sur les résultats de l'exercice 2, construisez le graphe d'appel qui correspond au code analysé.
- Proposez une visualisation du graphe de dépendance en utilisant le langage DOT pour créer votre schéma et GraphViz pour la visualisation de votre schéma DOT.

Exemples de GraphViz avec Java : https://github.com/nidi3/graphviz-java

Rendu: Comme l'exercice précédent, veuillez rendre le projet avec le code principal dans la classe *GraphVizMain* ainsi que le fichier DOT contenant le graphe généré lors de l'analyse statique.