Université de Caen Normandie
Département d'informatique
L3 informatique, 2021-2022
Unité INF5A1, Session 1
Le 22 novembre 2021, 9h-11h
Barème donné à titre indicatif, pouvant être légèrement modifié
Documents autorisés

Exercice 1 : Personne et Groupe

Important : vous pouvez utiliser ici les types vus en cours concernant le pattern Observer (AbstractModeleEcoutable, EcouteurModele), **sans les réécrire** (cela ne rapporte pas de points), ce qui vous permettra de gagner beaucoup de temps.

On dispose des deux interfaces suivantes :

```
public interface Personne {
   String getNom();
   void  setNom(String nom);
   int getAnneeNaissance();
   void ajoutEcouteur(EcouteurModele e);
}
```

```
public interface Groupe {
   int getNombrePersonnes();
   void ajout(Personne p);
   Personne getPersonne(int i);
   void ajoutEcouteur(EcouteurModele e) ;
}
```

Question 1. (4 points)

Ecrire une implémentation de Personne, nommée PersonneImpl
Ecrire une implémentation de Groupe, nommée GroupeImpl

Attention : il faut qu'un Groupe prévienne ses écouteurs si bien sûr on lui ajoute une personne, mais **aussi** si l'une de ses personnes est modifiée.

Question 2. (2 points)

Ecrire la méthode statique afficheGroupe(Groupe g) permettant d'afficher le contenu d'un groupe de la façon suivante (pour un groupe de 2 personnes nées en 2000 et 2002, nommées Pierre et Elise) :

Affichage du groupe :[Pierre,2000][Elise,2002]

Question 3. (3 points)

Ecrire le Decorator PersonneRajeunie, qui augmente[1] l'année de naissance d'une certaine valeur de rajeunissement. On proposera 2 constructeurs, l'un qui permet de spécifier cette valeur de rajeunissement, l'autre qui propose un rajeunissement par défaut de 2 ans.
Penser à transmettre les événements de l'objet écouté.

Question 4. (2 points)

On dispose par ailleurs du type Person (et de classes d'implémentations non présentées ici...), provenant d'un autre développeur, défini ainsi :

```
public interface Person {
   String getName();
   Date getBirthDate();
}
```

On souhaite pouvoir ajouter des instances de ce type à un Groupe.

- Indiquer le pattern à utiliser
- Ecrire le code correspondant

Pour simplifier, on pourra utiliser la méthode getYear() de la classe Date qui donne le nombre d'années écoulées pour cette date depuis 1900 (cette méthode est « dépréciée » mais simplifie le code à écrire).

Remarque : ce type n'étant pas écoutable, on ne gérera pas l'aspect événementiel.

Exercice 2 : Vérification de droits

On souhaite vérifier qu'une Personne (cf. exercice 1 pour voir ses méthodes) a le droit ou non d'accéder à certains services proposés par une application.

Pour cela, on va mettre en œuvre le pattern *Chain Of Responsibility*.

Deux types de vérifications (donc deux classes de maillons) sont à créer :

ListeNoire : une liste noire possède une méthode void ajoutPersonneInterdite(Personne p) permettant de bannir des utilisateurs. Lorsqu'un utilisateur appartient à sa liste, il lui interdit l'accès au service.

AgeSuffisant : une instance de cette classe est paramétrée (via son constructeur) avec une valeur d'âge minimal à avoir (sous forme d'entier) pour accéder à un service. Par exemple une instance crée par « new AgeSuffisant(18) » interdit l'accès aux mineurs. NB : « new Date() » crée une instance de la date actuelle, donc (new Date()).getYear() indique le nombre d'années écoulées depuis 1900.

[1] rappel : pour diminuer l'âge, il faut augmenter l'année de naissance

Une chaîne de vérification aura pour méthode principale « public void boolean personneAutorisee(Personne p) » et devra répondre *true* uniquement si tous ses maillons répondent *true*. Dès que l'un des maillons répond *false*, la vérification se termine immédiatement par un *false*.

Chaque maillon doit donc vérifier sa propre condition, si celle-ci est false il doit renvoyer false, et si elle est *true*, il doit déléguer la vérification à son maillon suivant (s'il en a un). Il y a donc une partie de code qui va être redondante entre ces 2 classes. Pour pallier ce problème, vous essaierez de mettre en place le pattern *Template Method* (si vous n'y parvenez pas, faites sans).

Question 4. (2 points)

Ecrire le diagramme de classe correspondant (avec les méthodes, et en y intégrant le pattern *Template Method* si vous y parvenez).

Question 5 (5 points)

Ecrire le code de vos classes

Question 6. (2 points)

Ecrire le code permettant de créer la chaîne de vérification suivante : Personne n'étant pas sur la liste noire et ayant un âge supérieur à 12.

En supposant que l'on ait reçu une référence Personne personne, écrire la ligne de code (avec un println) indiquant si l'utilisateur satisfait la chaîne.