INF8881 — Objets avancés
Examen final

Jean Privat

Mercredi 16 avril 2008

Consignes générales :
- Les documents sont autorisés mais pas nécessaires.
- Veuillez systématiquement justifier et argumenter vos réponses même lorsque ce n'est pas précisé dans l'énoncé des questions.
- En cas de doute sur la compréhension d'une question, n'hésitez-pas à demander des éclaircissements.

1 Exceptions déclarées en Java

Le langage Java oblige le programmeur à déclarer pour chaque méthode les exceptions possiblement levées.

Exemple :

```
class Train {
        void avancer() throws Derailleur, PlusDeCharbon {
                ...
        }
}
```

Lors des redéfinitions, l'ensemble des exceptions déclarées doit être un sous-ensemble des exceptions déclarées dans les super-méthodes.

Exemple :

```
class TrainASustentation extends Train {
        void avancer() throws Derailleur {
                ...
        }
}
```

Dans le cas des méthodes abstraites, le compilateur déduit automatiquement les exceptions déclarées manquantes par l'intersection des ensembles des exceptions déclarées dans les super-méthodes.

Exemple :

```
interface A { void foo() throws E1, E2, E3; }
interface B extends A { void foo() throws E1, E2; }
interface C extends A { void foo() throws E2, E3; }
interface D extends B, C {} // implicitly, foo() throws only E2
```

Questions

1. Quelles sont les contraintes que doivent respecter les classes `Ex1`, `Ex2` et `Ex3` pour que le code suivant soit valide :

```
interface E { void foo() throws Ex1; }
interface F { void foo() throws Ex2; }
interface G extends E, F { void foo() throws Ex3; }
```

2. La variation des exceptions est-elle covariante, contra-variante ou aucune des deux ? Justifiez votre réponse.
3. Expliquez la raison du sens de variation des exceptions déclarées.
4. Montrez les limites concrètes d'une telle contrainte dans la programmation de tous les jours. Donnez au moins un exemple.
5. Dans le cadre de la déduction par le compilateur des exceptions déclarées, que se passe-t-il lorsque l'intersection est vide ?
6. En fonction des relations entres les classes `Ex4` et `Ex5`, quelles sont les exceptions déclarées déduites pas le compilateur pour la méthode `foo` de l'interface `V` dans le code suivant :

```
interface T { void foo() throws Ex4; }
interface U { void foo() throws Ex5; }
interface V extends T, U { }
```

7. Que se passerait-il si la classe racine des exceptions (`Throwable`) était une interface, ou si Java était complètement en spécialisation multiple ? Donnez un exemple pour illustrer votre propos.
8. Proposez plusieurs solutions pour résoudre les problèmes identifiés à la question précédente. Discutez des solutions que vous proposés. Quels sont leurs avantages et leurs inconvénients ?

2 Raffinement et sélection multiple

L'objectif de cet exercice est de comparer la sélection multiple et le raffinement de classes sur un cas concret évoqué en cours : les visiteurs.

En préalable, voici trois modules écrits dans un pseudo-langage qui possède les caractéristiques suivantes :

- hiérarchie de modules ;
- raffinement de classes ;
- multi-méthodes.

```
# This module defines syntax trees of numeric expressions
module expression
import standard

# The root of the expression hierarchy
abstract class Expr
end

class Number
specialize Expr
        var value: Int
end

abstract class BinOp
specialize Expr
        var left: Expr
        var right: Expr
end

class Addition
specialize BinOp
end

class Multiplication
specialize BinOp
end
```

```
# Abstract expression visitor implemented only with multi-methods
module mm_visitor
import expression
class Visitor
        # the 'visits' method should:
        # * be called to start a visit on a node
        # * not be redefined in concrete visitors
        fun visits(e: Expr)
        do
                visit(e)
        end
        fun visits(e:BinOp)
        do
                visits(e.left)
                visits(e.right)
                visit(e)
        end
```

```
        # the 'visit' method should:
        # * not be called directly
        # * be redefined in concrete visitors to perform the job
        fun visit(e: Expr) is abstract
end
```

```
module mm_calc
import mm_visitor

class Calc
specialize Visitor
        var values: HashMap[Expr, Int] = new HashMap[Expr, Int]
        fun visit(e: Number)
        do
                values[e] = e.value
        end
        fun visit(e: Addition)
        do
                values[e] = values[e.left] + values[e.right]
        end
        fun visit(e: Multiplication)
        do
                values[e] = values[e.left] * values[e.right]
        end
end
```

Questions

Note : Plus que le code en lui-même, le correcteur attendra d'avantage de l'étudiant une réflexion sur les questions posées et les réponses proposées ainsi qu'une justification de ces réponses.

1. Créez un module `raf_calc` qui dépend du module `expression` et permet de calculer la valeur d'une expression. Utilisez du raffinement, vous n'avez pas droit à la sélection multiple, n'utilisez pas de visiteur.
2. Créez un module `raf_soustraction` qui dépend du module `raf_calc` et ajoute un nouveau type d'expression : la soustraction. Assurez-vous que le calcul d'une valeur fonctionne toujours. Vous pouvez utiliser du raffinement mais vous n'avez pas droit à la sélection multiple.
3. Créez un module `raf_visitor` qui dépend du module `expression` et définit une classe abstraite `Visiteur` qui permet d'abstraire un visiteur sur une expression (un peu comme dans `mm_visitor`). Vous pouvez utiliser du raffinement mais vous n'avez pas droit à la sélection multiple.
4. Créez un module `raf_opcount` qui dépend du module `raf_visitor` et qui définit un visiteur `OpCount` qui compte le nombre d'opérations binaire (addition et soustraction) d'une expression. Vous pouvez utiliser du raffinement mais vous n'avez pas droit à la sélection multiple.
5. Créez un module `raf_mm_oposite` qui dépend de `mm_calc`, ajoute un nouveau type d'expression, l'opération unaire « négation », et raffine les classe `Visitor` et `Calc` afin de gérer correctement ce nouveau type d'expression. Vous pouvez utiliser le raffinement et la sélection multiple. Discutez de l'utilisation conjointe du raffinement et de la sélection multiple.
6. **Question subsidiaire :** Que se passerait-il si on construisait un module vide dépendant des modules `raf_opcount` et `raf_mm_oposite` ?