Université Bordeaux **4TYE707U Programmation orientée objet**
Master Miage semestre 7 **Parcours Informatique de Gestion – 2022-2023**

Travaux Dirigés 1 – La bonne formation des objets – encapsulation

---

1. **Eclipse**

   Pour l'ensemble des travaux en salle machine, `Eclipse` est l'environnement de développement préconisé et `Java` le langage de programmation utilisé.

   Nous vous invitons dès à présent à vous référer à la documentation sur le site officiel `www.eclipse.org` mais également aux nombreux tutoriels disponibles sur Internet pour prendre en main cet outil. Pour configurer correctement votre environnement pour ce premier TD, vous devez :

   (a) Lancer Eclipse (version Oxygen) ;

   (b) Indiquer comme *workspace* le répertoire dans lequel vous travaillerez. Il est recommandé d'utiliser différents espaces de travail selon la matière. Par exemple, utilisez `~/espaces/travail/POO/workspace`.

   (c) Créer un projet td1. Vérifier que le JRE utilisé est bien la version 8.

   (d) Y inclure un paquetage pour le projet. Par exemple `fr.ubordeaux.miage.s7.poo.td1`

   (e) Enfin y créer des classes.

2. **Premières classes**

   Dans le but de programmer un site de commerce en ligne, nous allons créer un ensemble d'objets qui vont s'organiser dans un projet complet :

   — Des produits
   — Un catalogue de produits
   — Des clients
   — Des paniers

   Nous allons donc écrire les classes suivantes `Product`, `Catalog`, `Customer`, `Basket`.

   Ajouter les attributs et méthodes qui semblent nécessaires à leur bonne gestion. On essaiera d'être cohérent et d'apporter l'ensemble des attributs et méthodes nécessaires et suffisants pour chaque classe.

   — `Product` est composé du nom et du prix
   — `Catalog` est composé d'un tableau des produits en stock et d'un tableau des quantités des produits en stock.
     Nous nous limiterons à des tableaux de taille prédéfinie par une constante de la classe `Catalog`
     Remarque : puisqu'il s'agit d'un tableau, chaque produit pourrait avoir un numéro et ce numéro sert d'indice pour les deux tableaux.
   — `Customer` est composé du prénom, nom, adresse, email et numéro de téléphone du client.
   — `Basket` est composé d'un tableau indiquant les produits commandés. Il s'agit d'un tableau d'entiers ayant pour taille le nombre de produits du catalogue et stockant la quantité commandée de chacun des produits (par défaut, 0).
     `Basket` *agrège* un `Catalog` et un `Customer`.

Écrire une classe `Main` dans laquelle vous créerez la méthode principale `main`. Ce sera le premier envoi de message directement lancé à partir de la machine virtuelle Java.

```
public static void main(String[] args) {
// A completer
}
```

Pour construire le projet, dans cette méthode `main`

(a) Créer un objet `Customer customer`

(b) Créer un objet `Catalog catalog`

(c) Ajouter 3 produits au catalogue en quantités variables

(d) Créer un objet `Basket` pour le client `customer` et y ajouter 2 des produits en quantités variables.

Vous observerez qu'il est possible de créer une base de données incohérente avec des paniers qui contiennent des produits indisponibles, de créer des clients différents avec exactement les mêmes attributs, de créer à tort deux fois le même produit dans le catalogue et dans un panier, etc. Le but de la suite est de contraindre le projet à la cohérence de cette base de données.

3. **Bonne formation des classes d'objets : identifiant unique**

   Par définition, deux instances différentes des objets `Product`, `Customer`, `Basket` ne doivent pas posséder le même identifiant.

   Assurez-vous de l'automatisation de ce critère de bonne formation des objets par la numérotation automatique des instances de ces objets.

   Pour cela, dans ces classes, vous distinguerez un *attribut statique* `nextId` initialisé à 0 et incrémenté à chaque création d'une nouvelle instance. Chaque instance aura un attribut `id` qui prendra la valeur de `nextId` au moment de sa création. Vous éviterez que ces attributs ne puissent être modifiés d'une manière quelconque.

4. **Bonne formation des classes d'objets : référence unique des produits**

   Vous modifierez le projet de sorte que le catalogue soit seul responsable de la création des produits ou de leur destruction. Pour être précis, les produits seront dorénavant *composés* au catalogue et seulement au catalogue.

   `Catalog` contient une méthode `int addProduct(String name, double price, int quantity)` qui crée le produit, l'ajoute au catalogue et fournit à l'objet appelant l'identifiant unique du produit ajouté.

   Les produits seront identifiables grâce à leur numéro d'identification et seront accessibles uniquement pas le message `getProductById(int key)` envoyé à `Catalog`, où `key` est l'identifiant d'un produit.

   Les paniers doivent donc invoquer cette méthode de la classe `Catalog` pour avoir accès aux produits.

5. **Bonne formation des classes d'objets : catalogue unique**

   Une seule instance de `Catalog` doit être présente pour le projet complet.

   La création d'un objet `Catalog` ne doit pouvoir se réaliser qu'une seule fois. Cela est réalisable par une référence à un objet unique nommé `instance` et accessible seulement avec le message `getInstance()`. Le constructeur de cette classe sera inaccessible aux autres objets.

   Modifier la classe `Catalog` pour obtenir cet effet.

6. **Bonne formation des classes d'objets : attributs valides**

   Les différentes informations contenues dans les objets doivent être vérifiées (par exemple l'adresse électronique du client doit être dans un format de type `xxx@xxx.xxx`). Lors de la création des objets, s'assurer que la bonne formation des attributs soit vérifiée.

   Pour cela, vous chercherez du code Java sur Internet permettant de valider une adresse électronique avec l'emploi des classes de la bibliothèque `java.util.regex` et vous modifierez le constructeur de la classe `Customer` pour que seuls les contacts valides soient créés. Dans le cas contraire, un affichage en sortie standard sera produit comme suit :

```
1     if (/* A completer */) {
2       /* A completer */
3     }
4     else {
5                     System.out.println("message d'erreur");
6     }
```

7. **Bonne formation des classes d'objets : références valides**

   Les références des objets correspondent à des objets qui existent. Par exemple deux paniers différents (qui supposent une commande imminente) ne peuvent pas faire tous deux référence au même produit disponible à un seul exemplaire dans le catalogue. Faire en sorte que l'ajout d'un produit dans un panier fasse décroitre un compteur d'occurrences de ce produit dans le catalogue, et qu'en cas d'indisponibilité le panier et le catalogue soient inchangés après l'affichage d'un message d'erreur.

8. **Encapsulation**

   Le code suivant, écrit dans une classe autre que la classe `Customer`, doit provoquer une erreur de compilation car l'attribut `secondName` (nom de famille) ne doit être accessible que depuis la classe `Customer`

```
1 Customer user = new Customer("Pierre", "Alice",
2                 new Address(5, "rue du Commerce",
3                         new Town("Bordeaux", 33000)));
4 System.out.println(user.secondName);
```

   (a) Qu'est-ce qui, dans votre code, permet d'assurer ce comportement ?

   (b) Quelle est la façon correcte d'obtenir l'information (ici, le nom de famille) ? Écrire le code correspondant.

(c) Quel avantage trouvez-vous à cette façon ?

(d) Reprendre votre code de telle sorte que les attributs de chacune de vos classes ne puissent être accédés (lus ou écrits) que par les méthodes de cette classe.

9. **Encapsulation (suite)** Supposons, comme vu précédemment, que la classe `Catalog` compose les produits et possède une méthode `Product getProductById(int key)` qui retourne le produit d'identifiant `key`.
   Supposons que la classe `Product` possède une méthode `void setPrice(double price)` qui modifie le prix d'un produit.

   (a) Ecrire du code montrant qu'il est alors possible qu'un panier modifie le prix d'un produit.

   (b) Et donc que faut-il faire pour qu'il soit impossible pour un panier de modifier le nom et le prix d'un objet Product ?