

INTRODUCTION A LA PROGRAMMATION

Corrigé test semestre 1

Exercice 2 : Conception OO et programmation [50 points]

Voir le fichier source `Simulation.java`.

BARÈME : Le barème est donné en commentaire dans ce fichier

Exercice 3 : Concepts de base[20 points]

1. [3 points] Le code affiche

0

Les attributs de type `int` sont en effet initialisés par défaut à zéro. Comme Java ne permet que le passage par valeur des arguments, la méthode `m` appelée en ligne 14 n'a pas d'impact sur l'objet `a` créé en ligne 13.

BARÈME :

- (a) 1 pour la réponse
 - (b) 2 pour la justification : 1 pour le mention de l'initialisation par défaut et 1 pour la mention du passage par valeur.
2. [5 points]

Voici un code possible pour le type énuméré :

```
enum SemaphoreColor {
    RED("rouge", false),
    GREEN("vert", true),
    YELLOW("jaune", false);

    private SemaphoreColor(String name, boolean canPass) {
        this.name = name;
        this.canPass = canPass;
    }

    private String name;
    private boolean canPass;

    public String frenchName() { return name; }
    public boolean canPass() { return canPass; }
}
```

BARÈME : 0.5 point par valeur de l'énumération, 0.5 point par attribut, 0.75 par méthode (canPass et frenchName) et 1 pour le constructeur dont 0.5 pour sa déclaration en private

3. [7 points]

- (a) (1 pts) B a trois attributs (b, son attribut a spécifique et l'attribut a hérités de A).
- (b) (1 pts) deux constructeurs : celui par défaut et celui avec un paramètre.
- (c) (1 pts) un seul (le constructeur par défaut par défaut et les constructeurs ne sont pas hérités).
- (d) (1 pts) Le code affiche
 - A: 1
 - B: 3 2
 - B: 3 2
- (e) (1 pts) si on supprime la ligne 17, le code affiche
 - A: 1
 - B: 3 1
 - B: 3 1
- (f) (0.5 pts) oui, this() invoque simplement le constructeur par défaut.
- (g) (1.5 pts) le code ne compile plus : le constructeur par défaut par défaut disparaît s'il y a un constructeur explicite (ce qui est le cas dans A) or le constructeur par défaut par défaut de B invoque implicitement le constructeur par défaut de sa super-classe.

4. [5 points]

- (a) (3 pts) la ligne 10 essaie d'accéder à l'attribut a qui est privé, la ligne 11 déclare un @Override alors que la méthode m de B n'est pas une redéfinition de celle de A (pas la même liste de paramètres) et la ligne 12 essaie de créer une instance de classe abstraite.
- (b) (2 pts) correction : 1) remplacer a = unA par super(unA) (ou supprimer cette affectation) 2) supprimer le paramètre de m, remplacer void par A et faire un retour de type B ou null.

Exercice 3 : Gestion des exceptions [15 points]

1. Le programme affiche :

```
Parcel cannot be sent : Suspicious
@@@@@
Item(4.0,2.5)Item(5.0,3.0)Item(9.0,5.5)
Item(5.0,3.0)Item(9.0,5.5)
Item(9.0,5.5)
Sent
```

Justification : Le premier paquet envoyé en ligne 85 est vide. Il déclenche l'exception de la ligne 78 qui est rattrapée et traitée en 44 ce qui cause la première ligne d'affichage. Le paquet est vidé en ligne 88. on y ajoute trois articles. Le poids total du paquet est 18 ce qui est trop lourd et lance l'exception de la ligne 75 qui est rattrapée par la ligne 42. Ceci supprime le premier article du paquet. La poste retente un envoi mais le paquet reste trop lourd. La même exception est à nouveau lancée en 75 et rattrapée en 42. Lorsque le paquet ne contient plus que l'article de poids 9, il peut être envoyé car suffisamment léger. La ligne 41 s'exécute et la méthode `sendTo` se termine sans passer par les blocs `catch`.

2. Le programme affiche :

```
Parcel cannot be sent : Suspicious
@@@@@
Item(4.0,2.5)Item(5.0,3.0)Item(10.0,5.5)
Item(5.0,3.0)Item(10.0,5.5)
Item(10.0,5.5)
```

```
Parcel cannot be sent : Suspicious
```

Justification : même explication qu'avant sauf que le paquet se vide complètement car il n'est jamais assez léger. C'est alors l'exception due au fait qu'il est vide qui est lancée.

3. non, car la méthode `accept` lance aussi une `Exception` en ligne 78 qui n'est alors ni rattrapée ni déclarée. Comme il s'agit d'une «checked exception» le programme ne compile pas.
4. La ligne 39 lance une `NullPointerException` rattrapée par la ligne 42 qui cause le lancement d'une `IndexOutOfBoundsException` qui n'est jamais rattrapée. Le programme s'arrête en affichant la «stack trace» de cette exception.

BARÈME :

1. 4 points pour les affichages et 2 pour les explications (si les explications sont complètes ajouter 1 point bonus)
2. 4 points dont 2 pour la nouvelle ligne d'affichage et 2 pour les explications.
3. 2.5 points pour une explication voisine de celle du corrigé
4. 2.5 points pour une explication voisine de celle du corrigé

Exercice 4 : Déroulement de programme [23 points]

Le programme affiche :

```
C1::C
C2::C
C3::C
Nothing with C1::C
In a world of 6
A checks C2::C
nil
```

Justifications :

- la ligne 71 crée 3 objets de types `C` qu'il est licite de stocker dans un tableau `Named[]` car `C` implémente l'interface `Named`
- les lignes 75 et 76 affichent le nom de ces objets au moyen de la méthode polymorphique `getName()`. Cette dernière applique aussi le polymorphisme de la méthode `prefix` ce qui explique les noms différents associés à chaque objet.
- l'attribut statique de la ligne 5 compte le nombre d'instance créée de type `C`
- le programme met en oeuvre une interaction deux à deux entre ces objets en utilisant le schéma de conception «visitor pattern» vu en projet. Les interface `I1` et `I2` ainsi que la classe imbriquée `H` jouent le rôle des gestionnaires d'interaction. La classe `A` joue un rôle analogue à un `Interactor`. Les objets de la hiérarchie `C` jouent le rôle des `Interactable`.
- `A` n'interagit de façon spécifique qu'avec les `C2` et il n'y a pas de définition par défaut pour l'interaction avec `C3`. Ceci explique que :
 - lorsqu'il interagit avec `C1` c'est l'interaction par défaut prévue dans `I2` qui a lieu
 - lorsqu'il interagit avec `C2` c'est l'interaction spécifique de `H` qui a lieu (l'attribut statique de la ligne 5 compte le nombre d'instance créée de type `C`; il y en a 6 de créées au moment de l'exécution de la méthode de la ligne 61);
 - et lorsqu'il interagit avec `C3` il n'y a pas de schéma prévu par défaut dans `I2` et donc c'est celui de `I1` qui se met en place

BARÈME :

- 2 par ligne d'affichage correcte pour les 3 premières
- 3 par affichage correct de chaque cas (`C1`, `C2` et `C3`)
- Si une ligne d'affichage est incorrecte ou absente on donne la moitié des points si l'explication correspondante est correcte (démontre que le mécanisme est compris)

Les explications sont essentiellement prise en compte pour donner des points en cas d'affichage incorrect. Un bonus allant jusqu'à 3 points sera attribué si la justification est soignée (et voisine de celle du corrigé), que l'affichage soit correct ou pas.