Objets, délégation, structure simple, exceptions

Exercice 1 - Eclipse

À partir de maintenant, nous allons utiliser Eclipse (lancer la commande `eclipse` dans un terminal) comme environnement pour faire les TPs.

1. Créer un projet nommé TP4.
2. Modifier les propriétés du projet TP4 pour que les classes (fichiers .java) soit générées dans le répertoire `sources`.
3. Faire de même pour tous les futurs projets qui seront créés (Aller voir dans le menu Window > Preferences).
4. Vérifier que l'environnement d'exécution est bien Java-13, changer si ce n'est pas le cas.
5. Vérifier que l'environnement de compilation est bien 13, changer si ce n'est pas le cas.
6. Écrire une classe `Main` qui affiche Hello Eclipse.
7. A faire à la maison pour le compte rendu:
    1. Que fait `sysout + Ctrl + Space` dans un `main` ?
    2. Que fait `toStr + Ctrl + Space` dans une classe ?
    3. Définir un champs `foo` de type `int`, que fait `get + Ctrl + Space`, et `set + Ctrl + Space` .
    4. Dans le menu Source, comment générer un constructeur initialisant le champ `foo` ?
    5. Sélectionner le nom de la classe puis `Alt + Shift + R`, qu'obtient-on ? Même question avec le champ `foo` .
    6. Écrire `a = 2 + 3 + 4`, puis sélectionner `2 + 3` puis `Alt + Shift + L` .
    7. Écrire `new Integer(2)`, en gardant le curseur après ')', appuyer sur `Ctrl + 1`, que se passe-t-il ?
    8. Déclarer une variable `s` de type `String` et cliquer sur `String` en maintenant la touche `Ctrl` . Que se passe-t-il ?
    9. Dans la méthode `toString()`, que fait un `Ctrl + Clic` sur `super.toString()` ?
    10. Sélectionner le champs `foo`, puis `Ctrl + Shift + G`. Que se passe-t-il ?
    11. À quoi sert `Ctrl + Shift + O` ?
    12. À quoi sert `Ctrl + Shift + C` ?

    Apprenez les raccourcis que nous venons de voir, cela vous fera gagner du temps lors des TPs notés.

Exercice 2 - Caddie en amazone

Le but de cet exercice est de définir une classe pour représenter un caddie contenant un ensemble de livres (on réutilisera la classe `Book` du TP précédent).
On souhaite que chaque `ArrayShoppingCart` puisse définir un nombre maximal de livres que l'on peut stocker dans le caddie.
Nous allons écrire une classe `ArrayShoppingCart` utilisant un tableau pour stocker les livres, son constructeur ainsi qu'une méthode `add` qui permet d'ajouter un livre au caddie.
Pour tout l'exercice, vous écrirez un `main` de test dans la classe `ArrayShoppingCartTest`.

1. Est-il intéressant de stocker le nombre maximum de livres dans un champ statique ?
2. Écrire le constructeur ainsi que la méthode `add()` sachant que l'on vous demande de stocker les livres dans un tableau.
   Dans un premier temps, n'essayez pas de traiter le cas où il y a plus de livres que le nombre maximum de produits.
   Écrire, dans le `main`, un test pour ce cas précis. Que se passe-t-il ?
   Comment doit-on modifier la méthode `add` ?
3. Écrire une méthode permettant d'afficher le contenu du caddie avec le nombre de livres en en-tête, puis un livre par ligne.
   Attention à ce que votre code n'alloue pas trop d'objets!
4. Écrire une méthode `longestTitle` qui retourne le livre du caddie qui à le titre le plus long.
   On renverra `null` si il n'y a pas de titre le plus long.

Exercice 3 - Caddie Libre

L'implantation précédente obligeait le programmeur à choisir à l'avance un nombre maximal de livres à stocker dans un caddie. On souhaite enlever cette limitation en utilisant la classe `java.util.ArrayList` plutôt qu'un tableau pour stocker les livres dans le caddie.

1. Créer une classe `FreeShoppingCart`, son constructeur ainsi que la méthode `add`.
   Comment enlever le warning que le compilateur signale sur la méthode `ArrayList.add` ?
2. On souhaite maintenant écrire la méthode `longestTitle`. Utiliser pour cela un indice parcourant la liste et la méthode `ArrayList.get` pour obtenir les livres.
3. Écrire une nouvelle version de la méthode `longestTitle` et utilisant la méthode `ArrayList.iterator` puis les méthodes `Iterator.hasNext()` et `Iterator.next` sur l'itérateur renvoyé.
4. Écrire une troisième version de la méthode `longestTitle` en utilisant une boucle `foreach` (le `for(:)` de Java).
   Comment le compilateur compile-t-il une boucle `foreach` sur une collection ?
   Utiliser la commande `javap` si vous ne savez pas !

   ```
   javap -c MonMain.class
   ```

   Écrire dans la méthode `main` d'une classe `Test` une boucle `foreach` sur un tableau (par exemple, celui des arguments du `main`).
   Comparer les résultats avec ceux de la question précédente. Que pouvez-vous en conclure ?
5. Écrire la méthode `removeFirstOccurence` qui supprime la première occurrence d'un livre dans le caddie. Qu'elle est la complexité de cette méthode, dans le pire cas ?
6. Notez qu'il existe une méthode `remove` dans la classe `java.util.Iterator`, comment l'utiliser pour écrire `removeFirstOccurence` ?
   Quel est l'intérêt par rapport au code précédent ?
   Écrire le code correspondant.
   Et maintenant, quelle est la complexité dans le pire cas ?
7. Conclure en indiquant dans quel cas doit-on utiliser la boucle `foreach` et dans quel cas doit-on utiliser un itérateur explicitement sur une collection.
