COLLÈGE MONTMORENCY

# Atelier #4 : L'héritage (partie 2)

Travail préparatoire pour le TP3.

**Exercice1**

Dans cet exercice, vous allez **reprendre le programme de l'atelier4-partie1** et apporter les modifications demandées ici-bas.

**La classe Produit et ses dérivés :**

1. Ajoutez la méthode ci-dessous permettant de lire les renseignements d'un produit dans la classe `Produit`.

   Cette méthode doit faire appel aux méthodes qui permettent de lire tous les attributs d'un produit: le numéro, la description et le prix.

```
public void lireRenseignements(Scanner clavier) {
   System.out.println("Entrez les renseignements du produit");
   lireNumero(clavier);
   lireDescription(clavier);
   lirePrix(clavier);
}
```

2. Vous devez ajouter une méthode `lireRenseignements(Scanner clavier)` dans les classes qui dérivent de `Produit`. Cette méthode doit bien entendu être **redéfinie (@Overrice)** pour lire les renseignements spécifiques de chaque type de produit. Par exemple, dans la classe `ProduitBase`, la méthode doit lire les renseignements suivants: le numéro, la description, le prix et la catégorie. Utilisez `super.lireRenseignements()` quand c'est possible.

**Modifiez la classe `GestionEntrepot` comme ici-bas:**

3. Modifiez le corps de la méthode `ajouterProduit(ListeProduits listeProd)` de sorte à pouvoir ajouter différents types de produits saisis à partir du clavier dans la liste de produits `listeProd`. Cette méthode proposera à l'utilisateur un menu de choix de code du produit à ajouter comme ci-dessous. A la fin de l'ajout, on demande à l'utilisateur s'il veut ajouter un autre produit si la liste n'est pas pleine.

```
Codes des produits:
B: produit de Base
T: produit taxable
S: produit taxable TPS
Entrez le code: B
```

   Voici quelques indications pour coder la méthode:

- Déclarez 3 constantes de type `char` pour stocker les valeurs des codes «B», «T» et «S».
- Déclarez une variable de type `Produit` soit `produit`;
- Affichez le menu et lire le code du produit, soit `codeLu`.
- Utilisez la méthode statique `Character.toUpperCase(codeLu)` pour convertir le code lu en majuscule avant de le comparer aux constantes qui stockent les codes des produits.
    - Si le code lu est «B», instanciez un objet de la classe `ProduitBase`:
        - `produit = new ProduitBase()`
    - Si le code lu est «T», instanciez un objet de la classe `ProduitTaxable`
        - …
    - Note : vous pouvez utiliser une instruction `switch… case`
- Appelez ensuite la méthode `lireRenseignements()` pour lire les informations du produit. Java utilisera la méthode de la bonne classe en fonction du type effectif de la variable `produit` (c'est le polymorphisme).
    - `produit.lireRenseignements(clavier);`

L'utilisateur pourra effectuer plusieurs ajouts de produits si la liste des produits n'est pas pleine. Si la liste est pleine, aucune demande de lecture n'est demandée et le message suivant sera affiché: «`Le nombre maximum de produits est atteint`».
Voici des exemples de traces d'exécution.

Trace 1

```
Codes des produits:
B: produit de Base
T: produit taxable
S: produit taxable TPS
Entrez le code:  b
Entrez les renseignements du produit
Entrez le numéro du produit
10
Entrez la description du produit
Lait 1 litre
Entrez le prix du produit
2,25
Entrez la catégorie du produit de base, exemple Fruits et légumes, viandes...
Produits laitiers

Voulez-vous saisir un autre produit? o
Code des produits:
B: produit de Base
T: produit taxable
S: produit taxable TPS
Entrez le code: T
…
```

**Important!** pour tester cette méthode ainsi que les méthodes `lireRenseignements()` des sous-classes de `Produit`, vous devez saisir 3 produis de types différents et afficher la liste des produits pour vérifier que les produits sont affichés correctement.

**Trace2**: Cas où après l'ajout d'un produit, la liste devient pleine.
Avant de faire ce test, il faut fixer la taille de la liste à **4** dans la méthode `main()` de la classe `GestionEntrepot`, puis appelez la méthode `initialiserEntrepot()` qui insère 3 produits. Il reste alors une seule case dans le tableau.

```
Codes des produits:
```

```
B: produit de Base
T: produit taxable
S: produit taxable TPS
Entrez le code:  t
Entrez le numéro du produit
20
Entrez la description du produit
Stylo noir Bic
Entrez le prix du produit
1,45
Le nombre maximum de produits est atteint
```

**Trace3**: Cas où l'ajout d'un produit n'est pas possible puisque la liste est pleine
Avant de faire ce test, il faut fixer la taille de la liste à **3** dans la méthode `main()` de la classe `GestionEntrepot`, puis appelez la méthode `initialiserEntrepot()` **qui insère 3 produits. La liste devient alors pleine.** L'affichage serait comme suit:

```
Le nombre maximum de produits est atteint.
```

4. Ajoutez une méthode `public static void afficherProduit(ListeProduits listeProd)` qui affiche la liste des produits contenus dans la liste . Si la liste est vide, le message suivant devait être affiché à la console «`Aucun produit à afficher`». Vous devez appeler la méthode `listerProduits()` de la classe `ListeProduits`.

5. Ajoutez la méthode ci-dessous qui permet d'appliquer un rabais reçu en paramètre pour tous les livres. Les livres sont des produits de la classe `ProduitTaxeTPS` ayant pour attribut `type` la valeur «`l`».

   ```
   public static void AppliquerRabaisLivre(ListeProduits listeProd,
   double taux)
   ```

   Les étapes du traitement sont:

   - Parcourez la liste des produits et obtenez le produit à la case d'indice `i` à l'aide de `listeProd.obtenirProduit(i)`.
   - Utilisez l'opérateur `instanceof` pour savoir si ce produit est de la classe `ProduitTaxeTPS`.
   - Si c'est le cas, convertissez ce produit en un objet de la classe (`ProduitTaxableTPS`) pour pouvoir accéder au type du produit en utilisant l'accesseur de l'attribut `type`.
   - Si ce type est «`l`», appelez la méthode `appliquerRabais()` de la classe `Produit` avec le bon argument .

   Testez cette méthode avec un appel dans la méthode main() . `AppliquerRabaisLivre(listeProd, 10);`

   Vérifiez que l'affichage du livre de la liste est comme suit:

   ```
   Description: Comment gérer son stress
   Prix: 14,18$
   Type de produit: l
   Prix incluant les taxes : 14,88$
   ```