ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours de programmation aux sections IN/SC
Sam J.

INTRODUCTION A LA PROGRAMMATION

Corrigé de l'examen final

Exercice 1 : Conception OO et programmation [45 points]

Correction : voir le fichier `Simulation.java`.

BARÈME : Le détail du barème pour la conception est donné dans la fiche de correction. Si la conception est trop différente mais tient la route, la rubrique "Points ajoutés (autres conception ou bonus)" comptabilise les points donnés (à la place des points de détails de chaque sous-rubrique).
Il y a une rubrique "Bonus globaux divers" et "Malus globaux divers" pour comptabiliser des points à ajouter ou à retrancher globalement (usage de l'annotation `@Override` (+1), commentaires soignés etc.).

IMPORTANT : la solution attendue contient des éléments répétitifs (comme par exemple le codage de certains getters). Une description adéquate raccourcie en francais, à la place du code exact, sera accepté comme correct. Par exemple : «les getters pour les autres attributs se codent de façon analogue», après avoir codé une entête de getter proprement donnera tous les points pour les getters.
Par ailleurs les typos suivantes ont été notées sur le tableau pendant l'examen :

```
Annexe:
Speed -> double (on acceptera les deux)
DEFAULT_CENTER est static
Exercice 1 Question 2:
getWindValue() -> getWindData() (pareil pour la
seconde méthode
```

Exercice 2 : Concepts de base[38 points]

1. [**6 points**]
   (a) (`0x1 << pos`) est utilisé pour créer un masque dans lequel le bit à la position pos est activé. Par exemple, (`0x1 << 3) = 0b1000` en binaire. L'exécution d'un `&` avec ce masque et une valeur cible produira une valeur non nulle (2^pos) si la valeur a ce bit activé, et 0 sinon. La dernières instruction convertit cet entier en booléen.

   (b) La création du masque est analogue à la partie (a). Ensuite, si <verb+m+ est défini, un "bitwise ou" (`|`) avec le masque mettra le bit cible à 1, tout en laissant les autres bits tels quels (en raison de `1 | x = 1` et `0 | x = x`). Sinon, le masque est inversé avec (`~`) de sorte que seul le bit cible soit à zéro et que tous les autres deviennent à un. Ensuite, un "bitwise et" (`&`) avec ce masque mettra le bit cible à zéro, tout en laissant tous les autres tels quels (en raison de `1 & x = x` et `0 & x = 0`).

   (c) le programme affiche :

```
true
false
Original value: 12
Modified value: 8
Original value: -12
Modified value: -16
```

   Détail barème : donné directement dans la fiche de correction.

2. [**6 points**]
   (a) Le programme affiche :

```
colugo
hyrax
```

   Justification : il n'existe que le passage par valeur en Java. La ligne 14 est sans egget sur l'objet `w1` de `main` (`w2` est affectée à une copie de `w1`). Par contre l'objet référencé par `w2` peut être modifié.

   (b) Le programme affiche :

```
hyrax
hyrax
```

   Justification : cette affectation fait que `w1` et `w2` référencent le même objet (celui pointant sur la chaîne `"hyrax"`.

   (c) Le programme affiche :

```
colugo
colugo
```

   Justification : la paramètre se la méthode `setW` masque l'attribut de la classe `W` (le paramètre est affecté à lui-même). `setW` n'a donc plus d'effet sur l'attribut et les deux objets `w1` et `w2` gardent leurs valeurs initiales.

3. [**12.5 points**]
   (a) Le programme affiche :

```
A
B
ABC
```

Justification : Lorsque `new C()` est appelé, les constructeurs par défaut de `A` et `B` sont appelés dans l'ordre, ce qui affiche `A` puis `B`. Ensuite, `System.out.println` utilise la définition `toString()` de `C`, qui revient à la concaténation `a + b + c` du fait que `super` appelle la méthode de la classe homonyme qui lui est directement supérieure, et ceci affichera `"ABC"`.

(b) Incorrect. Ceci causera une récursion infinie (la méthode `toSTring` s'invoquant elle-même).

(c) Incorrect. Il n'y aura plus de constructeur par défaut dans `B`, or celui-ci est nécessaire au constructeur par défaut par défaut de `C`.

(d) Correct. Il existe une méthode `toString` par défaut héritée de `Object` et ce serait cette méthode qui serait employée pour les objets de type `B`.

(e) Incorrect. Le constructeur s'invoquerait lui même récursivement.

(f) Correct. Dans ce cas, le constructeur de la ligne 9 sera appelé en premier, réalisant l'affectation `b = "b" ;` puis `System.out.println(b)` sera appelé, affichant `b`.

(g) Correct. `super()` sera appelé par défaut même si nous ne l'écrivons pas explicitement.

(h) Incorrect. `A` n'a aucun constructeur prenant en paramètre une `String`.

Détail barème : donné directement dans la fiche de correction.

4. **[13.5 points]**

(a) Le programme affiche :

```
The content of the shelf is:
book::The old man and the waves, 250 pages
dvd::the-gleaming

The Shelf has : 200 items
The Shelf price is : 0.0
```

Justification : Comme `getAllItems()` renvoie le tableau complet, sa longueur est donne sa capacité et non pas le nombre d'objets effectivement stockés. `PriceCalculator.price(X)` ne fonctionne pas comme prévu et tous les appels de `price(X)` à l'intérieur de `price(Shelf)` se résoudront en `ShelfItem`, produisant 0.

(b) Le fait de placer `getPages()` dans `ShelfItem` oblige les éléments qui n'ont pas de pages, comme les `DVD`, à avoir une implémentation de cette méthode. L'utilisation d'un nombre «magique» tel que `-1` pour marquer la non-existence est obscure et constitue une abstraction peu fiable. Il serait préférable de le supprimer et de ne laisser `getPages()` que dans `Book`. De plus, l'ajout d'une méthode `String describe()` à chaque `ShelfItem` leur permettrait de gérer leur propre description, sans avoir à rendre apparent les détails de `Shelf` Si nous voulons éventuellement ajouter d'autres types d'éléments qui pourraient avoir des pages, il serait intéressant de placer `getPages()` dans une interface `HasPages`.

(c) `getAllItems()` renvoie l'ensemble du tableau au maximum de sa capacité, ce qui est peu pratique et constitue une grosse faille d'encapsulation (le contenu de la bibliothèque peut être modifié sans passer par son API publique). Plusieurs options sont envisageables dont celle-ci : ne plus fournir cette méthode et définir une API restreinte qui retourne le prix des éléments stockés dans la bibliothèque et son nombre d'objets réel.

(d) Le calcul du prix devrait se faire au moyen d'une méthode polymorphique, abstraite dans <verb+ShelfItem+, proprement redéfinie dans les sous-classes et utilisée dans une méthode de calcul de prix placée dans `Shelf`.

Détail barème : donné directement dans la fiche de correction.

Exercice 3 : Déroulement de programme [15 points]

Le programme affiche :

```
Generating ID: 6
Generating ID: 8
Generating ID: 10
Hello Bob (user)
Delighted to see you alice#8
Delighted to see you charlotte#10
```

Points importants de la justification :

1. (1.5 pts) L'instruction de la ligne 91 crée un objet de type `Platform` qui est une collection de hétérogène de `User` pouvant être concrètement soit des `FormalUser` soit des `InformlUser`. Les 3 lignes suivantes ajoutent cette collection un `InformallUser` (Bob) et deux `FormalUser` (Alice et Charlotte).
2. (1.5 pts) Lors de la construction du `InFormalUser`, le constructeur de la ligne 60 est appelé, il appelle à son tour celui de la ligne 48. Ce dernier va initialiser l'attribut `User.name` à la `"User`, l'attribut `InformalUser.name` à `"Bob"` et l'attribut `username` à `"user"`.
3. (2 pts) L'identificateur `id` va être initialisé au moyen de la méthode de la ligne 25 appelée sur l'instance de `SequentialIdGenerator` créée par le constructeur de la ligne 21 qui appelle celui de la ligne 19 en lui passant en paramètre la valeur `DefaultIdGenerator.START_ID + START_ID`. Il y a deux constantes statiques `START_ID`, la première dans l'expression provient de l'interface `DefaultGenerator` et vaut 0 la seconde de la classe `SequentialIdGenerator` et vaut 5. L'attribut `nextId` sera affiché comme valant 6 par la ligne 26 mais sera augmenté de 1 à la ligne suivante (vaudra 7).
4. (2pts) la construction des `FormalUser` se fait selon un schéma analogue : l'instance `SequentialIdGenerator` est la même que pour le `FormalUser` précédent, ce qui explique que l'attribut `nextId` continue d'être incrémenté (il passe de 7 à 8 lors de l'exécution de la ligne 26 pour "Alice").
5. (2pts) La ligne 95 appelle la méthode `welcomeUsers` de la ligne 83 qui invoque la méthode polymorphique `acceptSalutation` sur chacun de `User. Cette méthode va appeler la méthode de la ligne 65 ` en tant que `Saluter`.
6. (2pts) l'exécution de la ligne 65 provoque l'appel de la méthode 31 (définition par défaut de la méthode `salute` dans l'interface `Saluter` implémentée par la `Platform`) et qui prend en paramètre le `InformalUser "Bob"`. Ceci retournera la chaîne `"Hello"` concaténée avec le retour de l'appel à `toString` sur le `InformalUser` et qui retourne `"Bob (user)"`.
7. (2pts) le schéma d'exécution est similaire pour les `FormalUser`, sauf que la méthode `toString` est résolue dans `User` (à la ligne 54) car non redéfninie dans `FormalUser`.

BARÈME : Pour l'affichage :

- 2 points par ligne pour les 3 premières et 3 par ligne pour les 3 suivantes.
- Pour la justification 15 points (voir les points par éléments d'explication suggéré dans le corrigé ci-dessus). Si les explications sont plus verbeuses et/ou très différentes dans le forme du corrigé, donnez les points au pro rata des lignes de code (bien) expliquées.
- un bonus allant jusqu'à 1.5 si les affichages sont correcte et les explications soignées.

Attention au mode de calcul : on peut avoir quasiment tous les points sans donner l'affichage explicitement si on a donné des explications complètement correctes. On peut avoir presque tous les points s'il y a eu juste l'affichage mais pas les justifications.