IFT 3913 – Qualité du logiciel et métriques
Université de Montréal/DIRO, Automne 2018

Travail pratique #1

Conception et implantation d'un parseur

1. Objectif

Conception et implantation d'un parseur d'un langage d'expression de diagrammes de classes UML.

2. Conditions de réalisation

Groupe de deux au maximum. Travail à remettre au plus tard le **jeudi 4 octobre** lors de la séance de démo pour les rapports et à 23h55 pour le code via Studium.
Aucun retard ne sera accepté.

3. Travail à réaliser

Implanter un programme en Java permettant de charger un diagramme de classes UML à partir d'un fichier. Un diagramme est décrit selon la syntaxe décrite dans la section 5. Ce programme doit offrir une interface graphique permettant de choisir le fichier à parser et de pouvoir visualiser le résultat du parsing (voir un exemple d'interface graphique dans la section 6). Le logiciel doit être en mesure de s'exécuter à partir de la ligne de commande sur les ordinateurs linux qu'on trouve dans les laboratoires du DIRO. Vous devez aussi remettre un rapport de 3 à 5 pages décrivant le logiciel implanté et la conception du programme (au moins un diagramme de classe commenté). Il doit aussi contenir les lignes de compilation et d'exécution du programme. La qualité du français (précision et cohérence) sera prise en compte dans la correction.

4. Barème

1. Rapport incluant la conception du programme 30%
    a. Diagramme de classes
    b. Intégration (compilation et exécution)
    c. Court manuel utilisateur
2. Interface usager 20%
3. Logique et élégance (programme) 50%
    a. Capacité fonctionnelle
    b. Encapsulation
    c. Compréhensibilité

5. Syntaxe et exemple

Un diagramme de classes UML est généralement défini sous la forme d'un graphe dont les nœuds représentent les classes et les liens les différentes relations (association, agrégation et généralisation). Cependant, on peut définir un diagramme de classes sous un format textuel. La grammaire BNF suivante décrit un tel format :

```
<model> ::= "MODEL" IDENTIFIER <list_dec>;
<list_dec> ::= {<declaration>};
<declaration> ::= <class_dec> ";"
     | <association> ";"
     | <generalization> ";"
     | <aggregation> ";";
<class_dec> ::= "CLASS" IDENTIFIER <class_content>;
<class_content> ::= "ATTRIBUTES" <attribute_list>
     "OPERATIONS" <operation_list>;
<attribute_list> ::= [<data_item> {"," <data_item>}] ;
<data_item> ::= IDENTIFIER ":" <type> ;
<operation_list>::= [<operation> {"," <operation>}] ;
<operation> ::= IDENTIFIER <arg_declaration> ":" <type> ;
<arg_declaration> ::= "(" <arg_list> ")" ;
<arg_list> ::= [<data_item> {"," <data_item>}] ;
<type> ::= IDENTIFIER ;
<association> ::= "RELATION" IDENTIFIER "ROLES" <two_roles> ;
<two_roles> ::= <role> "," <role> ;
<role> ::= "CLASS" IDENTIFIER <multiplicity> ;
<multiplcity> ::= "ONE"
     | "MANY"
     | "ONE_OR_MANY"
     | "OPTIONALLY_ONE"
     | "UNDEFINED" ;
<aggregation> ::= "AGGREGATION" "CONTAINER" role "PARTS"
roles;
<roles> ::= <role> {"," <role>} ;
<generalization> ::= "GENERALIZATION" IDENTIFIER "SUBCLASSES"
<sub_class_names> ;
<sub_class_names> ::= IDENTIFIER {"," IDENTIFIER};
```

IDENTIFIER est un symbole terminal.
Un exemple de description de diagramme est donné dans le fichier Ligue.ucd

6. Exemple d'interface usager

La figure suivante donne, à titre d'indication seulement, un exemple d'interface graphique permettant de naviguer dans le diagramme parsé. Toute autre conception graphique de l'interface est possible à la condition de permettre cette navigation.