INF3140–10 : Devoir 1
Automne 2009

À remettre *au plus tard* jeudi 15 octobre 2009, à 15h30
Pénalité de retard de 10 % par jour (complet ou partiel)
Aucun travail ne sera accepté après jeudi 22 octobre 2009, à 15h30
À faire seul ou en équipe d'au plus deux (2) personnes

Remarque préliminaire : Pour ce travail, vous allez devoir utiliser l'environnement Unix de l'UQAM (machines `rayon[12].labunix.uqam.ca`), puisque l'outil `USE` (*A UML-based Specification Environment*) est installé sur ces machines. C'est aussi par l'intermédiaire de ces machines que vous pourrez utiliser l'outil `oto`, qui sert à la remise (et à la correction) électronique des travaux. Une séance de laboratoire, durant les heures normales de cours, sera organisée pour vous présenter les notions de base d'Unix et vous familiariser avec l'outil `USE` — Jeudi 1^{er} octobre, 15h30.

Modélisation conceptuelle et spécification de contraintes OCL (30 pts)

La figure 1 présente un modèle conceptuel UML pour des informations sur des compagnies, services, employés et véhicules.

1. Traduction du modèle dans la notation `USE` (10 pts)

Traduisez ce modèle conceptuel dans la notation textuelle de l'outil `USE`, en respectant de façon fidèle les noms (entités, relations, attributs, rôles) et cardinalités indiqués sur le diagramme.

Quelques compléments d'information :

- La relation (avec une flèche dont la tête est un triangle) entre `Employe` et `Directeur` est une *relation d'héritage*. Le type `Directeur` est donc un sous-type (une sous-classe) du type `Employe` — en d'autres mots, un `Directeur` est aussi un `Employe`.

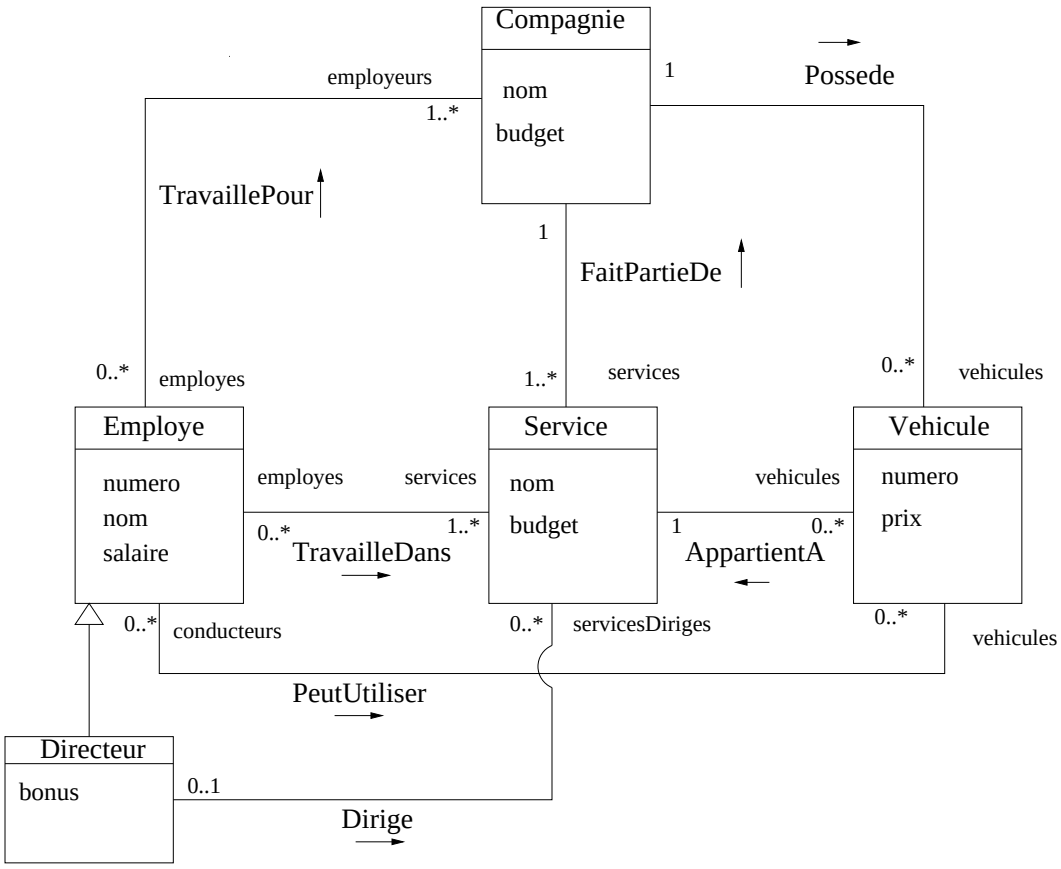

Figure 1: Modèle conceptuel UML pour des compagnies, employés, véhicules.

2. Spécification de requêtes (10 pts)

Spécifiez, en notation USE/OCL, les requêtes décrites ci-bas, en respectant de façon exacte le `nomIndiqué` — et en utilisant le bon *contexte*, indiqué par «Une requête... qui, pour un-e ...,». Si vous le désirez, vous pouvez introduire des requêtes auxiliaires addionnelles — c'est très utile dans certain(s) cas.

a. Une requête `nonConducteurs` qui, pour une compagnie, retourne l'ensemble des employés qui ne peuvent utiliser aucun véhicule.

b. Une requête `conducteursDeTousVehicules` qui, pour une compagnie, retourne l'ensemble des conducteurs qui peuvent utiliser tous les véhicules que possède la compagnie.

c. Une requête `vehiculesDeCompagnie` qui, pour un employé, retourne tous les véhicules de la compagnie (spécifiée en argument) que l'employé peut utiliser.

d. Une requête `dansUnMemeService` qui, pour un employé, détermine si un autre employé (spécifié en argument) travaille dans au moins un même service.

e. Une requête `employesTresBienPayes` qui, pour une compagnie, retourne tous les employés de la compagnie dont le salaire dépasse 10 % du budget de n'importe quel service dans lequel travaillent ces employés.

   Note : le `salaire` d'un `Directeur` est obtenu en combinant son `salaire` de base en tant qu'`Employe` et son `bonus` de `Directeur`.

3. Spécification d'invariants (10 pts)

Spécifiez, en notation USE/OCL, les invariants décrits ci-bas, en respectant le `NomIndiqué` pour l'invariant (nom qui doit apparaître entre «`inv`» et «`:`» dans la spécification de l'invariant) — à vous de bien identifier le contexte. Si vous le désirez, vous pouvez introduire des requêtes auxiliaires addionnelles — c'est très utile dans certain(s) cas.

a. `NumeroUnique` : Le numéro (type `Integer`) est unique pour chaque véhicule.

b. `SalairesEmployesOK` : Dans un service, la somme des salaires des employés qui travaillent dans ce service est inférieure ou égale au budget du service.

c. `ServicesValides` : Pour n'importe quel employé, les services dans lesquels il travaille sont nécessairement des services qui font partie de compagnies pour lesquelles l'employé travaille.

d. `PrixDesVehiculesOK` : Pour n'importe quel service, la somme totale des prix des véhicules qui appartiennent à ce service ne dépasse pas 10 % du budget du service.

e. `DirecteurOK` : Un directeur est nécessairement un des employés d'un service qu'il dirige et il est toujours mieux (strictement plus) payé (en tenant compte de son bonus) que n'importe quel employé qu'il dirige.

Ce qui vous est fourni

Un fichier d'archives (`Devoir1.tar.gz`) est disponible sur le site Web du cours : `http://www.info2.uqam.ca/~tremblay/INF3140/Devoirs/Devoir1.tar.gz`[1]

Ce fichier d'archives contient divers fichiers :

- `vehicules.use` : Un squelette de fichier, que vous devez compléter, pour spécifier le modèle conceptuel pour des compagnies, services, etc.
- `vehicules-ok.cmd` : Un fichier de commandes `USE` définissant un groupe d'objets concrets compagnies, employés, etc. respectant le modèle et les invariants.
- `vehicules-pasok.cmd` : Un fichier de commandes `USE` définissant un groupe d'objets concrets compagnies, employés, etc. ne respectant pas le modèle ou les contraintes.
- `requetes-vehicules.cmd` et `req[a-e].cmd` : Des fichiers de commandes `USE` évaluant diverses requêtes sur les objets construits avec le fichier `vehicules-ok.cmd`.
- `makefile` : Un fichier qui permet d'automatiser certaines commandes pour vérifier vos solutions.

Les principales commandes associées à l'utilisation du `makefile` sont les suivantes (commandes exécutées au niveau du *shell* Unix) :

- `make modele` : Vérifie la syntaxe du modèle décrit dans `vehicules.use` (indépendamment des objets concrets).

  Note : `modele` est la cible par défaut du `makefile`, donc «`make`» (sans argument) équivaut à «`make modele`».
- `make objets` : Vérifie que le modèle décrit dans `vehicules.use` correspond bien aux objets concrets décrits dans `vehicules-ok.cmd`.
- `make requetes` : Compile le modèle décrit dans `vehicules.use`, construit les objets décrits dans `vehicules-ok.cmd`, puis évalue diverses requêtes sur ces objets.

  Note : Une exécution correcte devrait se terminer en produisant une ou plusieurs lignes ayant la forme suivante :

  ```
  use> -> true : Boolean
  ```

  Note : Des cibles auxiliaires ont aussi été définies pour chacune de requêtes : `reqa`, `reqb`, ..., `reqe`. Par exemple, on peut vérifier uniquement la requête de la question b. avec «`make reqb`».

[1] Pour décompresser ce fichier d'archives et obtenir les fichiers, il suffit d'exécuter la commande suivante (notez le «-» à la fin de la ligne) :

```
$ gzcat Devoir1.tar.gz | tar xvf -
```

- `make contraintes-ok` : Compile le modèle décrit dans `vehicules.use`, construit les objets décrits dans `vehicules-ok.cmd`, puis vérifie que les diverses contraintes (invariants) sont satisfaites pour les objets décrits dans `vehicules-ok.cmd` — ce qui devrait être le cas si votre modèle et vos contraintes sont correctement définis.

  Note : Une exécution correcte de cette commande devrait produire une sortie se terminant par une ligne ayant l'allure suivante (le temps indiqué peut évidemment varier) :

  ```
  checked ... invariants in 0.018s, 0 failures.
  ```

- `make contraintes-pasok` : Comme la précédente, mais pour divers objets ne satisfaisant pas les contraintes. Dans ce cas, il y devrait y avoir un certain nombre de *failures*.
- `make clean` : Fait le ménage, c'est-à-dire, supprime les fichiers qui peuvent être générés de façon automatique.
- `make remise` : Voir plus bas.

Ce que vous devez remettre

Remise papier

Vous devez remettre une version papier du fichier suivant :

- `vehicules.use`

Pour la remise de ce document papier, vous devez utiliser **la page couverture (avec grille de correction)** disponible à la page suivante.

Remise électronique

Vous devez aussi remettre une version électronique du fichier précédent, et ce exécutant la commande suivante sur la machine Unix :

```
$ make remise
```

Toutefois, vous devez *auparavant* modifier la valeur de `CODES_PERMANENTS` dans le fichier `makefile` de façon à indiquer vos codes permanent si vous travaillez en équipe (de deux personnes maximum) ou votre code permanent si vous travaillez seul.

Remarques sur la correction :

- Si le fichier remis contient des erreurs de syntaxe, alors vous perdrez une très (très!) grande partie des points.
- J'effectuerai la correction (tant du modèle que des requêtes et des invariants) avec possiblement d'autres objets que ceux décrits dans le fichier `vehicules-ok.cmd` — donc avec des *tests privés* additionnels.
- Une partie des points sera aussi accordée pour le style ⇒ utilisez des expressions les plus simples, claires et explicites possible.
- Pour les exercices 2 et 3, chaque requête ou invariant comptera pour deux (2) points : approx. 25 % pour la syntaxe correcte, 25 % pour les tests publics, 25 % pour les tests publics et 25 % pour le style.

Travail remis à Guy Tremblay

INF3140-10 : Devoir 1
À remettre au plus tard jeudi 15 octobre 2009, 15h30
Ne pas mettre dans une enveloppe

Nom	
Prénom	
Code permanent	
Courriel	

Nom	
Prénom	
Code permanent	
Courriel	

Formalisation du modèle conceptuel		10 pts
Spécification correcte des requêtes		10 pts
Spécification correcte des invariants		10 pts
Total		30 pts