Université de Montréal

**Devoir 1**

Par  
Marc Laliberté

Et

Ounissa Nait Amer

Bacc. en Informatique

Travail présenté à Edouard Batot

Dans le cadre du cours IFT-3913

Qualité du logiciel et métriques

Octobre 2018

Diagramme de classes

Nous avons tout d’abord commencé à lire la grammaire BNF fournie avec l’énoncé. Nous nous sommes basé sur celle-ci pour créer nos classes modèles.

Nous avons un opter pour une hiérarchie simple ou chaque partie du programme (classe, attributs, opérations, data item) est un objet en soit et tout est encapsulés dans la classes Model (voir diagramme de classe).

Pour minimiser les efforts de codage, nous avons aussi choisis de créer deux classes interfaces java, soient *Modelable* et *Returnable*, ce qui nous a permis de générer rapidement des méthodes semblables sur plusieurs classes. D’ailleurs, pour éviter la robustesse et faciliter l’entretient (exemple si jamais on doit modifier beaucoup de choses pour le tp2), nous avons préférés utilisé des interfaces et ne pas utiliser des classes parents.

Nous avons donc créé le diagramme de classe en se basant sur ces choix. De plus, avec une certaine structure, nous avons pu commencer a séparé le travail qui devait être abattu. Ce fut donc facile de travailler une personne sur le *parsing* et l’autre sur l’interface graphique et ce, en évitant au maximum les *merge conflicts*.

Pour ce qui est du couplage, nous avons essayé de le minimiser au maximum. Le *Parser* est la classe qui interagit surtout avec les modèles et c’est cette classe qui crée le Model qui contient tout. D’ailleurs, nous sommes conscient que le *Parser* est une grosse classe en soit, mais nous n’avons pas vu l’utilité le séparer en plusieurs objets ou classes.

C’est donc les classes de l’interface graphique qui demande un Model au *Parser* lorsque l’on charge un fichier. Ce sont ces classes qui s’occupent de remplir l’interface graphique.  
  
Pour nous aider avec les tests de *Parsing* et vérifier si tout s’est bien déroulé lors du chargement du fichier, nous avons créé la classe *PrinterHelper* qui nous a aidé a afficher les données qui sont chargées et retournés par nos expressions régulières.

Intégration

Manuel utilisateur