INF8881-30 Hiver 2008 - Examen Intra

Question #1 (30%)

Voici le modèle d'un petit programme écrit dans un langage orienté objet très expressif qui permet, entre autres, la redéfinition des variables d'instance (attributs) :

```
# Note: les classes de la bibliothèque sont : String, Nombre, Liste[E]

class Employeur
  var employes : Liste[Travailleur]
end

class Universite specialize Employeur
  var etudiants : Liste[Etudiant]
end

class Personne
  var nom : String
end

class Travailleur specialize Personne
  var salaire : Nombre
  var employeur : Employeur
end

class Professeur specialize Travailleur
  redef employeur : Universite
  var etudiants : Liste[Etudiant]
end

class Etudiant specialize Personne
  var universite : Universite
  var directeur : Professeur
end

class Stagiaire specialize Etudiant, Travailleur
end
```

Votre tâche est de récrire ce programme dans des langages qui sont moins expressifs mais qui offrent un système de types statiques sûrs.

a) Récrivez ce programme en Java 1.5. Cette version de Java a les propriétés suivantes : héritage simple des classes, héritage multiple des interfaces (une classe peut hériter d'une interface mais pas l'inverse), généricité bornée et typage statique sûr. Note : n'utilisez pas les tableaux, cars ils sont l'exception au typage statique sûr.

b) Récrivez le programme en Java 1.4. Cette version de Java diffère de la version 1.5 en ce qu'elle ne supporte pas la généricité. Note : n'utilisez pas les tableaux, cars ils sont l'exception au typage statique sûr.

c) Récrivez ce programme dans une variation du langage d'origine qui est statiquement sûr (supporte l'héritage multiple, la généricité, la redéfinition d'attributs, etc., mais limite l'expressivité pour rendre le typage statique sûr).

Veuillez, dans la réalisation de ce travail, observer les quelques consignes suivantes :

1. En Java, veuillez utiliser des accesseurs get/set plutôt que des variables d'instance (attributs).
2. Veuillez modéliser le code de façon à conserver le plus d'information de type possible et minimiser l'usage de coercition (*casts*). Par exemple, en b), n'utilisez pas la classe Liste pour stocker une liste d'étudiants; utilisez plutôt la classe ListeEtudiant.
3. À la suite de chacun des programmes (a, b et c), veuillez discuter des avantages et des inconvénients de votre solution proposée.

Question #2 (10%)

Dessinez le méta-modèle complet du programme de la question 1, décrivant les classes, propriétés locales et propriétés globales de ce programme, ainsi que les relations qui les lient, similairement à la figure 3.2 de la thèse de Jean Privat.

Question #3 (20%)

Dans l'article *« Adding Wildcards to the Java Programming Language »* les auteurs identifient deux conditions nécessaires pour que la capture du *wildcard* soit légale (voir section 3.2 de l'article).

a) Illustrez la première condition (type unique) par un exemple complet de code Java qui ne respecte pas cette condition. Expliquez pourquoi c'est illégal.

b) Illustrez la seconde condition (*top level*) par un exemple complet de code Java qui ne respecte pas cette condition. Expliquez pourquoi c'est illégal.

Veuillez, dans la réalisation de ce travail, observer les quelques consignes suivantes :

1. Développez des exemples de code originaux, qui ne s'inspirent pas directement des exemples de l'article. En particulier, ne faîtes pas appel aux classes Set<> et Stack<> (ni à leur cousines telles List<>, Collection<>, etc.).
2. Pour chacune des conditions des exemples, veuillez expliquer comment cette condition pourrait être ennuyantes dans l'écriture de certains programmes.

Question #4 (40%)

Votre tâche est de modéliser un système relativement complexe. On vous offre un langage similaire à celui utilisé pour le programme de la question 1, qui supporte :

1. typage statique (non-sûr)
2. spécialisation multiple
3. généricité (avec covariance, i.e. le type Liste<Etudiant> spécialise le type Liste<Personne>)
4. redéfinition de méthodes et de variables d'instance (avec covariance)

Le système que vous devez modéliser est celui d'un réseau de magasins et d'entrepôts pour la vente d'équipement électronique. Vous devez construire un système d'un minimum de 15 classes. Chaque classe doit posséder quelques variables d'instances et/ou méthodes. Lorsque possible (et logique!), essayez d'inclure des variables d'instances ou des méthodes qui seront redéfinies de façon covariante.

Voici une liste de quelques classes que vous devez inclure dans votre système :

OrdinateurDeTable, OrdinateurPortable, TablePC, IPod, Walkman, ChaineStereo, Entrepot[X], Magasin[X].

Un article doit avoir un prix (de vente) et un coût (de production).

Vous pouvez assumer l'existence d'une bibliothèque de classe contenant les classes usuelles (String, Number et/ou Currency, List, Set, etc.)

a) Écrivez le programme (i.e. le modèle, sans corps de méthodes)
b) Indiquez, dans votre programme (avec des flèches) toutes les infractions qui seraient détectées par un compilateur qui imposerait une politique de typage statique sûr. Indiquez la raison de l'infraction (expliquez brièvement).