UNIVERSITE DES SCIENCES ET TECHNOLOGIE HOUARI BOUMEDIENE

Department d'informatique

Programmation orienté

objet

Projet de tp POO : Gestion d'une pharmacie

FAIT PAR :

Nom : **ACHACHI**

Prénom : **EL-Hadj ALI**

Mat : **181832008538**

Groupe : **MATH**

Le schéma représentant la modélisation détaillée

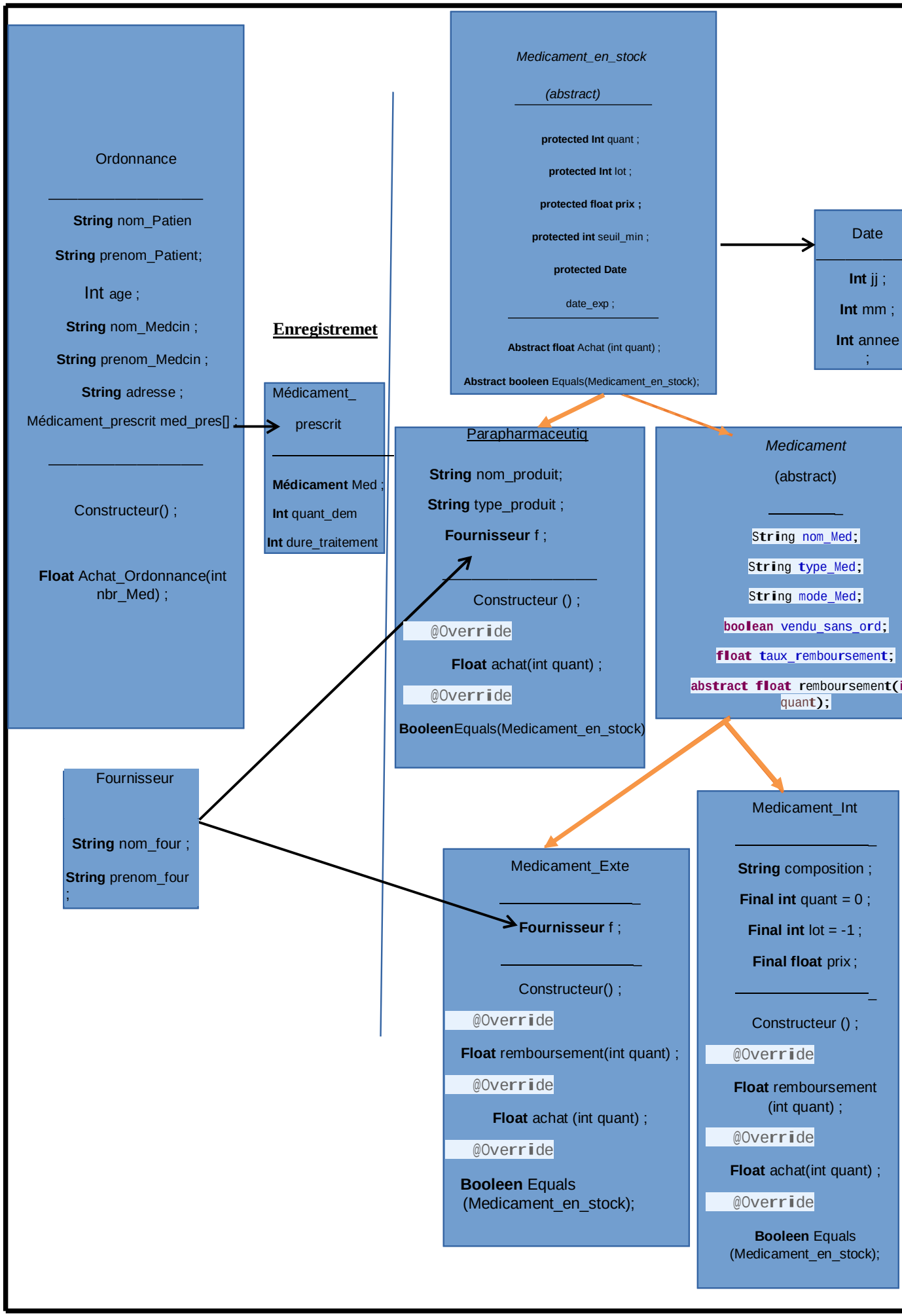
Hiérarchie des classes :

Ordonnance / Medicamen_en_stock(Médicament(interne/extern),parapharma)

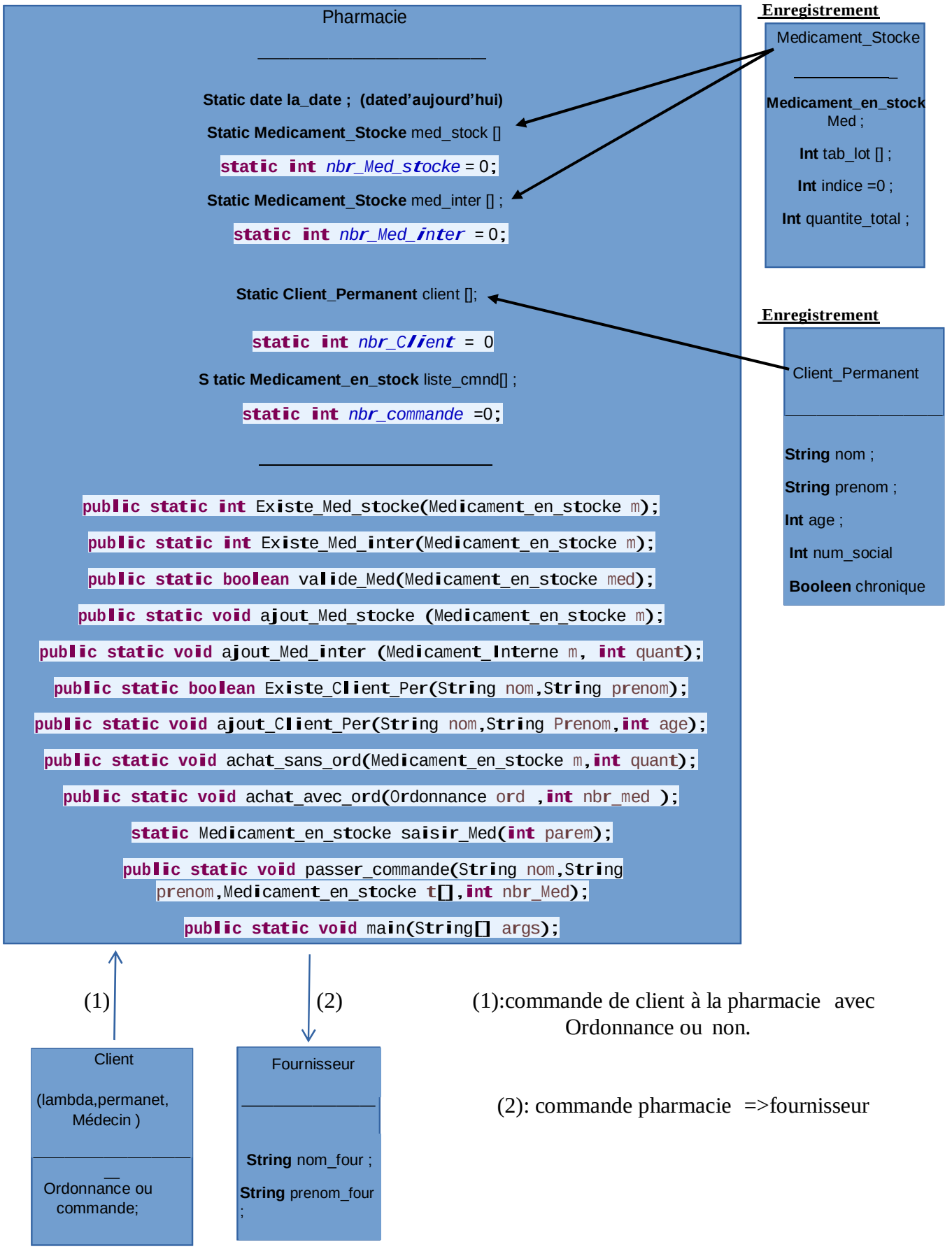
Note :

 Hérite de

 Utilise dans


Ordonnance
String nom_Patien
String prenom_Patient;
Int age ;
String nom_Medcin ;
String prenom_Medcin ;
String adresse ;
Médicament_prescrit med_pres[] ;
Constructeur() ;
Float Achat_Ordonnance(int nbr_Med) ;
Enregistremet
Médicament_ prescrit
Médicament Med ;
Int quant_dem
Int dure_traitement
Medicament_en_stock
(abstract)
protected Int quant ;
protected Int lot ;
protected float prix ;
protected int seuil_min ;
protected Date date_exp ;
Abstract float Achat (int quant) ;
Abstract booleen Equals(Medicament_en_stock);
Date
Int jj ;
Int mm ;
Int annee ;
Parapharmaceutiq
String nom_produit;
String type_produit ;
Fournisseur f ;
Constructeur () ;
@Override
Float achat(int quant) ;
@Override
BooleenEquals(Medicament_en_stock)
Medicament
(abstract)
String nom_Med;
String type_Med;
String mode_Med;
boolean vendu_sans_ord;
float taux_remboursement;
abstract float remboursement(int quant);
Fournisseur
String nom_four ;
String prenom_four ;
Medicament_Exte
Fournisseur f ;
Constructeur() ;
@Override
Float remboursement(int quant) ;
@Override
Float achat (int quant) ;
@Override
Booleen Equals (Medicament_en_stock);
Medicament_Int
String composition ;
Final int quant = 0 ;
Final int lot = -1 ;
Final float prix ;
Constructeur () ;
@Override
Float remboursement (int quant) ;
@Override
Float achat(int quant) ;
@Override
Booleen Equals (Medicament_en_stock);

La classe d'exécution : (la pharmacie)



(1):commande de client à la pharmacie avec Ordonnance ou non.

(2): commande pharmacie =>fournisseur

Explication sur les methodes :

1)-la classe Medicament_en_stock : (la classe Medicament , la classe parapharmaceutique)

```
abstract float achat(int quant):
```

-vérifier l'existence de l'objet dans le tableau med_stock/med_inter ca dépendra L'instance de l'objet

-vérifier si la quantité de ce médicament/parapha est suffisante + la date d'expiration est supérieure a la date d'achat

-retourner le montant total (en appelant à la méthode remboursement) si les conditions d'achat sont valides, si non ajouter le produit s'il est externe/parapharma a la liste des commandes et retourner 0

```
abstract boolean equals(Medicament_en_stocke m):
```

-pour les médicaments interne/externe : retourner true si le médicament courant et le médicament en paramètre sont égaux en nom et en type et en mode de prise

Pour les produits parapha : retourner true si le produit courant et le produit en paramètre sont égaux en nom et en type

2)-la classe Medicament : (la classe Medicament_interne, la classe Medicament_externe) :

```
abstract float remboursement(int quant) :
```

-retourner le prix *(1- taux _remboursement)*la quantité

3)-la classe Ordonnance :

```
Achat_ordenance(int nbr_med) :
```

```
-Calculer le montant total de tous les medicaments prescript existe dans la pharmacie et construire un tableau contenant tous les medicament prescript inexistants
```

-en cas d'inexistant d'un médicament prescrit et le client est un client permanent, on lui demande s'il veut faire un achat partiel, s'il excepte on lui donne le prix à payer, sinon on annule l'achat

- Si le client n'est pas permanent, on lui demande si il veut l'être, en cas d'acceptation on l'ajout à la liste des clients permanent, sinon on annule l'achat.

4)-la classe pharmacie :

```
public static int Existe_Med_stocke(Medicament_en_stocke m):
```

-retourner l'indice du médicament s'il existe dans le tableau med_inetr, sinon on retourne -1

```
public static int Existe_Med_inter(Medicament_en_stocke m):
```

-retourner l'indice du médicament s'il existe dans le tableau med_inetr, sinon on retourne -1

```
public static boolean valide_Med(Medicament_en_stocke med) :
```

-vérifier si la date d'expiration du médicament/parapha est supérieure à la date actuel

```
public static void ajout_Med_stocke (Medicament_en_stocke m):
```
-ajouter le médicament en cas d'inexistence si non augmenter juste la quantité

```
public static void ajout_Med_inter (Medicament_Interne m, int quant):
```
-ajouter le médicament en cas d'inexistence si non augmenter juste la quantité

```
public static boolean Existe_Client_Per(String nom,String prenom):
```
-retourner true si le client existe dans le tableau client_per

```
public static void ajout_Client_Per(String nom,String Prenom,int age):
```
-ajouter le client s'il n'existe pas dans le tableau client_per

```
public static void achat_sans_ord(Medicament_en_stocke m,int quant):
```
-vérifier si le médicament peut être vendu sans ordonnance (s'il est de type Médicament)
-si la condition passée est vérifiée : on fait l'appelle à la méthode achat de ce Médicament/parapharmaceutique
- si il y a des médicaments commandés alors on les demande au fournisseur correspondant

```
public static void achat_avec_ord(Ordonnance ord ,int nbr_med ):
```
-appelle à la méthode Achat_ordonnance + -si il y a des médicaments commandés alors on les demande au fournisseur correspondant

```
static Medicament_en_stocke saisir_Med(int parem):
```
-si parem=1 : on saisit que le nom et le type (et le mode du prise s'il s'agit d'un medicament)
-si parem =0 : on saisit tous les propriétés du médicament/produit parapharmaceutique

```
public static void passer_commande(String nom,String prenom,Medicament_en_stocke t[],int nbr_Med):
```
-passer tous les medicament_externe/parapha commandes au fournisseur afin de les importer

Les cas d'utilisation :

1: Ajouter un médicament ou bien un produit parapharmaceutique au stocke :

avec la méthode **saisir_Med()** , on a la possibilité de saisir soit un médicament interne ou externe ou bien un produit parapharmaceutique ;

-si l'utilisateur a saisi un médicament interne, alors demander la quantité à saisir , et puis ajouter le médicament en cas d'inexistence si non augmenté juste la quantité avec la méthode **ajout_Med_inter (Medicament_Interne m, int quant)** .

-si l'utilisateur a saisi un médicament externe ou un produit parapharmaceutique, alors ajouter le médicament en cas d'inexistence si non augmenté juste la quantité avec la méthode **ajout_Med_stocke (Medicament_en_stocke m).**

2:Vendre un Médicament sans ordonnances :

avec la méthode **saisir_Med()** , on a la possibilité de saisir soit un médicament interne ou externe ou bien un produit parapharmaceutique ;

après avoir demander la quantité, et avec la méthode :
achat_sans_ord(Médicament_en_stocke m,int quant) , qui fait un appelle à la méthode **achat()** ; de la classe Medicament_en_stock , et cette dernière vérifie si le médicament peut être vendu sans ordonnance, et que la quantité est suffisante. Si la vérification est positive, elle mis a jour la quantité, et elle renvoie le prix à payer.

Si la quantité de médicament demander est insuffisante et que ce dernier est un médicament externe ou un produit parapharmaceutique, on l'ajout à la liste des médicaments à commander au fournisseur.
3:Vendre des médicaments avec ordonnances :

on crée un nouveau objet ordonnance , avec les informations donnée ,et on fait appelle a la méthode **achat_avec_ord(Ordonnance ord ,int nbr_med) ,** qui appelle la méthode **achat_ordonnance()** de la classe Ordonnance , cette dernière vérifie l'existence des médicament , et prépare une liste des médicament manquant pour les demander au fournisseur plu tard en cas ou et calcule le prix_total avec la méthode **achat()**, si c'était le cas , on vérifie si le client est un client permanent ,avec la méthode **Existe_Client_Per(String nom,String prenom)** de la classe Pharmacie2 ,si il l'est ,on lui demande si il veut faire un achat partiel , si il excepte on lui donne le prix à payer, si non on annule l'achat .

Si le client n'est pas permanent, on lui demande si il veut l'être, en cas d'acceptation on l'ajout à la liste des client permanent avec la méthode **ajout_Client_Per(String nom,String Prenom,int age)** de la classe Pharmaie2 , si non on annule l'achat .

4 : Ajouter un client permanent :
cela se fait avec la méthode **ajout_Client_Per (String nom, String Prenom, int age) ,**
5 : Passer une commande par le médecin ou bien par le client:

-passer tous les medicament_externe/parapha commandes au fournisseur afin de les importer

Finalement , on peut dire que ce diagramme comprend tout ce que on a vu dans le module poo (l'encapsulation, casting, héritage , polymorphisme, les classe abstraites) , par exemple dans la méthode `achat_sans_ord(Medicament_en_stocke m,int quant) :` on peut faire l'achat avec n'importe quelle médicament(interne/externe ou bien produit parapharmaceutique) , et tout cela grâce à la notion de polymorphisme et down casting