UNIVERSITÉ DE ROUEN

Architecture Logicielle – TP 08 2014–2015

Les patrons de responsabilité: le *proxy dynamique*

Exercice 1 – Codage d'un Proxy dynamique

Un logiciel de gestion de personnel permet d'enregistrer et manipuler toutes les informations relatives aux employés d'une entreprise. Certaine de ces informations sont confidentielles (comme le salaire et le numéro de sécurité sociale), d'autres peuvent alimenter un annuaire (adresse de bureau, numéro de téléphone, etc.). Ce logiciel utilise la classe suivante :

```
1  public class Employee implements IEmployee {
2    private String name;
3    private String address;
4    private String phoneNumber;
5    private String SSN;
6    private String department;
7    private float salary;
8
9    public Employee(String name, String address, String phoneNumber,
10     String SSN, String department, float salary) {
11     this.name=name; this.address=address; this.SSN=SSN;
12     this.phoneNumber=phoneNumber; this.department=department;
13     this.salary=salary;
14   }
15
16   public String getName() { return name; }
17   public String getAddress() { return address; }
18   public String getPhoneNumber() { return phoneNumber; }
19   public String getSSN() { return SSN; }
20   public String getDepartment() { return department; }
21   public float getSalary() { return salary; }
22
23   public void setName(String name) { this.name = name; }
24   public void setAddress(String address) { this.address = address; }
25   public void setPhoneNumber(String phoneNumber) {
26     this.phoneNumber = phoneNumber; }
27   public void setSSN(String ssn) { this.SSN = SSN; }
28   public void setDepartment(String department) {
29     this.department = department; }
30   public void setSalary(float salary) { this.salary = salary; }
31 }
```

On souhaite mettre en oeuvre une politique de sécurité visant à donner accès à certaines informations à une catégorie de personnel particulière. Une politique de sécurité sera représentée par un objet dont la classe implémente l'interface suivante :

```
1  public interface IPolicy {
2    public void allow(String rule);
3    public void deny(String rule);
```

```
4     public boolean isAllowed(String rule);
5  }
```

Cette classe permet de déterminer si un accès particulier est autorisé grâce à la méthode `isAllowed()`. Par exemple : `isAllowed("setAddress");` où `setAddress` est le nom de la méthode à invoquer. Les méthodes `allow()` et `deny()` permettent respectivement d'ajouter ou bien de retirer des droits accès (on pourra utiliser la classe `java.util.HashSet` pour enregistrer les règles d'accès). Pour simplifier la création de ces politiques, on écrira deux classes implémentant l'interface `IPolicy` : `AllowFirst` qui interdit les règles qu'elle contient, et `DenyFirst` qui autorise uniquement les règles qu'elle contient. Voici un programme d'exemple :

```
1  public class Test {
2    public static void main(String[] args) {
3      // Partie serveur :
4      IEmployee e=new Employee("Marcel-Paul Schutzenberger", "Universite de
           Paris VII", "0235020202", "1729999999999", "Dep. Mathematiques",
           1697);
5      IPolicy pol=new DenyFirst();
6      pol.allow("getName");
7      pol.allow("getAddress");
8
9      IEmployee returned_employee=(IEmployee)PolicyProxy.newInstance(e, pol,
           new Class[] { IEmployee.class } );
10
11     // Partie cliente :
12     System.out.println("Nom de l'employe : "+returned_employee.getName());
13     System.out.println("Adresse de l'employe : "+returned_employee.
           getAddress());
14     System.out.println("Numero de telephone de l'employe : "+
           returned_employee.getPhoneNumber());
15   }
16 }
```

Ce programme instancie un objet `Employee`, ainsi qu'un objet de type `DenyFirst` à partir desquels il créé un proxy. Cette partie (lignes 3 à 10) représente la mécanique coté serveur, normalement invisible du client. La suite du programme représente l'activité du client. Celui-ci affiche les valeurs des attributs de l'objet en appelant les accesseurs : avec succès pour les deux premiers mais le troisième provoque une exception car le proxy ne trouve pas `getPhoneNumber` dans l'objet de politique de sécurité qui lui a été associé.

Question 1.1 : Donnez le schéma UML de l'ensemble du système.
Question 1.2 : Implémentez les classes `AllowFirst` et `DenyFirst`. Vous trouverez les fichiers squelettes sur votre liste de diffusion.
Question 1.3 : Implémentez la classe `PolicyProxy`. Assurez-vous que ce proxy soit générique et puisse être utilisé avec n'importe quel type de classe.
Question 1.4 : Ajoutez dans la classe proxy une méthode `String whoAmI()` qui renvoie le nom de la classe de l'objet ainsi que la liste des interfaces qu'elle implémente. Appliquez la sur l'objet `returned_employee`.
Question 1.5 : Le prototype de la méthode `newInstance` n'est pas consistant : on pourrait créer un proxy pour une ou des interfaces que l'objet contrôlé n'implémente pas, de même si on oublie une des interfaces effectivement implémentées (voir la ligne 9 du programme d'exemple). Ajoutez la méthode

```
1  public Object newInstance(Object e, IPolicy p);
```

qui corrige ce problème.