TD2

Exercice 1 : Gestion de Compte Bancaire

Objectif : Travailler sur l'instanciation, les constructeurs et l'encapsulation.

Créer une classe **CompteBancaire** avec les attributs privés suivants :

- titulaire (chaîne de caractères)
- solde (flottant)

Implémenter les méthodes suivantes :

- Un **constructeur** qui initialise un compte bancaire avec le nom du titulaire et un solde initial de 0.
- Une méthode **deposer** qui permet d'ajouter un montant au solde du compte.
- Une méthode **retirer** qui permet de retirer un montant du solde du compte (si le solde est suffisant).
- Des méthodes **getSolde** et **getTitulaire** pour accéder aux attributs privés.

Questions :

- Pourquoi l'encapsulation est-elle importante dans la gestion d'un compte bancaire ?
- Comment le constructeur est-il utilisé pour initialiser correctement les objets CompteBancaire ?

Exercice 2 : Surcharge de Méthodes pour les Calculs

Objectif : Travailler sur la surcharge de méthodes.

Crée une classe **Calculatrice** avec les méthodes suivantes :

- Une méthode **addition** qui accepte deux ou trois nombres et retourne leur somme. Si un seul nombre est fourni, elle retourne ce nombre.
- Une méthode **soustraction** qui accepte deux ou trois nombres. Si deux nombres sont fournis, elle retourne la différence entre le premier et le deuxième. Si trois nombres sont fournis, elle soustrait les deux derniers au premier.

Exemple :

```
calc = Calculatrice();
System.out.println(calc.addition(10, 5));  // Retourne 15
System.out.println(calc.addition(10, 5, 2));  // Retourne 17
System.out.println(calc.soustraction(10, 5));  // Retourne 5
System.out.println(calc.soustraction(10, 5, 2));  // Retourne 3
```

Exercice 3 : Gestion des Produits

Objectif : Travailler sur l'instanciation, les constructeurs et l'encapsulation.

Créer une classe **Produit** avec les attributs privés suivants :

- nom (chaîne de caractères)
- prix (flottant)
- quantite (entier)

Implémenter les méthodes suivantes :

- Un constructeur qui initialise un produit avec un nom, un prix et une quantité.
- Une méthode **ajusterPrix** qui permet de changer le prix du produit.
- Une méthode **ajusterQuantite** qui permet de changer la quantité disponible du produit.
- Des méthodes **getNom**, **getPrix**, et **getQuantite** pour accéder aux attributs.

Questions :

- En quoi l'encapsulation aide-t-elle à contrôler les modifications sur les attributs du produit ?
- Comment le constructeur facilite-t-il l'instanciation de nouveaux produits ?

Exercice 4 : Classe pour les Vecteurs

Objectif : Travailler sur la surcharge des méthodes et l'instanciation.

Créer une classe **Vecteur** avec les attributs :

- x (flottant)
- y (flottant)

Implémenter les méthodes suivantes :

- Un constructeur qui initialise un vecteur avec deux coordonnées, x et y.
- Une méthode **ajouter** qui accepte un autre vecteur ou deux coordonnées séparées et retourne la somme des vecteurs ou de leurs coordonnées.
- Une méthode **soustraire** qui accepte un autre vecteur ou deux coordonnées séparées et retourne la différence des vecteurs ou de leurs coordonnées.

Exemple :

```
Vecteur v1 = Vecteur(2, 3);
Vecteur v2 = Vecteur(5, 7);
System.out.println(v1.ajouter(v2));  // Retourne un nouveau vecteur (7, 10)
System.out.println(v1.ajouter(1, 1));  // Retourne un nouveau vecteur (3, 4)
```

Questions :

- Comment la surcharge de la méthode ajouter permet-elle de gérer différents types d'arguments ?
- En quoi le constructeur facilite-t-il l'instanciation de nouveaux vecteurs ?

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/cnam/utc503/td2?rev=1725534453

Last update: **2024/09/05 13:07**