Examen (1h45, documents autorisés)

Nom : **Prénom :**

Préambule : Ne pas mettre de commentaires de documentation sauf s'ils sont nécessaires à la compréhension.

Les exercices sont relativement indépendants.

Barème indicatif :

Exercice	1	2	3	4	5	6
Points	4	2	2	2	5	5

Exercice 1 Étant données les classes du listing 1, répondre aux questions suivantes en justifiant les réponses apportées.

1.1 Peut-on réellement définir les deux méthodes `m1` de la classe `B` ?

1.2 Y a-t-il un lien entre les méthodes `m1()` de `B` et celles de `A` ?

1.3 Combien y a-t-il de méthodes dans la classe `B` ?

1.4 Y a-t-il un lien entre les méthodes nommées `m1` dans `Test` et celles nommées `m1` de `B` ?

1.5 Pour chaque instruction de la méthode `main` de `Test`, indiquer si le compilateur l'accepte ou non et, dans l'affirmative, le résultat à l'exécution.

Exercice 2 : Collections

On considère le programme Java du listing 2.

2.1 Indiquer ce qu'il affiche quand on remplace les « ... » par :

1. un objet créé du type List
2. un objet créé du type Set
3. un objet créé du type SortedSet

2.2 Pour chacun des cas précédents, indiquer la classe Java à utiliser pour créer l'objet.

Télécommande universelle

L'objectif de ces exercices est de définir une télécommande universelle capable de piloter plusieurs équipements de n'importe quel type. Cette télécommande se limite à pouvoir démarrer et arrêter un tel équipement. La figure 1 présente une visualisation possible de cette télécommande.

Listing 1 – Les classes `A`, `B` et `Test`

```
1  class A {
2     public void m1() {
3        System.out.println("A.m1()");
4     }
5
6     public void m1(String t1, String t2) {
7        System.out.println("A.m1(" + t1 + ",␣" + t2 + ")");
8     }
9  }
10
11
12 class B extends A {
13    public void m1() {
14       System.out.println("B.m1()");
15    }
16
17    public void m1(String texte) {
18       System.out.println("B.m1(" + texte + ")");
19    }
20 }
21
22
23 class Test {
24
25    private static void m1(A a, B b) {
26       System.out.println("Test.m1(A,␣B)");
27    }
28
29    private static void m1(B b, A a) {
30       System.out.println("Test.m1(B,␣A)");
31    }
32
33    public static void main(String[] args) {
34       A a = new A();
35       B b = new B();
36       A ab = new B();
37       B ba = new A();
38
39       a.m1();
40       a.m1("test");
41       a.m1("1", "2");
42       b.m1();
43       b.m1("test");
44       b.m1("1", "2");
45       ab.m1();
46       ab.m1("test");
47       ab.m1("1", "2");
48
49       m1(a, a);
50       m1(a, b);
51       m1(b, a);
52       m1(ab, a);
53       m1(a, ab);
54       m1(b, b);
55    }
56 }
```

Listing 2 – La classe Collecteur

```
1  import java.util.*;
2
3  public class Collecteur {
4
5        static public void collecter(Collection<Integer> c, int[] elements) {
6              for (int s : elements) {
7                    c.add(s);
8              }
9        }
10
11       public static void main(String[] args) {
12             Collection<Integer> c = ...;
13             int[] tab = { 2, 5, 1, 10, 1, 2, 15, 1 };
14             collecter(c, tab);
15             System.out.println(c);
16       }
17
18 }
```

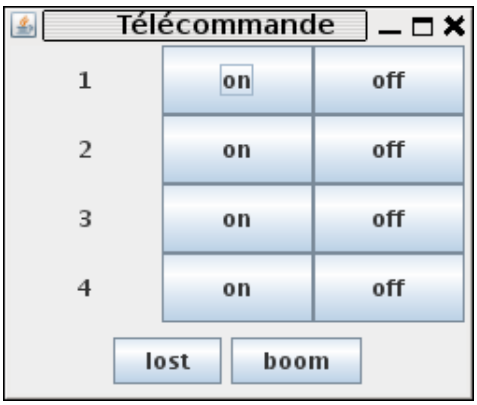

FIGURE 1 – Exemple de visualisation de la télécommande

La télécommande est composée de plusieurs lignes numérotées. Une ligne commande un équipement. Le bouton « on » permet de mettre en route cet équipement et le bouton « off » permet de l'arrêter.

Le bouton « lost » permet de simuler la perte de la télécommande. Ceci se traduit par le fait de rendre invisible la fenêtre Swing qui la représente. Le bouton « boom » permet de simuler une chute de la télécommande qui provoque sa casse et donc l'arrêt de l'application.

Nous allons considérer deux équipements, un ventilateur et un spot. Le premier peut être démarré et arrêté. Le second peut être allumé et éteint. Le code de ces deux classes est fourni aux listings 3 et 4. On constate que ces deux équipements sont simplement simulés grâce à des affichages sur la sortie standard.

Exercice 3 : La classe Couple

Commençons par programmer la classe Couple que nous utiliserons dans la suite. Elle permet de regrouper deux valeurs que l'on notera a et b et qui peuvent être de types différents. Par exemple, on pourrait considérer un couple composé d'un réel et d'une chaîne de caractères et un autre de deux points. Quand on construit un couple, on doit préciser les valeurs de a et b. Il est ensuite possible d'accéder individuellement aux valeurs de a et b et de les modifier.

3.1 Indiquer quel est le concept qui permet d'écrire une classe Couple quand on ne connaît pas encore les types de a et b.

3.2 Écrire la classe Couple.

3.3 Expliquer comment créer un premier couple construit à partir des chaînes de caractères « un » (valeur de a) et « deux » (valeur de b) et un deuxième à partir de l'entier 5 et le caractère « x ».

Exercice 4 : Le patron Commande

Pour assurer l'indépendance de la télécommande universelle vis-à-vis des différents équipements, on utilise le patron de conception *Commande*. Le diagramme de classe de ce patron présenté par GoF est donné à la figure 2.

L'invocateur est ici la télécommande. Il utilise la méthode `exécuter` de Commande qui appelle une méthode du récepteur qui ici sera un ventilateur ou un spot. L'invocateur n'a pas à connaître les récepteurs, il connaît simplement la notion de *commande*.

4.1 Expliquer les mécanismes objet qui font que l'invocateur n'a pas à connaître le récepteur.

Listing 3 – La classe Ventilateur

```
1  public class Ventilateur {
2
3      public void demarrer() {
4          System.out.println("Le␣ventilateur␣est␣en␣marche.");
5      }
6
7      public void arreter() {
8          System.out.println("Le␣ventilateur␣est␣arrêté.");
9      }
10
11 }
```

Listing 4 – La classe Spot

```
1  public class Spot {
2
3        public void allumer() {
4           System.out.println("Le␣spot␣est␣allumé.");
5        }
6
7        public void eteindre() {
8           System.out.println("Le␣spot␣est␣éteint.");
9        }
10
11 }
```

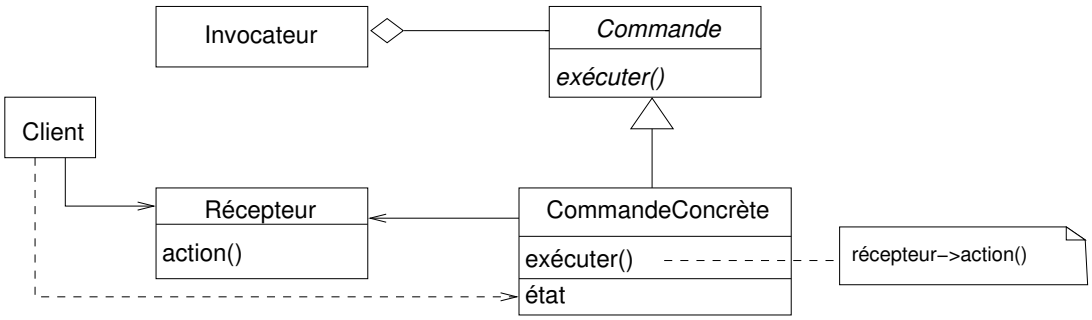

FIGURE 2 – La patron de conception Commande d'après GoF.

4.2 Écrire en Java le code de Commande.

4.3 Écrire en Java une première commande qui permet d'allumer un spot et une seconde qui permet d'éteindre un spot. On supposera que des commandes équivalentes sont définies pour démarrer et arrêter un ventilateur.

Exercice 5 : La télécommande

Définissons maintenant la télécommande, plus précisément un modèle de télécommande. Quand on crée une télécommande, on précise le nombre d'équipements qu'elle peut commander. Dans l'exemple de la figure 1, la télécommande peut piloter jusqu'à 4 équipements. Lorsque l'on crée une télécommande, les commandes « on » et « off » de chaque ligne sont initialisées à une commande sans effet. Les opérations disponibles sur une télécommande permettent de :

- obtenir le nombre d'équipements que peut piloter cette télécommande ;
- connecter un équipement à la télécommande en précisant le numéro de la ligne concernée et les deux commandes pour démarrer et arrêter l'équipement ;
- actionner le bouton pour démarrer un équipement associé à un certain numéro ;
- actionner le bouton pour arrêter un équipement associé à un certain numéro ;
- obtenir le couple de commandes pour démarrer et arrêter l'équipement associé à une ligne.

5.1 Écrire la classe `TélécommandeUniverselleModele` (on pourra écrire simplement `TUM`). Cette classe devra :

- définir un attribut de classe constant « cmdNOP » correspondant à la commande sans effet.
- utiliser l'interface `java.util.List` pour conserver les commandes des équipements.

5.2 Dessiner le diagramme de séquence correspondant au scénario suivant qui suppose que les objects télécommande, ventilateur et spot existent.

- connecter le ventilateur sur la ligne 1 de la télécommande ;
- connecter le spot sur la ligne 2 de la télécommande ;
- actionner le bouton « démarrer » de la ligne 2 de la télécommande ;
- actionner le bouton « arrêter » de la ligne 2 de la télécommande.

5.3 Écrire un programme principal qui correspond au scénario précédent.

Exercice 6 : La télécommande graphique

On souhaite maintenant définir un composant graphique en Swing qui correspond à une télécommande universelle. L'apparence d'une telle télécommande est donnée à la figure 1.

6.1 Expliquer comment créer la vue de ce composant. On pourra répondre directement sur le sujet en annotant la figure 1.

6.2 Expliquer comment rendre actifs les différents boutons « on », « off », « lost » et « boom ».