**Description des classes du projet de POO :**

En premier lieu voici la liste des fichiers structurés nécessaires au fonctionnement des opérations décrites plus bas :

|  |  |  |
| --- | --- | --- |
| **Contenu du fichier** | **Nom du fichier** | **Format** |
| Clients permanents. | Clients.stk | NumSS,Nom,Prenom,Age,Chronique-traitement,quant-traitement2,quant2-… |
| Commandes de clients. | Commandes.stk | nomMed,NumSS,quantité,durée- |
| Commandes servies de clients. | CommandesServies.stk | nomMed,NumSS,quantité ,numLot,prix,datePeremp- |
| Commandes de médecins. | CommandesMedecin.stk | nomMed,quant-nomMed,quant-…-|nom,nomClinique,codeCommande |
| Commandes servies de médecins. | CommandesMedecinS.stk | numLot,nomMed,quant,prix,datePer-numLot,nomMed,quant,prix,datePer-…-|nom,nomClinique,codeCommande |
| Fournisseurs. | Fournisseurs.stk | Nom,numero,mail- |
| Médicaments en interne. | Interne.stk | Nom|type|mode|sousOrd|[ingred1-dose1 ;ingred2-dose2…. ;]| |
| Médicaments produits en externe. | Externe.stk | Nom|type|mode|sousOrd|nomFournis| |
| Stock de médicaments en interne. | StockInterne.stk | nomMedicament-Quantité1,numLot1,dateExp1,prix1-  Quantité2,numLot2,dateExp2,prix2-… |
| Stock de médicaments produits en externe. | StockExterne.stk | // |

Quelques clarifications sur certains éléments des fichiers :

* NumSS est un long qui représente le numéro de sécurité sociale du client.
* Chronique est un boolean, true si le client a une maladie chronique, false sinon.

Au tour des classes maintenant, il y en a au total 13 différentes :

**Paire<T, Y>** : est une classe générique qui consiste d’un paire de deux éléments (T valeur et Y valeur2) **public,** son utilité principale est de simplifier l’enregistrement de certaines information ex : Ingrédient et son dosage, médicament et sa quantité …etc. La classe est munie uniquement d’un constructeur avec paramètres et une fonction toString ().

**Médicament** : est une classe munie de quatre attributs **private** (String nom, type, mode et boolean ordonnance), un constructeur avec paramètres et les getters des attributs.

**MedicamentEnInterne** : est une classe fille de la classe **Médicament** munie d’un attribut **private** en plus de type ArrayList<**Paire**<String, Integer>> qui représente les ingrédients contenus dans le médicament et leurs dosages. La classe est munie d’un constructeur qui fait appel au constructeur de la classe mère et vérifie les dosages, si leur somme n’est pas 100% alors il y a certainement une erreur et l’utilisateur en sera notifié.

**Fournisseur** : est une classe munie de trois attributs String **private** nom numéro et mail. La classe est munie d’un seul getter pour l’attribut nom et deux constructeurs ; le premier récupère les informations du fournisseur depuis le fichier correspondant avec son nom uniquement, l’autre instancie un nouveau fournisseur et l’ajoute au fichier correspondant.

* **public** boolean existe () renvoi true si le fournisseur est sur le fichier des fournisseurs, false sinon.
* **public** void fournis (**Médicament** Med, Médicaments, Stock, inter) prend en entrée le médicament fournis, la liste des médicaments produits en externe et le stock courant de médicaments. Ajoute le médicament au stock et à la liste des médicaments s’il n’en fait pas parti.

**MedicamentProduitEnExterne** : est une classe fille de la classe **Médicament** munie d’un attribut **private** en plus de type **Fournisseur** qui représente le fournisseur du dit médicament. La classe est munie de deux constructeurs différents, l’un est un constructeur qui prend a la place d’un objet Fournisseur un String qu’il utilise pour récupérer le fournisseur en question avec le premier constructeur de **Fournisseur**, l’autre constructeur prend trois paramètres de type String qui représentent le nom numéro et adresse mail du fournisseur qui sera ajouté au fichier des fournisseurs avec le second constructeur de la classe **Fournisseur**.

**MédicamentPrescrit** : Une classe très basique qui a comme attributs une String nom et deux int quantité et durée tous **protected**. Elle est munie d’un constructeur avec paramètres simple. Son utilité est uniquement en tant qu’entrée pour les opérations d’achat des autres classes.

**MedicamentEnStock :** La classe la plus importante pour les traitements, elle a cinq attributs **private**, **Médicament** Med int quantité, numeroLot Date dateExpiration et double prix. Elle est munie d’un constructeur avec paramètres simple et un getter pour chaque attribut et un setter pour l’int quantité.

* **public** double TauxDeRemboursement () Renvoi le taux de remboursement du médicament.

**ClientPermanent** : est une classe munie de six attributs **private**; String nom, prénom ; int âge ; long numeroDeSecuritéSociale ; boolean chronique et HashMap<String, Integer> traitements. Munie de deux constructeurs, le premier est un constructeur avec paramètres ordinaire et le second qui a comme paramètre un long qui représente le numéro de sécurité sociale qui sera utilisé pour récupérer les informations du client du fichier correspondant.

* **public** void AjouterClient (boolean interm) permet d’ajouter un client au fichier des clients, quand le paramètre est a true l’ajout se fait vers un fichier intermédiaire (opération nécessaire pour un autre traitement).
* **public** void Acheter (boolean ord, prescription, Stock, exter, inter) prends en entrée un booléen qui indique si le client a une ordonnance, la liste des médicaments prescrits et le stock courant. Vérifie la disponibilité des médicaments, permet de passer commande si le client le veut, calcule le prix de la transaction, mets à jour le stock, mets à jour la liste des traitements du client.
* **public** void PasserCommande (String nomMedicament, int quantité, int durée) ajoute une commande au fichier des commandes de clients, la fonction précédente y fait appel.
* **public** void AnnulerCommande (String nomMedicament, Stock, inter, exter) elle annule une commande en l’enlevant du fichier des commandes si pas encore servie sinon l’enlever du fichier des commandes servies et restaurer les médicaments vendus au stock.
* **public** void RetirerCommande (String nomMedicament) le client se présente pour récupérer sa commande, si elle à été servie elle lui calcule le prix à payer et la commande est enlevée du fichier des commandes servies.
* **public** **static** void ServirCommandesClients (Stock, exter, inter) fait le parcours du fichier des commandes, vérifie pour chaque commande la disponibilité du médicament dans le stock et dans le cas ou un lot correspond à la commande la transférer vers le fichier des commandes servies après avoir enlevé du stock le nombre de pièces commandées, puis faire signe à l’utilisateur de contacter le client concerné.

**MédecinConventionné**: Une classe munie de deux attributs **private** de type String nom et nomClinique (nom est le nom du médecin) et un constructeur avec paramètres normal.

* **public** void PlacerCommande (meds) prends en entrée la liste des médicaments à commander et leurs quantités. Ajoute la commande au fichier des commandes de médecins et lui affecte un code qui sera utilisé pour annuler ou retirer la commande.
* **public** void AnnulerCommande (String code, Stock, exter, inter) annules la commande en la retirant du fichier ou elle est présente avec son Code, si elle est sur le fichier des commandes servies restituer les médicaments servies au stock.
* **public** **static** void ServirCommandes (Stock, exter, inter) fait le parcours du fichier des commandes, vérifie pour chaque commande la disponibilité de tout les médicament dans le stock et dans le cas ou un lot correspond pour chaque élément de la commande la transférer vers le fichier des commandes servies après avoir enlevé du stock le nombre de pièces commandées de chaque médicament, puis faire signe à l’utilisateur de contacter le médecin/clinique concerné.
* **public** void RetireCommande (String code) retire la commande dont le code est en entrée depuis le fichier des commandes servies.

**Main**: La classe sans attributs qui contient la fonction main (), c’est un test des opérations des autres classes. On y retrouve toutes les opérations de l’interface graphique et deux opérations importantes ;

* **public** **static** void Création () qui crée les fichiers à vide s’ils n’existent pas.
* **public** **static** void Initialisation () qui lance la première fenêtre.

**ButtonMenu :** Une classe d’Oracle qui transforme une colonne d’une JTable en une colonne de boutons.

**Donnees** : Une classe sans attributs qui regroupe un ensemble de d’opérations nécessaires au fonctionnement et la synchronisation des autres classes mais qui sémantiquement n’appartiennent pas aux autres classes.

* **public** **static** **MedicamentEnInterne** RecupMedicInter (String ligne) décode la chaîne en entrée et en extrait un **MedicamentEnInterne**.
* **public** **static** **MedicamentProduitEnExterne** RecupMedicExter (String ligne) décode la chaîne en entrée et en extrait un **MedicamentProduitEnExterne**.
* **public** **static** String coderMedic (**Médicament** Med) code le médicament en entrée dans le format qui lui correspond.
* **public** **static** HashMap<String, **MedicamentEnInterne**> RecupererMedicInter () récupère les médicaments en interne depuis le fichier correspondant dans une HashMap.
* **public** **static** HashMap<String, **MedicamentProduitEnExterne**> RecupererMedicExter () récupère les médicaments produits en externe depuis le fichier correspondant dans une HashMap.
* **public** **static** void EnregistrerMedicaments (boolean interne, exter, inter) selon le paramètre booléen, si true mets à jour le fichier des médicaments en interne sinon celui des médicaments produits en externe. On y fait appel quand un fournisseur fournis quoi que ce soit ou si le stock de médicament en interne est mis à jour.
* **public** **static** void EnregistrerStock (Stock, exter, inter) mets à jour les fichiers de stock de médicaments en interne et de médicaments produits en externe.
* **public** **static** HashMap<String, ArrayList<**MedicamentEnStock**>> RecupererStock (exter, inter) prends en entrée deux HashMap qui contiennent les listes des médicaments vendus par la pharmacie (les résultats de RecupererMedicInter () et RecupererMedicExter ()). Renvoi le stock actuel depuis les fichiers de stock.
* **public** **static** void AjoutStockInterne (**MedicamentEnStock** Med, meds, Stock, exter) prends en entrée le médicament en interne à ajouter au stock, la liste des médicaments en interne et le stock. Ajoute un lot d’un médicament en interne au stock et s’il ne figure pas sur la liste des médicaments en interne l’y ajouter. (L’équivalent de fournis () pour les médicaments en interne).
* **public static** void NettoyerStock (Stock, exter, inter) Parcours le stock et enlève tout les lots dont la quantité restante est de 0 ou dont la date de péremption est dépassée et invite l’utilisateur à contacter les fournisseurs concernés. Si le nombre de lots restants du médicament est de 0 le médicament lui-même est retiré du stock.
* **public** **static** void Acheter (boolean ord, prescription, Stock, exter, inter) c’est l’équivalent de Acheter () de **ClientPermanent** pour les clients lambda i.e. pas de mise à jour des traitements et pas de commandes.
* **public** **static** String [] getModes () Qui renvoie les modes de prise des médicaments.
* **public** **static** String [] getTypes () Qui renvoie les types de médicaments.
* **public** **static** long Aujourdhui () Qui renvoie la date actuelle en ms.

**NB:**

Le type de Stock = HashMap<String, ArrayList<**MedicamentEnStock**>>

Le type de prescription = ArrayList<**MédicamentPrescrit**>

Le type de exter = HashMap<String, **MedicamentProduitEnExterne**>

Le type de inter = HashMap<String, **MedicamentEnInterne**>