TRAVAUX PRATIQUES

TP1 : LE PRODUIT

Objectif : Manipulation des Classes, variables, méthodes et Objets.

Un grossiste vend des produits. Il désire mémoriser :
· le code, sous forme de chaine de caractères,
· la dénomination, sous forme de chaine de caractères,
· le prix, sous forme d'entiers.

On désire créer des méthodes permettant :

· La création du produit en passant en argument le code du produit.
· La mise à jour de la dénomination, en passant en argument une chaîne de caractères,
· La mise à jour du prix, en passant en argument un entier,
· La consultation du code, on retourne une chaîne de caractères,
· La consultation de la dénomination, on retourne une chaîne de caractères,
· La consultation du prix, on retourne un entier.

1- Définir le code Java pour la classe Produit.

2- Définir le code Java pour tester la classe Produit :

On créera deux produits :

· produit1 de type Produit ayant :
    un code = « produit100 »
    une dénonination = «crayon de papier »
    un prix = 1
· produit2 de type Produit ayant :
    un code = « produit200 »
    une dénonination = «crayon de couleur»
    un prix = 2

On affichera :
Le code et la dénomination du produit1.
Le code, le prix du produit2 et sa dénomination.

TP2 : LE VECTEUR

Objectif : Manipulation des Classes, variables, méthodes et Objets.

Créer une classe Vecteur ayant 2 variables d'instance entières x et y et une méthode publique norme() permettant de calculer la norme d'un vecteur. Utilisez cette classe pour créer des vecteurs, afficher leurs données x et y ainsi que la norme.

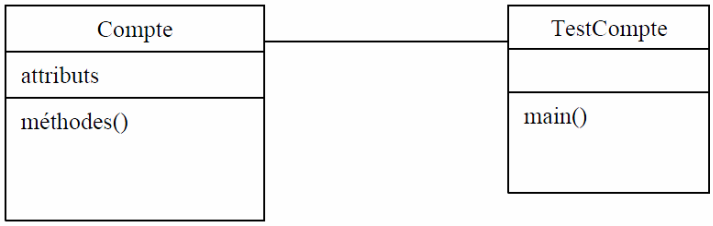
TP3 : LE COMPTE BANCAIRE

Objectif :
Manipulation des Classes, variables, méthodes et Objets.

On veut gérer des comptes bancaires et pouvoir pour un compte donné :
- Déposer un montant
- Retirer un montant

On veut connaître pour ce compte :
- Le solde, c'est-à-dire la somme des montants déposés moins la somme des montants retirés. On veut aussi connaitre à qui appartient ce compte.

La **classe « Compte »** permettra d'instancier des objets tels que compte1, compte2, etc. La classe « **TestCompte »** permettra de tester notre définition de comptes.



1- Créer un projet Banque, puis un package gestionCompte (Netbeans).

2- Définissez une classe **Compte.java**, qui initialisera deux attributs d'instance **nom** (chaine) et **solde** (entier). Lesolde pourra être mis à 0 au départ.

3- Définir pour cette classe les méthodes :

depotDe() : Reçoit en paramètre le montant à déposer. A ajouter au solde.

retraitDe() : Reçoit en paramètre le montant à retirer. A retrancher du solde.

getSolde() : Retourne la variable solde.

affiche() : permettra d'afficher le nom du titulaire et le solde de son compte.

3- Créer la classe TestCompte.java qui vous permettra :

- D'instancier plusieurs comptes
- De procéder à des dépots
- De procéder à des retraits
- D'afficher le solde du compte

TP4 : GESTION DU COMPTE AVEC ENCAPSULATION

Objectif : Empêcher depuis l'extérieur les actions directes sur les champs de classes.

On demande de ne plus manipuler directement dans « TestCompte » la variable solde de l'objet de classe « Compte ».

Pour être sûr du respect de cette consigne, on utilise l'**encapsulation**. A cette fin, on utilisera le mot clé « **private** » lors de la déclaration des variables de Compte.

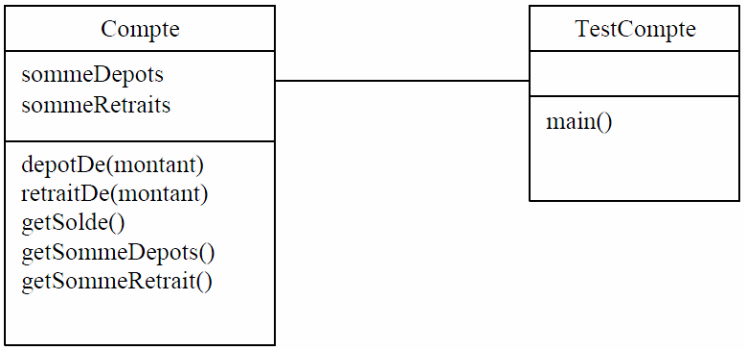
TP5 : SUPPRESSION DE LA VARIABLE SOLDE

Objectif : Poursuivre la manipulation des classes et objets.

Calculer le solde à partir des montants déposés et retirés.
Nous remplaçons la variable **solde** par deux variables « sommeDepots » et « sommeRetraits ». Ces deux variables seront bien entendu encapsulées.

Votre nouvelle définition nécessite de corriger les méthodes précédemment écrites.

Nous conservons la méthode « getSolde() » en modifiant celle-ci, puisque la variable **solde** n'existe plus. Sa fonctionnalité est toujours identique c'est-à-dire retourner la somme des dépôts moins la somme des retraits.



Modifiez légèrement TestCompte.java pour faire les tests.

TP6 : GESTION DES DECOUVERTS

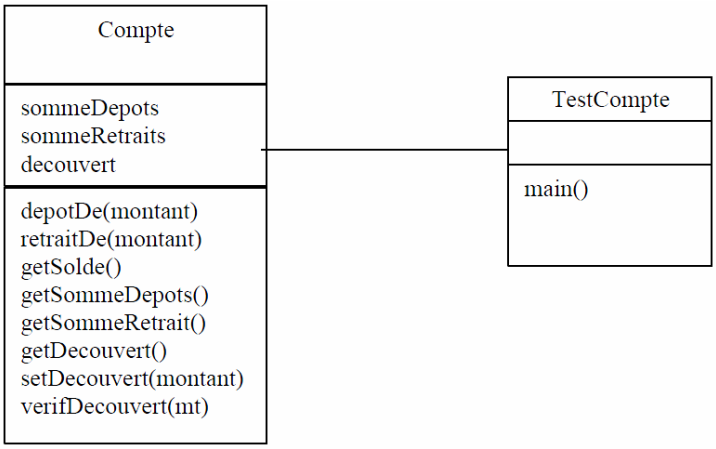
Objectifs : Création et utilisation de fonctions

Nous souhaitons gérer les découverts. Il nous faut donc une nouvelle variable d'instance « decouvert ».

Dans un premier temps, cette variable sera initialisée à 0 par le constructeur.
Mettre à disposition des getters et des setters pour cette nouvelle variable.

La méthode « retraitDe() », peut conserver sa signature. Avant de retirer la somme demandée, il faut vérifier si l'on peut retirer ce montant par rapport au solde et au montant du découvert autorisé. Prévoir une méthode « **verifDecouvert()** » effectuant ce contrôle. Nous ajoutons ainsi une nouvelle fonctionnalité, qui pourra être utilisée ailleurs que dans la méthode « retraitDe() ». Toutefois cette nouvelle méthode ne doit pas être appelée par « TestCompte ».

Les messages indiquant qu'il ne peut y avoir de retrait sont à gérer par l'appelant et non par la classe « Compte ». La méthode « retraitDe() » pourra renvoyer de l'information (booléen , chaîne de caractères…) que « TestCompte » récupérera et exploitera pour afficher un message circonstancié



On peut prévoir à la création des comptes :
- soit un compte ayant un montant de découvert autorisé à 0 (c'est le 1er constructeur)
- soit un compte ayant un montant de découvert variable.

Cela, nous oblige à maintenant à gérer deux constructeurs :

Suite à ces modifications, « TestCompte » doit toujours fonctionner de la même façon. Cependant pour tester cette fois votre traitement avec un découvert valorisé et vérifier le retour de « retraitDe() », vous pouvez ajouter quelques lignes de codes.