INF8881 — Principes avancés des langages à objets
Examen Intra — Contrats
Mercredi 11 février 2009 — Durée 3 heures

1 Programmation par contrats

Les contrats sont des expressions booléennes qui permettent d'introduire une partie des spécifications dans les programmes. Lors de l'exécution, lorsqu'un contrat n'est pas vérifiée, une exception est levée. Le langage EIFFEL est l'un des rares langage à objets muni de contrats. Dans les autres langages, il faut simuler les contrats à la main.

Il existe trois grandes catégories de contrats :

- les préconditions (mot-clé `require` en EIFFEL) doivent être validées à l'entrée des méthodes : elles concernent l'objet receveur et les paramètres de la méthode. C'est à la charge du client de s'assurer que les préconditions sont respectés avant d'utiliser une méthode ;
- les postconditions (mot-clé `ensure` en EIFFEL) doivent être validées à la sortie des méthodes : elles concernent l'objet receveur et la valeur de retour. C'est à la charge de l'implémentation de la méthode de s'assurer que les postconditions sont respectés à la sortie d'une méthode ;
- les invariants de classe (mot-clé `invariant` en EIFFEL) sont des postconditions pour toutes les méthodes de la classe : elles concernent l'objet receveur.

Question 1 : Expliquez pourquoi on peut considérer les invariants de classes comme des *propriétés* des objets au sens du cours ? Est-ce la même chose pour les deux autres type de contrats ?

2 Cas de la pile

Prenons l'exemple la classe `PlateStack` modélisant une pile d'assiettes munie des propriétés suivantes :

- la méthode `push(e : Plate)` qui empile une assiette ;
- la méthode `pop : Plate` qui dépile une assiette de la pile et la retourne ;
- l'attribut `height : Int` qui indique le nombre d'assiettes sur la pile.

Question 2 : Donnez les contrats à ajouter à la classe `PlateStack` (il y en a au moins un de chaque catégorie).

3 Expression du typage statique

Question 3 : Montrez que le typage statique des méthodes et des attributs pourrait s'exprimer au moyen des contrats en remplaçant les annotations de types des propriétés de la pile d'assiette par des contrats.

Question 4 : Quel est néanmoins l'avantage du typage statique par rapport à un typage dynamique couplé avec des contrats ?

4 Contrats et spécialisation

Comment se comportent ces contrats vis-à-vis de la spécialisation et du sous-typage ?

Question 5 : Les contrats doivent-ils s'hériter ? Est-ce la même chose pour les 3 catégories ?

Question 6 : À quelles conditions sur les contrats obtient-on la substituabilité ?

Question 7 : Retrouve-t-on la règle de contravariance pour les contrats qui expriment le typage statique ? (section 3)

Illustrez votre raisonnement en utilisant le cas de la classe `MonotonePlateStack` qui spécialise la classe `PlateStack` en imposant que les assiettes d'une pile aient toute la même couleur (considérez qu'il existe une méthode `color : Color` dans la classe `Plate`).

5 Simulation des contrats

Si les contrats ne sont pas disponibles dans un langage (JAVA ou C++ par exemple), il est possible de simuler facilement les cas simples à la main.

Question 8 : Sur l'exemple de la pile d'assiettes et de la pile monotone d'assiettes, donnez le pseudo-code des méthodes incluant les contrats.

Question 9 : Comment feriez-vous de façon générale pour améliorer la factorisation du code dans le cas des invariants de classes et de l'héritage des contrats ?