Cégep André-Laurendeau

420-235-AL (Hiver 2024)

Travail pratique 1

Classes, objets, constructeurs, copies, etc.

Échéance : Mardi 5 mars à 23h59, ou tel qu'indiqué sur Léa

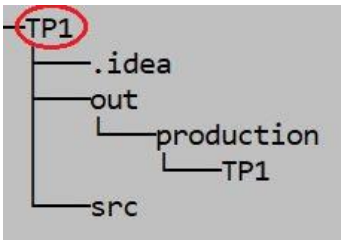
David Giasson; Michel Généreux

Objectif

Utiliser les techniques de programmation orientée objet vues dans les deux premiers modules du cours (classes et objets, encapsulation, constructeurs, accesseurs et mutateurs, comparaison et copie d'objets) pour concevoir les classes et méthodes répondants aux exigences d'un besoin logiciel.

Directives

- Ce travail peut être réalisé **seul ou en équipe de deux :**
  - Si le travail est fait en équipe, indiquez le nom des deux coéquipiers en commentaire en haut de chacun des fichiers de code source que vous remettrez.
- Une fois terminé, le travail doit être remis sur Léa :
  - Compressez le répertoire qui contient le projet IntelliJ pour votre code, c'est-à-dire le répertoire parent du dossier nommé « .idea ». Dans l'exemple suivant, on compresserait donc le répertoire TP1 (encerclé) :



  - Déposez ce fichier .zip sur Léa (une seule remise par équipe)
  - Pour vérifier que la remise a bien fonctionné :
    - Téléchargez votre fichier .zip depuis Léa
    - Décompressez le fichier .zip
    - Ouvrez le projet dans IntelliJ et assurez-vous que tout fonctionne correctement
- Le travail doit être remis au plus tard à la date officielle indiquée sur Léa. Après cette date, 10% de pénalité sera enlevé par jour de retard.
- Toute forme de plagiat entrainera automatiquement la note zéro (0) ainsi qu'un rapport officiel à votre dossier. N'oubliez-pas d'indiquer vos références si vous utilisez des extraits de code trouvés sur internet ou provenant d'une autre personne.
- La qualité du français est également importante. Jusqu'à 10% de la note finale de votre travail pourrait être retirée pour cause d'un mauvais usage du français.

Critères d'évaluation

- Modélisation des classes demandées
- Fonctionnement correct du programme
- Qualité du code (noms des méthodes et variables, formatage, lisibilité, etc.)

Mise en contexte

Vous souhaitez créer un logiciel pour vous aider à planifier le « build » de votre prochain PC, un peu à la manière de *PCPartPicker*. Étant donné que ce programme sera principalement utilisé par vous-même et d'autres étudiants de la technique, il n'a pas besoin d'interface graphique ni d'outils complexes. Votre tâche consiste à coder la logique d'affaire pour créer et modifier des « configurations » de composants matériel. Pour vous guider, vous avez établi une série d'exigences fonctionnelles sur les pages suivantes, ainsi qu'une classe *Main* qui simulera les actions possibles de l'utilisateur.

Travail à réaliser

Préalables (5%)

Pour ce travail, un fichier de départ (Main.java) vous est fourni qui contient le code qui servira à tester les opérations décrites ci-dessous. En premier lieu, vous devez créer un nouveau projet IntelliJ pour ce TP (nommez le « TP1_VotreNom »), et y ajouter le fichier Main.java. Ensuite, créez une configuration (vous pouvez l'appeler comme vous voulez, mais « Main » est une bonne option) qui permet de lancer l'exécution de l'application à partir de ce fichier.

Classe 1 : Les composants (25% -- environ 60 lignes de code)

Un **Composant** est défini par :

- Une **catégorie** – une chaine de caractères « *final* », qui doit être formattée tout en majuscules si elle possède 3 caractères ou moins; retirez également les espaces au début et à la fin s'il y en a.
- Une **marque** et un **nom** – ces attributs peuvent être « *public* »; vous n'avez pas à formatter ces chaines ni valider qu'elles sont non-vides.
- Son **prix** – une valeur réelle de montant minimum de 0 (dans le cas où la valeur entrée serait négative).
- Un **rabais** – également une valeur réelle, qui doit être comprise entre 0.0 (0%) et 1.0 (100%).

Le **constructeur** de la classe doit permettre d'initialiser correctement tous les attributs de l'item, sauf le rabais qui est nul (0.0) par défaut. Codez les méthodes pour **obtenir** ou **modifier** le prix ou le rabais du composant, en respectant les contraintes décrites précédemment. Notez que la méthode pour obtenir le prix du composant doit appliquer automatiquement le pourcentage de rabais. Codez également une méthode permettant de **copier le composant**, ainsi qu'une autre pour le **comparer avec un autre composant** (en ignorant son prix et son rabais). Finalement, ajoutez une méthode « ***toString*()** » permettant de le convertir en chaine de caractères de la forme suivante :

```
[CPU] Intel Core i5-13600K
```

où la première partie est la catégorie du composant et sa marque et son nom viennent après.

Vous pouvez aussi ajouter n'importe quelle autre méthode ou attribut qui vous semble pertinent.

Critères de correction	Points
Attributs (types, noms[1] et modificateurs d'accès pertinents)	5
Constructeur (avec ajustement du formatage de la catégorie)	4
Méthode (ou constructeur[2]) pour copier le composant	4
Méthodes pour obtenir ou modifier le prix ou le rabais	4
Méthode pour comparer avec un autre composant	4
Méthode *toString()* pour convertir en chaine de caractères	4

Classe 2 : Les configurations (65% -- environ 120 lignes de code)

Une **Configuration** possède les attributs suivants :

- Une **description**, qui n'a pas besoin d'être formattée ni validée.
- Un **prix maximum**, qui sera considéré comme « infini » si sa valeur est définie à **0** (ou moins).
- Un **tableau de composants** (accompagné d'un **nombre de composants**).
- Une constante pour le **nombre maximum possible** de composants, soit **20** composants.

Le constructeur de la classe *Configuration* exige 3 paramètres : la **description**, le **prix maximum**, et un **tableau initial de composants** qui devront être copiés dans le tableau de composants de la configuration.

La configuration propose également les méthodes suivantes :

- *Configuration copier()* → Retourne une copie profonde de la configuration (y compris une copie de chacun de ses composants); Dans la copie, on ajoute le suffixe « (copie) » à la fin de la description. Vous pouvez également créer un constructeur de copie et l'utiliser dans votre méthode *copier* pour obtenir un point bonus.

- *void afficherTout()* → Affiche la description de la configuration et son prix maximum entre parenthèses (ou « montant illimité » si le maximum est 0), puis affiche la liste de tous ses composants suivi de leur prix avant taxes, de la manière suivante :

```
Build Intel Gen13 (max 1250,00$) :
   1: [CPU] Intel Core i5-13600K (330,00$)
   2: [Carte mère] Asus ROG Strix B760 (200,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (90,00$)
   4: [GPU] Asus RTX 4060 (460,00$)
```

[1] Suggestion : bien que permis en Java, évitez les identificateurs avec diacritiques (accents).
[2] Vous pouvez créer un constructeur de copie et l'utiliser dans votre méthode « copier() » afin d'obtenir un point bonus.

- *getX()* et *setX()* → Ajoutez les méthodes *get* et *set* pour la description et le prix maximum, ainsi qu'une méthode *get* pour le nombre de composants présents dans la configuration.

- *double calculerTotal(double taxe)* → Retourne le total du prix de tous les composants de la configuration, et ajoute le pourcentage de taxe passé en paramètre (entre 0.0 et 1.0).

- *Composant rechercher(String categorie)* → Chercher et retourne le composant ayant la catégorie passée en paramètre; S'il n'y en a pas dans la configuration, retourne *null*.

- *boolean ajouter(Composant composant)* → Ajoute un composant à la configuration si :
  - Le tableau de composants n'est pas déjà plein;
  - Il n'y a pas déjà un autre composant ayant la même catégorie dans le tableau;
  - L'ajout de ce composant ne fera pas dépasser le prix maximum (avant taxes), sauf si le prix maximum est illimité (0).

  Si l'ajout est réussi, la méthode retourne *true*, et *false* sinon.

- *boolean retirer(Composant composant)* → Retire un composant de la configuration s'il s'y trouve; Le composant visé peut avoir un prix ou un rabais différent, car seules sa catégorie, sa marque et son nom comptent. Si le retrait est réussi, la méthode retourne *true*, et *false* sinon.

- *boolean remplacer(Composant composant)* → Remplace un composant présent dans la configuration par celui passé en paramètre s'ils ont la même catégorie. Cette méthode peut être vue comme un retrait du composant existant suivi de l'ajout du nouveau composant; De plus, les mêmes contraintes que lors d'un retrait et d'un ajout de composant s'appliquent. Si l'opération est réussie, la méthode retourne *true*, et *false* sinon.

Vous pouvez aussi ajouter n'importe quelle autre méthode ou attribut qui vous semble pertinent.

Critères de correction	**Points**
Attributs (types, noms et accesseurs correspondants)	5
Constructeur (avec copie des items initiaux dans l'inventaire)	5
Méthode *copier()*	5
Méthode *afficherTout()*	5
Méthode *calculerTotal()*	5
Méthode *rechercher()*	5
Méthode *ajouter()*	10
Méthode *retirer()*	10
Méthode *remplacer()*	10
Qualité du code (respect des normes de programmation)	5

Remise et autres consignes (5%)

Une fois terminé, remettez votre travail sur Léa dans un fichier zip contenant :

- Dossier TP1_VotreNom (racine du projet)
  - Fichier TP1_VotreNom.iml (ou portant nom similaire) pour votre **projet IntelliJ**
  - Dossier « **src** » contenant vos classes *Composant* et *Configuration* ainsi que le *Main;* Attention : le fichier « *Main* » ne devrait pas avoir été modifié (sauf pour y ajouter votre ou vos noms en haut) car lors de la correction il sera remplacé par la version de départ.
  - Les autres dossiers ne sont pas nécessaires mais peuvent être remis quand même.

Annexe : Exemple du résultat attendu

```
=== Test #1 : Affichage des configurations initiales ===

Configuration vide (montant illimité) :
Total: 0 composants pour 0,00$ (taxes incluses)

Build Intel Gen13 (max 1250,00$) :
   1: [CPU] Intel Core i5-13600K (330,00$)
   2: [Carte mère] Asus ROG Strix B760 (200,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (90,00$)
   4: [GPU] Asus RTX 4060 (460,00$)
Total: 4 composants pour 1242,00$ (taxes incluses)

Build AMD Gen5 (max 1000,00$) :
   1: [CPU] AMD Ryzen 5 5700X (260,00$)
   2: [Carte mère] MSI B550 Gaming Wifi (150,00$)
Total: 2 composants pour 471,50$ (taxes incluses)


=== Test #2 : Recherche et manipulation de composants ===

[CPU] Intel Core i5-13600K
CPU
true
[Carte mère] MSI B550 Gaming Wifi
150.0
0.0
[RAM] GSkill Trident-Z DDR5 16GB
RAM
82.5
0.25
true
null
```

```
=== Test #3 : Ajouts de composants ===

[RAM] GSkill Trident-Z DDR4 32GB ajouté à la configuration (total=545,00$)
[SSD] Western Digital SN850X 1TB ajouté à la configuration (total=645,00$)
# Rabais de 20% sur [GPU] Gigabyte RTX 4060 (nouveau prix: 320,00$)
[GPU] Gigabyte RTX 4060 ajouté à la configuration (total=965,00$)

Build AMD Gen5 (max 1000,00$) :
   1: [CPU] AMD Ryzen 5 5700X (260,00$)
   2: [Carte mère] MSI B550 Gaming Wifi (150,00$)
   3: [RAM] GSkill Trident-Z DDR4 32GB (135,00$)
   4: [SSD] Western Digital SN850X 1TB (100,00$)
   5: [GPU] Gigabyte RTX 4060 (320,00$)
Total: 5 composants pour 1109,75$ (taxes incluses)

Il y a déjà un autre composant de cette catégorie: [CPU] AMD Ryzen 5 5700X
Il y a déjà un autre composant de cette catégorie: [GPU] Gigabyte RTX 4060
L'ajout de ce composant ferait dépasser le prix maximum: [SSD] Samsung 980 Pro 2TB

=== Test #4 : Retraits de composants ===

[RAM] GSkill Trident-Z DDR4 32GB retiré de la configuration (total=830,00$)
[GPU] Gigabyte RTX 4060 retiré de la configuration (total=510,00$)
[CPU] AMD Ryzen 5 5700X retiré de la configuration (total=250,00$)

Build AMD Gen5 (max 1000,00$) :
   1: [Carte mère] MSI B550 Gaming Wifi (150,00$)
   2: [SSD] Western Digital SN850X 1TB (100,00$)
Total: 2 composants pour 287,50$ (taxes incluses)

Composant introuvable: [RAM] GSkill Trident-Z DDR4 32GB
Composant introuvable: [GPU] Gigabyte RTX 4060
Composant introuvable: [CPU] AMD Ryzen 5 5700X

=== Test #5 : Copies de configurations ===

Build AMD Gen5 (copie) (max 1000,00$) :
   1: [Carte mère] MSI B550 Gaming Wifi (150,00$)
   2: [SSD] Western Digital SN850X 1TB (100,00$)
Total: 2 composants pour 287,50$ (taxes incluses)

[CPU] AMD Ryzen 7 7800X3D ajouté à la configuration (total=750,00$)
[Carte mère] MSI B550 Gaming Wifi retiré de la configuration (total=600,00$)
[RAM] GSkill Trident-Z DDR5 16GB ajouté à la configuration (total=676,50$)

Build AMD Gen7 (max 1000,00$) :
   1: [SSD] Western Digital SN850X 1TB (87,00$)
   2: [CPU] AMD Ryzen 7 7800X3D (500,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (76,50$)
Total: 3 composants pour 763,03$ (taxes incluses)
```

```
=== Test #6 : Remplacement de composants ===

[SSD] Western Digital SN850X 1TB retiré de la configuration (total=576,50$)
[SSD] Samsung 980 Pro 2TB ajouté à la configuration (total=826,50$)
[CPU] AMD Ryzen 7 7800X3D retiré de la configuration (total=326,50$)
[CPU] AMD Ryzen 7 7800X3D ajouté à la configuration (total=676,50$)
Il n'y a pas de composant de la catégorie Carte mère
[GPU] Gigabyte RTX 4060 ajouté à la configuration (total=996,50$)
[RAM] GSkill Trident-Z DDR5 16GB retiré de la configuration (total=920,00$)
L'ajout de ce composant ferait dépasser le prix maximum: [RAM] GSkill Trident-Z
  DDR4 32GB
[RAM] GSkill Trident-Z DDR5 16GB ajouté à la configuration (total=996,50$)

Build final (max 1000,00$) :
   1: [SSD] Samsung 980 Pro 2TB (250,00$)
   2: [CPU] AMD Ryzen 7 7800X3D (350,00$)
   3: [GPU] Gigabyte RTX 4060 (320,00$)
   4: [RAM] GSkill Trident-Z DDR5 16GB (76,50$)
Total: 4 composants pour 1145,98$ (taxes incluses)


=== Test #7 : Validation finale ===

Build Intel Gen13 (max 1250,00$) :
   1: [CPU] Intel Core i5-13600K (330,00$)
   2: [Carte mère] Asus ROG Strix B760 (200,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (90,00$)
   4: [GPU] Asus RTX 4060 (460,00$)
Total: 4 composants pour 1242,00$ (taxes incluses)

Build AMD Gen5 (max 1000,00$) :
   1: [Carte mère] MSI B550 Gaming Wifi (150,00$)
   2: [SSD] Western Digital SN850X 1TB (100,00$)
Total: 2 composants pour 287,50$ (taxes incluses)

Build AMD Gen7 (max 1000,00$) :
   1: [SSD] Western Digital SN850X 1TB (87,00$)
   2: [CPU] AMD Ryzen 7 7800X3D (500,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (76,50$)
Total: 3 composants pour 763,03$ (taxes incluses)

[GPU] Gigabyte RTX 4060 retiré de la configuration (total=676,50$)
[GPU] Asus RTX 4060 ajouté à la configuration (total=1136,50$)
Build final (montant illimité) :
   1: [SSD] Samsung 980 Pro 2TB (250,00$)
   2: [CPU] AMD Ryzen 7 7800X3D (350,00$)
   3: [RAM] GSkill Trident-Z DDR5 16GB (76,50$)
   4: [GPU] Asus RTX 4060 (460,00$)
Total: 4 composants pour 1306,98$ (taxes incluses)


Process finished with exit code 0
```