1 Projet LOTS

Graphes d'objets

L'objectif du projet est d'implémenter les graphes d'objets dans différents langages et avec différentes techniques, pour mettre en valeur les problèmes de typage statique.

1.1 Spécification des graphes d'objets

Ce sont des `Graphes` composés de `Sommets` et `Arêtes`, les 3 constituant des classes à parité. Ces graphes peuvent être spécialisés dans différents domaines, par exemple des `Molécules`, composées d'`Atomes` et `Liaison`, en chimie et des `Réseaux`, composés de `Noeuds` et de `Liens`, en informatique.

L'objectif est de définir des classes qui interdiront, par construction, l'existence de *chimères*, c'est-à-dire d'objets hybrides qui mélangeraient des caractéristiques issues de différents domaines d'application (chimie et télécommunications par exemple). Il s'agit donc de mettre en valeur les erreurs de types qui peuvent survenir, soit à la compilation, soit à l'exécution, pour montrer que le code protège efficacement contre ces erreurs.

Spécifications détaillées des associations A ces spécifications générales s'ajoutent le fait que les spécifications doivent être implémentées dnas les règles de l'art :

- elle doivent être navigables dans les deux sens, chaque objets connaissant les instances des deux autres classes avec lesquelles il est lié ;
- les associations ne sont pas des agrégations : on peut par exemple enlever un sommet d'un graphe pour le mettre dans un autre ;
- pour simplifier et pour traiter le cas où le rôle d'une association est mono-valué, on considérera qu'un sommet ou arête n'appartient qu'à un seul graphe ;
- du fait du caractère bidirectionnel des associations, la mise à jour doit vérifier des contraintes de symétrie : si l'objet `a` référence l'objet `b`, alors l'inverse doit être vrai aussi ; en particulier, par contrapposition, si l'objet `a` ne référence plus l'objet `b`, alors `b` ne doit plus référencer `a` ; toute mise à jour est donc double !
- lorsque les techniques d'implémentation permettent des erreurs de type à l'exécution, il faut veiller à ce que l'erreur de type de survienne pas alors que la mise à jour est faite à moitié.

1.2 Techniques à implémenter

Il faut utiliser toutes les techniques vues en cours, c'est-à-dire :

covariance pure : seul le langage EIFFEL la permet ;

simulation de la covariance : l'alternative pour tous les autres langages, en se basant sur la surcharge statique pour les langages qui le permettent (JAVA, C++, C#, SCALA, ..) ;

généricité F-bornée : elle peut être utilisée dans tous les langages considérés, sauf PRM/NIT ; en EIFFEL, la covariance s'applique aussi aux instances génériques, donc son utilisation peut être un peu différentes ;

types virtuels : c'est la solution la plus simple et élégante, mais peu de langages la proposent, PRM/NIT, EIFFEL (types ancrés) et SCALA.

1.3 Langages considérés

Sont à considérer :

- les langages à objets *mainstream* comme JAVA, C++ et C# ;
- les langages plus exotiques, qu'ils soient en perte de vitesse comme EIFFEL, en pleine ascension comme SCALA, ou confidentiels comme PRM/NIT ;
- des langages moins directement ciblés par le cours comme ADA 2005 ou OCAML : pour ces derniers langages, il faudra voir ce qui est ou n'est pas faisable.

1.4 Cahier des charges

Chaque étudiant doit respecter les deux contraintes suivantes :

- implémenter les 3 techniques principales, par simulation de la covariance, généricité F-bornée et types virtuels ;
- le faire en JAVA, C++, plus un ou deux (ou plus) autres langages.

Noter que le choix de C# comme troisième et dernier langage ne permet pas de respecter la première contrainte : prenez le en quatrième langage.

Attention, il ne s'agit pas de faire une implémentation complète mais uniquement le squelette : il faut juste le petit nombre de méthodes nécessaire à la mise à jour des 3 associations, pour construire et décconstruire des graphes, et ce qu'il faut pour afficher les objets de façon satisfaisante.

Le projet est personnel, même si vous pouvez, bien sûr, travailler à plusieurs. Le rendu du projet se fera par démonstration individuelle en salle de TP la semaine après les examens de première session. Inutile de rédiger un rapport.