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 [56 points]

Correction : voir le fichier `Simulation.java`.
See the file `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 additionnels ou à retrancher globalement (usage de l'annotation `@Override` (+1), commentaires soignés etc.).
EN : The details of the grading schema for the conception part are given in the exercice1.ods correction form. If the conception is too different but still makes sense, fill the heading "Added points (other design or bonus)" instead of the detail points in each subheading. There is a section "Bonus globaux divers" and "Malus globaux divers" to count points or to deduct globally (use of the annotation `@Override` (+1), neat comments etc.).

Exercice 2 : Concepts de base[35 points]

1. [**5 points**]
   (a) Le programme affiche

   ```
   a better day ?
   ```

   Justification : Le constructeur appelé à la ligne 15 appelle celui de la ligne 10 qui appelle à son tour celui de la ligne 9. Le constructeur de la ligne 9 appelle implicitement celui de la ligne 3 qui initialise l'attribut `a` à `"better"`. L'exécution de la ligne 10 se poursuit et initialise l'attrbut `b` à `"day ?"`. La ligne 16 appelle implicitement la méthode de la ligne 11 qui appelle celle de la ligne 5.

   (b) Non, car il est appelé par celui de la ligne 10.

   (c) Non, car on ne peut pas restreindre les droits lors de la redéfinition de méthodes et la méthode `toString` héritée de plus haut est `public`.

   (d) Non, car il y aurait un appel récursif infini au constructeur de la ligne 10.

   (e) Non, car il n'existe pas de constructeur de `A` prenant en paramètre une `String`.

   Détail barème : donné directement dans la fiche de correction.

2. [**6 points**] Une solution possible :
   A) `package1;` et `public m2(){ }`
   B) `package1;` et `private m2(){ }` (le nom du paquetage peut être différent)
   C) `package1;` et `protected m3(){ }`

   Détail barème : donné directement dans la fiche de correction.

3. [**12.5 points**] *Note : les justifications n'étaient pas demandées*
   (a) Incorrect. (il manque le `abstract` et il n'y a pas redéfinition de `m`)
   (b) Incorrect. (une classe peut implémenter mais pas étendre une interface)
   (c) Correct.
   (d) Incorrect. (une classe n'étend pas une interface)
   (e) Correct.
   (f) Incorrect. (une interface n'implémente pas une interface)
   (g) Correct.
   (h) Correct.
   (i) Correct.
   (j) Correct.

   Détail barème : donné directement dans la fiche de correction.

4. [**10.5 points**]
   (a) Le programme affiche :

   ```
   7 5
   4
   ```

(b) Oui. le programme fourni ne construit aucune instance de `Collector` (et une classe peut être abstraite même si elle ne contient pas de méthode abstraite).

(c) Non car il existe déjà une méthode homonyme et que le type de retour ne fait pas partie de la signature (ce ne serait pas une surcharge licite).

(d) Oui car on peut accéder à un membre statique au travers d'une instance (et que les classes sont dans le paquetage par défaut)

(e) La méthode `add` de la ligne 24 est une surcharge de celle de la ligne 17.

(f) La méthode `add` de la ligne 17 est une redéfinition de celle de la ligne 3.

(g) Il y a en fait deux réponses acceptables pour cette question :
- la réponse initialement attendue : parceque la classe `Collector` n'a pas de méthode `add` à deux paramètres ;
- la réponse avancée (et de base non prévue) parceque la méthode possible à deux paramètres héritée de `ArrayList` exige ici un `Integer` en second paramètre.

Il faut changer l'instruction en :

```
(Garbage)tray.add(2, true);
```

Détail barème : donné directement dans la fiche de correction.

Exercice 3 : Déroulement de programme [15 points]

Le programme affiche :

```
Car(Alice's car)
GasEngine(500.0,6000.0)
Chassis(1000.0)
performance : 4.0

Car(Bob's car)
GasEngine(400.0,2000.0)
ElectricEngine(100.0,400.0)
Chassis(1500.0)
performance : 3.0
Alice's car is faster
```

Points importants de la justification :

1. (1 pt) La performance d'une voiture est sa puissance (paramétrée par une valeur `rpm`) divisée par son poids
2. (1 pt) son poids est la somme du poids de ses moteurs auquel s'ajoute le poid du chassis
3. (2pt) le chassis de la voiture de Alice est contruit par le constructeur de la ligne 74 qui appelle celui de la ligne 73. `Chassis` est une classe imbriquée de `Car` c'est la méthode `addChassis` de `Car` qui en construit une instance. Le poids du chassis de la voiture de Alice vaut 1000.
4. (3pt) la voiture de Alice a un seul moteur de type `GasEngine`. la ligne 85 appelle le constructeur de la ligne 17 qui appelle celui de la ligne 18 avec les paramètres w=500, p=300 et rpm= `PowerProduct.POWER*20` (= 8000) ; ceci donne un poids de 500 à ce moteur et une puissance de : 300 * `PowerProduct.super.power(8000)`(=8000) (appel à la méthode de la ligne 3) divisé par `PowerProduct.POWER` (=400) ce qui donne 300 * 8000/400 = 6000.

5. (1pt) La performance de la voiture de Alice est sa puissance (6000) divisée par son poids (1500) ce qui donne 4.
6. (3pt) De façon analogue, la voiture de Bob a un poids de 400 + 100 + 1500 = 2000. La puissance de son `GasEngine` vaut 200 * `PowerProduct.super.power(4000)`(=4000) ; ce qui donne 200 * 4000/400 = 2000. La méthode `power(double rpm)` lorsqu'appelée sur `ElectricEngine` fait appel à la méthode de la ligne 3 qui retourne 4000 (la puissance affichée est la valeur de l'attribut `maxPower` qui vaut 400, ce n'est pas la valeur de la puissance parametrée par les rpm). La somme de la puissance des deux moteurs est donc 2000 + 4000 = 6000.
7. (1) la performance de la voiture de Bob est sa puissance (6000) divisée par son poids (2000) ce qui donne 3.

BARÈME : Pour l'affichage :

— 1.5 point par ligne d'affichage correcte (sauf les deux premières de chaque "voiture", sauf les premières Car(..) qui valent 1 point
— Pour la justification 12 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 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.