TD 8 - TP télécommande - construction d'`Adapter`

Objectif pédagogique

A l'issue de ce TP, vous devrez

- avoir réfléchi à la manière de faire le lien entre des interfaces différentes ;
- savoir comment intégrer une classe extérieure dans une application sans la modifier ;
- savoir ce qu'est le design pattern `Adapter`.

Dans cette partie, nous allons ajouter de nouveaux appareils à la télécommande et voir comment il va être possible de les insérer facilement.

Le sujet va vous conduire vers ce qu'on appelle un `Adapter`. Ce mot fait partie des notions à retenir du cours.

Rendu

A l'issue du Tp, vous devrez rendre sur arche le diagramme de classe intégrant le variateur de lumière ainsi que déposer votre code dans votre dépot.

1 Variateur de Lumière (1h10)

On vous fournit sur arche une classe `Cheminee` représentant une cheminée électrique. Cette classe est caractérisée par un attribut `intensite` correspondant à l'intensité lumineuse du variateur.

```
/**
 * classe qui permet de modeliser une cheminee electrique
 * ce classe doit etre utilisee avec la telecommande
 *
 * ATTENTION: CONTRAINTE IMPORTANTE
 * CETTE CLASSE NE DOIT JAMAIS ETRE MODIFIEE
 * (IMAGINEZ QUE VOUS NE DISPOSEZ QUE DU .CLASS)
 */
public class Cheminee {

  /**
   * intensite de la cheminee modulable par le variateur
   * valeur comprise entre 0 et 100;
   */
```

```
  int intensite;

  /**
   * constructeur par defaut
   * un variateur eteint
   */
  public Cheminee()
  {
    this.intensite=0;
  }

  /**
   * permet de changer l'intensite de la cheminee
   * @param i nouvelle intensite de la cheminee
   */
  public void changerIntensite(int i)
  {
    if (i>=0&&i<=100)
    this.intensite=i;
  }

  /**
   * retourne l'intensite de la cheminee
   * @return intensite de la cheminee
   */
  public int getLumiere()
  {
    return this.intensite;
  }

  /**
   * methode toString, affiche cheminee et la valeur intensite
   * @return description de l'objet this
   */
  public String toString()
  {
    return("cheminee: "+intensite);
  }
}
```

Le probleme posé par ce TP vient du fait que la classe `Cheminee` ne fonctionne pas comme un appareil habituel. En effet, elle ne possède pas les méthodes `allumer` et `eteindre` et n'implémente pas l'interface `Appareil`.

L'objectif de cette partie est de trouver un moyen pour que cette classe `Cheminee` puisse être utilisable par un objet de type `Telecommande` sans modifier son contenu.

Attention :
Comme indiqué en commentaire, le contenu du fichier `Cheminee.java` NE DOIT EN AUCUN CAS ETRE MODIFIE. Il est ainsi interdit d'ajouter ou de modifier une méthode ou un attribut de la classe.

Cette contrainte de non modification s'explique par le fait que la classe `Cheminee` doit être considérée comme une classe fournie par une personne extérieure. Il n'est pas

dit que vous disposiez des sources et même si c'était le cas, il est possible que cette classe soit utilisée dans d'autres programmes que vous ne contrôlez pas.

1.1 Première solution

Une première solution consisterait à modifier la classe `Cheminee` pour la faire implémenter de l'interface `Appareil` et y ajouter les méthodes adéquates. Cependant, les contraintes du sujet empêchent cette approche.

Dans tous les cas, on supposera par la suite, qu'éteindre la cheminée revient à mettre son intensité à 0 et qu'allumer la cheminée consiste à augmenter son intensite de 10.

1.2 Phase de réflexion (20 min)

Réfléchissez à la manière de faire cette adaptation en limitant au maximum le nombre de lignes de l'application à modifier (avec le contrainte que la classe `Cheminee` reste fixe).

Comme point de départ, vous pouvez

- réfléchir à ce qui est **nécessaire** pour pouvoir manipuler un objet avec une télécommande ;
- ensuite vous poser des questions sur comment prendre en compte la classe `Cheminee` ;
- enfin, proposer une solution à partir de ces réflexions.

En cas de besoin, n'hésitez pas à discuter avec votre enseignant.

Question 1
Expliquer votre solution pour manipuler la classe `Cheminee` comme un `Appareil`.

1.3 Phase de conception (10 min)

Question 2
Une fois que vos idées sont claires, écrivez un diagramme de classe qui présente votre conception avec l'ensemble des classes disponibles depuis le début des TP.

Validez le diagramme de classe avec votre enseignant avant de passer à la suite

1.4 Phase de réalisation (10 min)

Question 3

Ecrire la/les classes correspondant à ce diagramme de classe et le profil des méthodes attendues.

1.5 Phase de test (10 min)

Pour vérifier que la classe télécommande fonctionne encore, validez les anciens tests effectués sur la télécommande et ajoutez deux nouveaux tests permettant de vérifier qu'il est possible d'allumer et d'éteindre une cheminée.

Question 4

Compléter la classe `TestTelecommande` pour valider la classe `Telecommande`.

1.6 Utilisation de la telecommande (10 min)

La classe `TelecommandeGraphique` fournie sur arche permet de générer automatiquement une interface graphique pour manipuler une télécommande. Il est possible de créer une interface en passant en appelant le constructeur de cette classe en passant un objet `Telecommande` **deja construit avec les appareils ajoutés**.

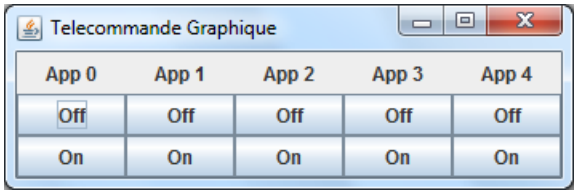

Figure 1: Interface graphique correspondant à une telecommande avec 5 appareils.

Attention :
Pour que l'interface graphique fonctionne, vous devez ajouter une méthode `public int getNombre()` qui retourne le nombre d'appareils de la télécommande.

Question 5

Écrire un `main` qui construit une interface graphique manipulant un objet `Lampe`, un objet `Hifi` et un objet `Cheminee` en utilisant votre travail.

Rendu

Une fois que tout est validé, sauvez votre projet sous la version `fin_cheminee`. Pensez à déposer votre diagramme de classe sous arche.

2 Utilisation d'une bibliothèque fournie (40 min)

2.1 Présentation de la classe Thermostat (5 min)

On vous fournit la classe de nom complet `thermos.Thermostat` permettant de modifier la température d'une pièce. Le nom complet de la classe signifie que cette classe se nomme `Thermostat` et appartient au package `thermos` qu'il faudra donc inclure.

Cette classe dispose de trois méthodes

- un constructeur par défaut
- une méthode `void monterTemperature()` qui augmente la température de 1 degré (max 25)
- une méthode `void baisserTemperature()` qui baisse la température de 1 degré (min 15)

Elle dispose en outre d'une interface graphique qui apparaît à la construction de l'objet et se met automatiquement à jour lorsqu'une méthode est appelée.

Désormais, vous ne disposez plus du code source sur arche mais seulement des fichiers `.class` dans une archive `.jar`. De plus, vous ne savez pas comment l'interface graphique fonctionne et n'avez pas à le savoir.

2.2 Ajout d'un fichier .jar sous Eclipse (5 min)

Il est possible d'ajouter ce fichier `.jar` dans votre projet Eclipse, ainsi les classes contenues dans cette archive `.jar` seront reconnues dans votre projet.

Pour cela, il suffit de déposer le fichier `.jar` dans votre projet et d'aller dans le menu contextuel du projet `-> Properties -> Java Build Path -> Libraries -> add external jar`.

Si l'ajout s'est bien passé, vous devriez voir apparaitre une nouvelle bibliothèque en dessous de la bibliothèque contenant l'ensemble des classes java `JRE System Library`.

Question 6

Ajouter le fichier `.jar` dans votre projet et dans votre dépot.

2.3 Solution Adaptateur (20 min)

Nous allons réutiliser la même démarche que précédemment pour faire un `Adapter` permettant d'utiliser la classe `Thermostat`. Éteindre correspond à baisser la température d'un degré et allumer à l'augmenter.

Question 7

Dessiner un diagramme de classe qui propose une solution de type `Adapter` pour intégrer la classe `Thermostat`

Question 8

Ecrire le code correspondant.

2.4 Utilisation d'un thermostat (5 min)

Question 9

Faire un `main` qui permette de vérifier que vous pouvez modifier le thermostat avec la télécommande (éventuellement son interface graphique).