La pharmacie [gs02] - Exercice

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Version 22 mai 2018

Table des matières

C++ - La pharmacie (TP)

Mots-Clés Gestion ■
Requis Axiomatique impérative, Classes, Classes (suite) ■
Fichiers dtpharmacie1.txt ■
Difficulté ●●○ (3 h) ■

Objectif
Cet exercice réalise la gestion informatique des clients et médicaments d'une pharmacie.

1 La pharmacie / pgpharmacie

1.1 Modélisation des clients

Un client est caractérisé par un nom et un crédit. Le crédit représente la somme que ce client doit à la pharmacie. Le crédit peut être négatif si le client a versé plus d'argent que le montant.

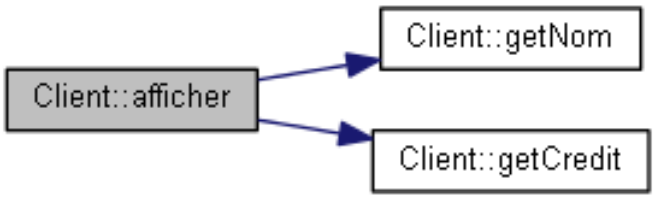
Écrivez une classe `Client` ayant pour attributs le `nom| du client (chaîne de caractères)` `et le \lstinline`credit@ actuel (réel).

Écrivez un constructeur à deux paramètres initialisant les attributs.

Écrivez des accesseurs `getNom` du nom et `getCredit` du crédit.

Écrivez une méthode `afficher` qui affiche (où `[x]` désigne le contenu de l'attribut `x`) :

```
Client [nom] : [credit] euros
```



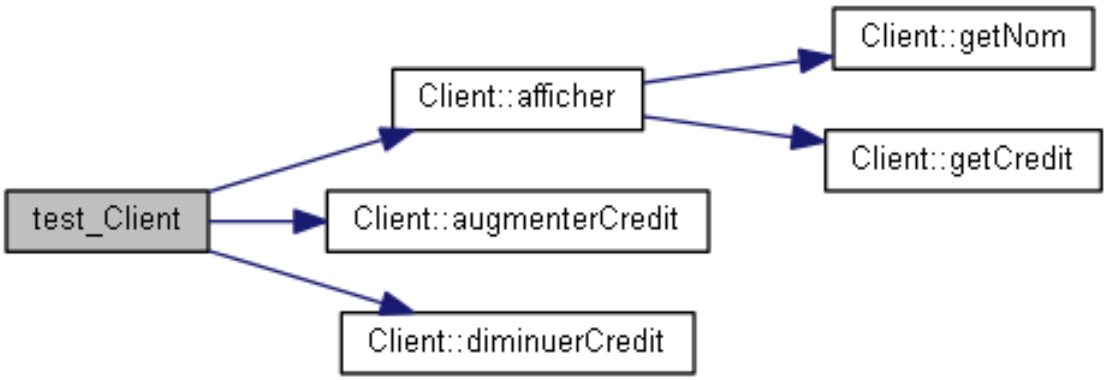
Écrivez une méthode `augmenterCredit(m)` qui augmente le crédit d'un montant `m` (réel).

Enfin, écrivez une méthode `diminuerCredit(m)` qui diminue le crédit d'un montant `m` (réel).

Écrivez une procédure `test_Client` qui :

- Instancie un `Client` de nom "malfichu" et de crédit 0.
- Augmente son crédit de 30 euros.
- Puis le diminue de 20 euros.

Affichez le client après chaque opération.



Testez. Résultat d'exécution :

```
Client malfichu : 0 euros
Client malfichu : 30 euros
Client malfichu : 10 euros
```

1.2 Modélisation des médicaments

Un médicament est caractérisé par un nom, un prix et une quantité en stock.

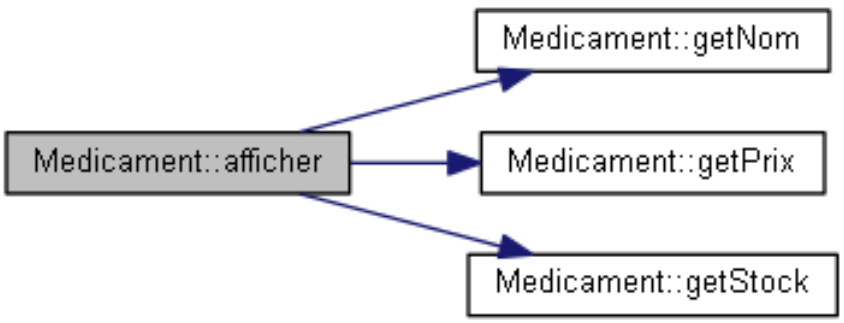
Écrivez une classe `Medicament` ayant pour attributs le `nom` du médicament (chaîne de caractères), le `prix` courant (réel) et le `stock` actuel (entier).

Écrivez un constructeur à trois paramètres initialisant les attributs.

Écrivez des accesseurs `getNom` du nom, `getPrix` du prix et `getStock` du stock.

Écrivez une méthode `afficher` qui affiche (où `[x]` désigne le contenu de l'attribut `x`) :

```
Medicament [nom] ([prix] euros) : [stock]
```



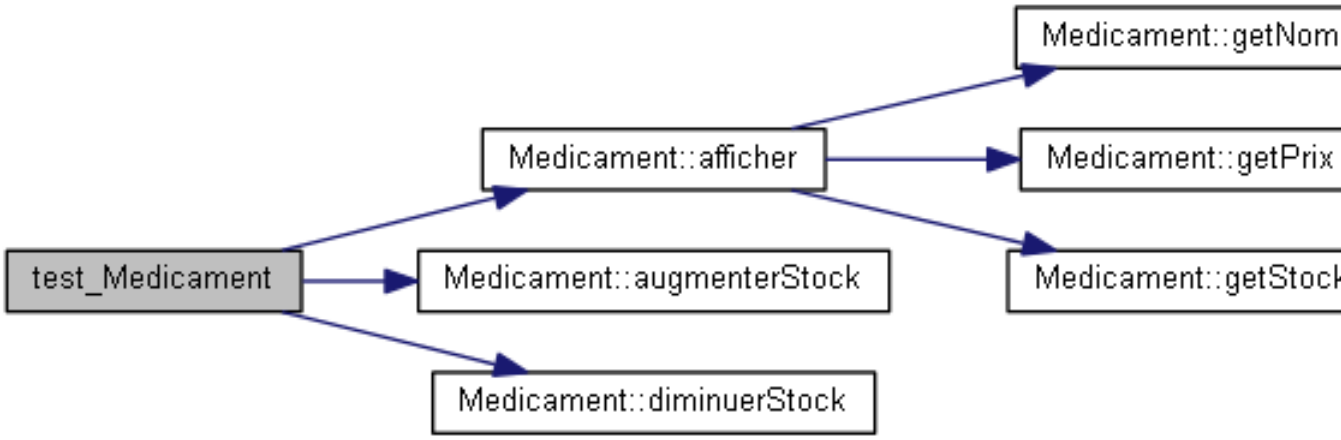
Écrivez une méthode `augmenterStock(q)` qui augmente le stock d'une quantité `q` (entier).

De même, écrivez une méthode `diminuerStock(q)` qui diminue le stock d'une quantité `q` (entier).

Écrivez une procédure `test_Medicament` qui :

- Instancie un `Medicament` de nom "aspiron" de prix courant 20.4 et de stock actuel 5.
- Augmente son stock de 30.
- Puis le diminue de 20.

Affichez le médicament après chaque opération.



Testez. Résultat d'exécution :

```
Medicament aspiron (20.4 euros) : 5
Medicament aspiron (20.4 euros) : 35
Medicament aspiron (20.4 euros) : 15
```

1.3 Modélisation de la pharmacie

Une pharmacie peut être vue comme une classe d'objets ayant pour attribut une liste de clients et une liste de médicaments.

Les fonctionnalités souhaitées par la pharmacie sont :

- Afficher les clients et leurs crédits ainsi que les médicaments et leurs stocks.
- Ajouter des clients ou des médicaments.
- Approvisionner le stock d'un médicament.
- Traiter un achat fait par un client.

Écrivez une classe `Pharmacie` ayant pour attributs la liste (vecteur) des `Client`s `clients` et la liste (vecteur) des `Medicament`s `meds`.

Outil C++

Le modèle de classes `vector` est définie dans la bibliothèque `<vector>`.

Au vu des fonctionnalités souhaitées,
Écrivez les profils des constructeurs et des méthodes.

Écrivez le constructeur par défaut.

Écrivez les accesseurs :

- `nclients` du nombre de clients.
- `nmedicaments` du nombre de médicaments.

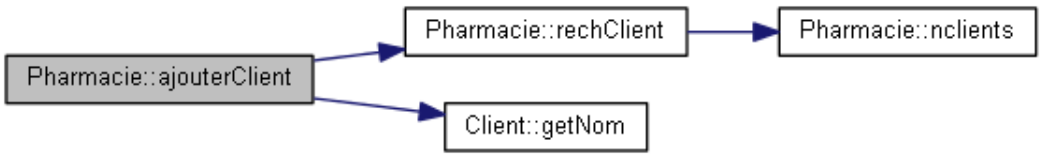
Écrivez une méthode interne `afficherStocks` qui affiche les médicaments et leurs stocks.

De même, écrivez une méthode interne `afficherCredits` qui affiche les clients et leurs crédits.

Déduisez une méthode `afficher` qui affiche les stocks et les crédits.

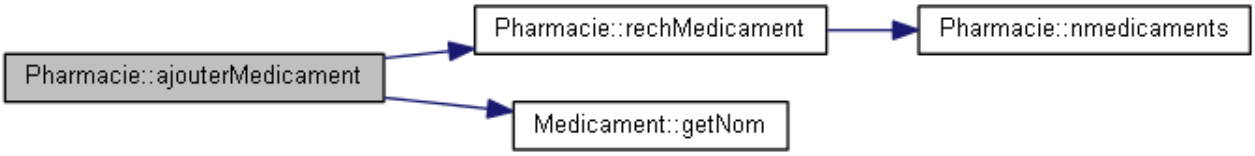
Écrivez une méthode interne `rechClient(nom)` qui recherche un `nom` (chaîne de caractères) dans le vecteur des `Client`s et renvoie son indice, -1 s'il n'existe pas.

Écrivez une méthode `ajouterClient(x)` qui ajoute un `Client x` à la liste des `Client`s si ce dernier n'existe pas. Elle ne fait rien si son nom existe.



De même, écrivez une méthode interne `rechMedicament(nom)` qui recherche un `nom` (chaîne de caractères) dans le vecteur des `Medicament`s et renvoie son indice, -1 s'il n'existe pas.

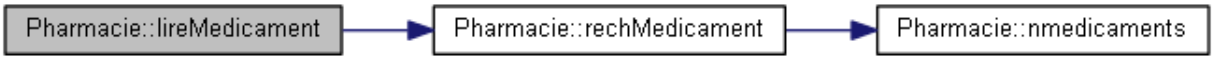
De même, écrivez sa duale `ajouterMedicament(x)` qui ajoute un `Medicament x` à la liste des `Medicament`s si ce dernier n'existe pas. Idem elle ne fait rien si son nom existe.



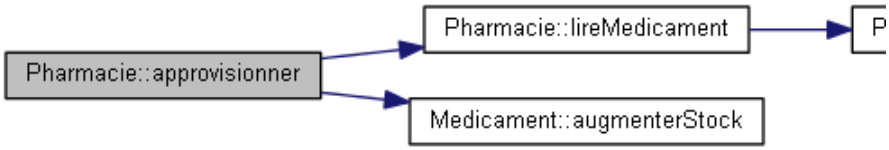
Écrivez une méthode interne `lireMedicament(nc)` qui prend comme paramètre la référence à un entier `nc`. Elle demande le nom (chaîne de caractères) d'un médicament et vérifie si celui existe. Dans l'affirmative,

- La méthode renvoie `Vrai` et restitue dans `nc` l'indice du médicament.
- Sinon elle renvoie `Faux` (et `nc` n'est pas significatif).

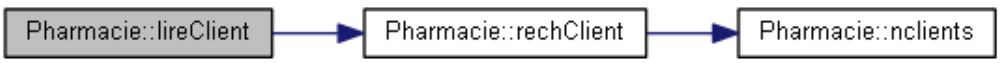
Cette méthode doit boucler jusqu'à ce qu'un médicament soit trouvé ou que l'utilisateur tape « FIN » pour signifier qu'il souhaite mettre fin à la saisie. Elle est utilisée par les opérations `approvisionner` et `traiterAchat`.



Écrivez une méthode `approvisionner` qui approvisionne le stock d'un médicament. Le nom du médicament à approvisionner ainsi que la quantité à ajouter au stock doivent être saisis au clavier. Lorsque le nom du médicament est introduit, il faut vérifier qu'il s'agit bien d'un nom connu dans la liste des médicaments de la pharmacie. Le programme doit boucler jusqu'à introduction d'un nom correct. Rappel : Cette procédure de vérification est prise en charge par la méthode interne `lireMedicament`.

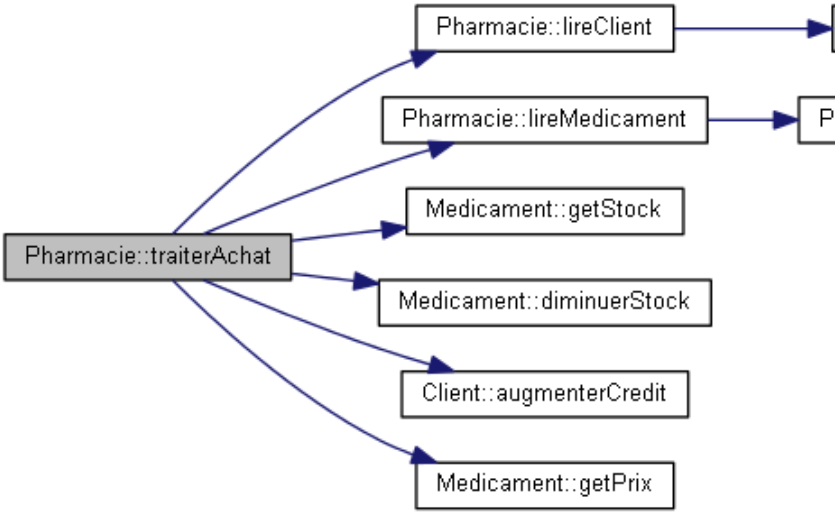


Écrivez une méthode interne `lireClient(nc)` pour les clients, similaire à l'opération `lireMedicament`, c.-à-d. dupliquez la méthode `lireMedicament` puis modifiez-la en conséquence. Elle est utilisée par la méthode `traiterAchat`.



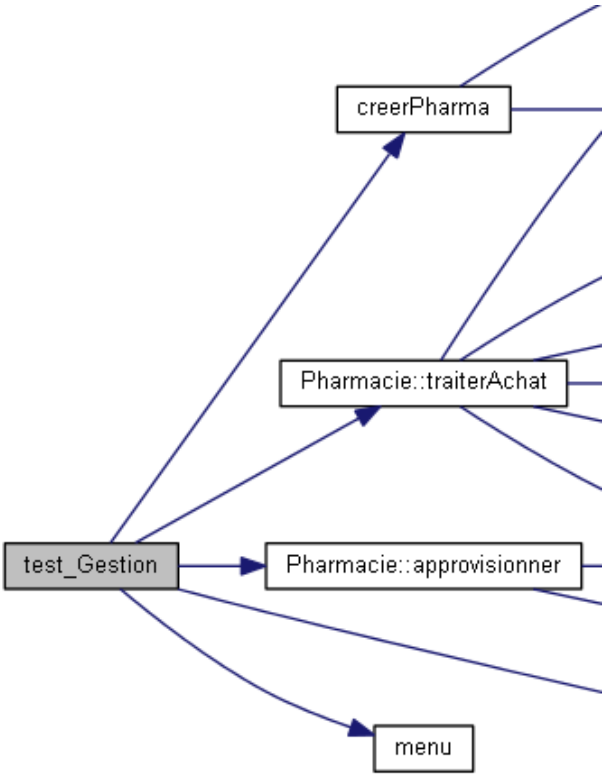
Écrivez une méthode `traiterAchat` qui traite l'achat d'un client, l'achat portant sur un médicament donné dans une quantité donnée. Pour cette transaction le client paie un certain prix. Une opération d'achat aura pour effet de déduire la quantité achetée du stock du médicament correspondant et d'augmenter le crédit du client (d'un montant

équivalent au montant de l'achat moins la somme payée). Les noms du client et du médicament doivent être saisis au clavier. Le programme doit boucler jusqu'à introduction de noms connus aussi bien pour les clients que les médicaments. Ces procédures de vérification sont prises en charge par des méthodes `lireClient` et `lireMedicament`. La quantité achetée et le montant payé seront aussi saisis depuis le terminal. Ils seront supposés corrects.



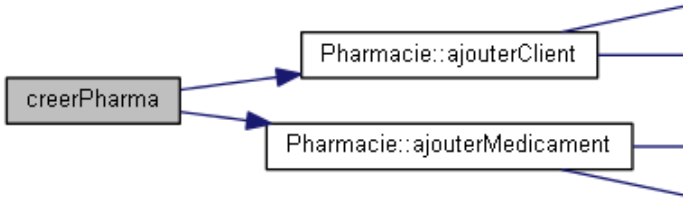
1.4 Programme de gestion

Voici le graphe des appels de la procédure de gestion.



Écrivez une procédure `creerPharma(p)` qui crée une petite `Pharmacie` dans `p` en y ajoutant les éléments suivants :

```
Client("malfichu",0.0)
Client("palichon",0.0)
Medicament("aspiron",20.4,5)
Medicament("rhinoplexil",19.15,5)
```



Écrivez une fonction `menu` qui affiche le menu ci-après et renvoie le choix de l'utilisateur :

```
0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
```

Écrivez une procédure `test_Gestion` qui instancie une `Pharmacie p` puis appelle la procédure `creerPharma(p)` pour créer une petite pharmacie.

Lancez la boucle de gestion en affichant le menu et en effectuant le choix de l'utilisateur.

Testez. Exemple d'exécution fourni en téléchargement.
@[rspharmacie3.txt] Exemple d'exécution :

```
0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 2
Nom du medicament (FIN==fin)? aspiron
Quantite? 2

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
```

```
 -> 3
Affichage des stocks:
 Medicament aspiron : 7
 Medicament rhinoplexil : 5
Affichage des credits:
 Client malfichu : 0 euros
 Client palichon : 0 euros

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 1
Nom du client (FIN==fin)? malfichu
Nom du medicament (FIN==fin)? aspiron
Montant du paiement? 30.0
Quantite achetee? 3

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 3
Affichage des stocks:
 Medicament aspiron : 4
 Medicament rhinoplexil : 5
Affichage des credits:
 Client malfichu : 31.2 euros
 Client palichon : 0 euros

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 1
Nom du client (FIN==fin)? palichon
Nom du medicament (FIN==fin)? aspiron
Montant du paiement? 5
Quantite achetee? 5
Achat impossible... quantite insuffisante

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 1
Nom du client (FIN==fin)? palichon
Nom du medicament (FIN==fin)? rhinoplexil
```

```
Montant du paiement? 200
Quantite achetee? 5

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 3
Affichage des stocks:
 Medicament aspiron : 4
 Medicament rhinoplexil : 0
Affichage des credits:
 Client malfichu : 31.2 euros
 Client palichon : -104.25 euros

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 1
Nom du client (FIN==fin)? febril
Nom du client (FIN==fin)? malfichu
Nom du medicament (FIN==fin)? placebo
Nom du medicament (FIN==fin)? FIN

0: Quitter
1: Achat de medicament
2: Approvisionnement en medicaments
3: Etats des stocks et des credits
 -> 0
```