

2020-2021

Module X9IB010 Programmation Objet pour les biologistes
Développement d'un logiciel pour une pharmacie



Simon CHEVOLLEAU
Nicolas DIBOT
M2 Bioinformatique pour les Biologiques (BB)

1. Cahier des charges du logiciel

Le but de ce projet était de réaliser un **logiciel de gestion des stocks des médicaments pour une pharmacie**, à travers un programme en C++ utilisant une modélisation objet.

Un menu permet à un pharmacien n'ayant pas de compétences en programmation d'accéder à l'intégralité des fonctionnalités de notre programme de manière intuitive. Il ne s'agit néanmoins pas d'une interface graphique mais d'options activables dans le terminal.

La plupart des fonctionnalités que nous avons développées sont celles décrites dans l'énoncé:

- Il s'agissait dans un premier lieu de **décrire précisément chaque type de médicament** avec une série d'attributs leur correspondant (nom, laboratoire titulaire, forme etc...).
- Le deuxième point essentiel était de permettre **une gestion des stocks** des médicaments: permettre à la fois au pharmacien de définir la taille du stock pour chaque médicament, en fonction de ses livraisons, mais aussi de la délivrance des médicaments aux patients. Ainsi, le pharmacien peut-être informé, via un fichier spécifique généré quotidiennement, de la rupture de stock de certains médicaments.
- Le troisième point est l'archivage des instances réalisées lors d'un lancement ponctuel du programme (état des stock, noms des médicaments disponibles...), afin de les garder à jour lors des lancements suivants.

Nous avons aussi développé quelques fonctionnalités qui n'étaient pas explicitement demandées mais qui nous semblaient pertinentes :

- L'intégralité du programme, commentaires et interface, est rédigé en **anglais**, afin de le rendre disponible à l'international et de pouvoir le vendre très cher à Google. De plus, la répartition des commentaires a été conçue de manière à être à la fois rigoureuse et précise mais aussi succincte par souci de lisibilité et d'efficacité.
- L'enregistrement des données à la fermeture du programme se fait par **sérialisation**, c'est-à-dire en enregistrant toutes les instances dans un fichier binaire capable de mémoriser la notion d'instance sans avoir à utiliser un constructeur pour les charger à chaque nouvelle utilisation. Il semblerait que ce soit une pratique assez standard dans ce genre de logiciel, et nous avons envie d'apprendre à la maîtriser. Nous avons pour cela utilisé la librairie **Boost**.

2. Organisation du programme

2.1 Contraintes et choix

Notre **première approche** a été de proposer une organisation du programme avec 3 classes:

- Une classe **Pharmacy**, représentant l'interface du pharmacien pour la gestion de son stock et l'ensemble des fonctionnalités à disposition. Une couche supplémentaire d'interface sera ajoutée dans le main, avec un menu, mais la majorité des options proposées seront des méthodes de cette classe.
- Une classe **Prescription**, permettant de modéliser une prescription donnée pour un patient.
- Une classe **Drug** permettant de décrire tous les attributs d'un médicament et de contenir un attribut numérique donnant le nombre de médicaments disponibles.

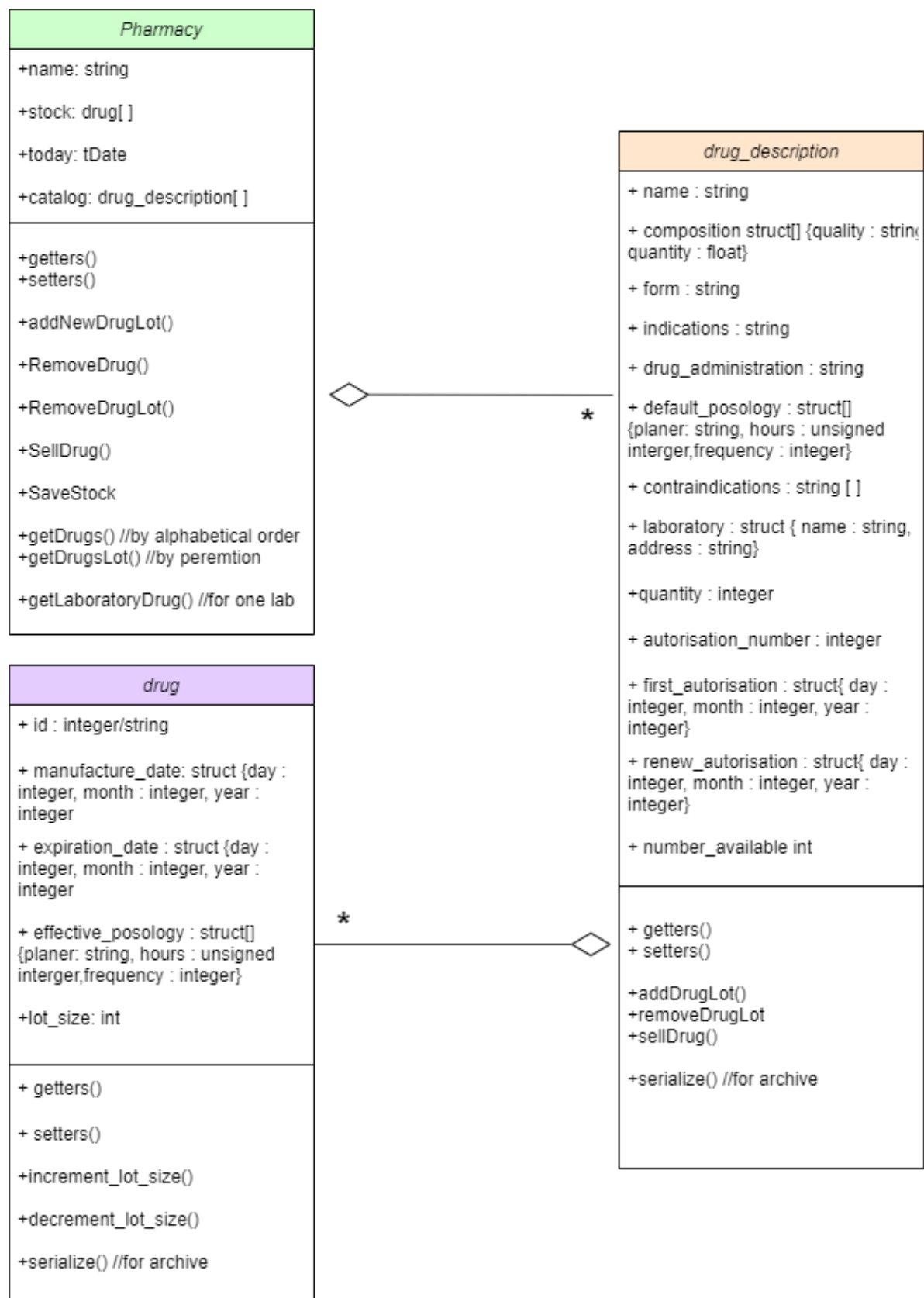
Néanmoins, cette approche présentait **des fragilités**, elle n'a donc pas été retenue.

D'une part, la classe *Prescription* était inutile, dans la mesure où il n'est pas utile d'archiver les prescriptions déjà traitées si le stock de la pharmacie est mis à jour lors de ce traitement. D'autre part, la classe *Drug* ne permettait de prendre en compte que pour un même médicament, plusieurs lots avec des dates d'expiration différentes pouvant coexister.

Ainsi, nous avons **réorganisé l'organisation** de cette manière:

- Une classe **Pharmacy** comme expliqué précédemment
- Une classe **Drug_description**, contenant l'ensemble des informations fixes pour un même médicament, ainsi qu'un vecteur contenant toutes les instances de la classe suivante.
- Une classe **Drug** correspondant à un lot de médicaments avec la même date d'expiration. Chaque instance de lot de cette classe sera, pour un même médicament, stockée dans un vecteur contenu dans une instance de **Drug_description**.

2.2 Diagramme UML et explications



| figure 1 : *diagramme UML des relations entre les classes du logiciel de gestion de pharmacie*

Les trois classes décrites dans la partie précédente sont représentées dans le diagramme UML (cf Figure 1).

Il n'y a **aucune relation d'héritage**, ce n'était pas nécessaire dans le cadre de ce projet. On aurait pu imaginer un héritage entre la classe *Drug_description* (classe mère) et la classe *Drug*, impliquant que cette dernière ne représente non plus un lot de médicaments mais une boîte individuelle de médicament. Néanmoins, par souci d'optimisation de l'instanciation (il aurait fallu instancier les attributs instance par instance pour cette classe), nous n'avons pas retenu cette option.

Il y a donc une relation d'**aggrégation** (losange blanc) **entre la classe *Pharmacy* et la classe *Drug_description*** :

- Une pharmacie peut exister sans qu'il n'y ait de types de médicaments dans son catalogue.
- Une pharmacie peut avoir entre 0 et une infinité de types médicaments. Ils sont contenus dans un tableau de *Drug_description*, qui est l'un des attributs de la classe *Pharmacy*.
- Un type de médicament ne peut correspondre qu'à une seule pharmacie (spécifiquement dans notre logiciel qui ne fonctionne qu'avec une pharmacie, ce n'est bien sûr pas le cas dans la vie réelle)

Il y a aussi une relation d'**aggrégation** (losange blanc) **entre la classe *Drug_description* et la classe *Drug***.

- Un type de médicament (*Drug description*) peut exister sans qu'il n'y ai de lots de médicaments lui correspondant.
- Un type de médicaments peut avoir entre 0 et une infinité de lots médicaments lui correspondant. Ils sont contenus dans un tableau de *Drug*, qui est l'un des attributs de la classe *Drug_description*.
- Un lot de médicament ne peut correspondre qu'à un seul type de médicament.

3. Comportement des classes et fonctionnement des méthodes

3.1 Classe *Pharmacy*

La classe *Pharmacy* se comporte comme l'**interface** du logiciel, et permet donc d'intervenir indirectement sur les instances des autres classes. Ainsi, des noms de signature de fonction que l'on aurait pu attendre dans l'une des autres classes s'y trouvent.

Signature de la fonction	Description du fonctionnement
<code>addNewDrugLot()</code>	<p>Cette fonction permet d'enregistrer dans le stock un nouveau lot (<i>Drug</i>) d'un médicament dont la description a déjà été instanciée.</p> <p>Elle affiche un message à l'écran pour demander d'écrire le nom de la description du médicament, puis vérifie si cette description a été instanciée.</p> <ul style="list-style-type: none">- Si ce n'est pas le cas, elle affiche un message d'erreur demandant d'ajouter cette description.- Sinon, elle instancie une classe <i>Drug</i> qu'elle stocke dans le vecteur de médicaments de l'instance <i>Drug_description</i> correspondante.
<code>getSpecificDrugLots()</code>	<p>Cette fonction permet d'afficher tous les lots disponibles pour une description écrite par l'utilisateur, en parcourant tous les indices d'un attribut de type <i>vector</i> de <i>Drug_description</i> contenant toutes les instances des lots de <i>Drug</i> correspondant à cette instance.</p> <p>Si l'instance de la description demandée n'existe pas, elle affiche un message d'erreur.</p>
<code>getLaboratoryDrug()</code>	<p>Cette fonction affiche les noms de tous les médicaments appartenant à un laboratoire.</p> <p>Le nom du laboratoire est écrit par l'utilisateur, si il n'existe pas, un message d'erreur s'affiche.</p>
<code>removeDrug()</code>	<p>Cette fonction permet de supprimer toutes les instances d'une description de médicament donnée par l'utilisateur, si plusieurs descriptions existent avec le même nom (conflit de noms), ainsi que toutes les instances des lots de médicaments contenus dans celle-ci.</p>

	Le nom du médicament est écrit par l'utilisateur, si il n'existe pas, un message d'erreur s'affiche.
removeDrugLot()	<p>Cette fonction permet de supprimer un lot spécifique d'une description de médicament spécifique. Les lots étant différenciés les un des autres par un identifiant (ID), on demande à l'utilisateur d'entrer, en plus du nom du médicament correspondant à ce lot, l'ID du lot à supprimer.</p> <p>Si le médicament n'existe pas, un message d'erreur s'affiche. De même si l'id du lot ne correspond à rien, un message d'erreur s'affiche.</p>
sellDrug()	<p>Cette fonction permet de vendre un médicament à un client présentant une prescription. Elle demande à l'utilisateur le nom et le nombre de médicaments souhaités.</p> <p>Elle se contente de trouver l'instance adéquate du médicament. Elle appelle ensuite une autre fonction sellDrug de la classe <i>Drug_description</i> qui se chargera de vérifier s'il reste un nombre suffisant du médicament souhaité, et de décrémenter le stock en conséquence, et de supprimer les lots vides suite à la soustraction des boîtes de médicaments.</p> <p>Si le médicament n'existe pas, un message d'erreur s'affiche.</p>
save()	Cette fonction permet de sauvegarder toutes les instances par "serialization". Toutes les instances vont alors avoir leurs attributs sauvegarder, les attributs scalaires sont enregistrés de manière simple. Alors que les attributs complexes comme les classes ou conteneurs (vecteur, chaînes de caractères...) seront enregistrés à l'aide d'une méthode de serialization qui doit exprimer la façon d'enregistrer les attributs de la classe ou de la struct. Pour les conteneurs une library de serialization s'en occupent.
getDrugs()	Cette fonction affiche l'ensemble des noms des descriptions de médicaments dans l'ordre alphabétique. Le vecteur contenant ce nom a d'ores-et-déjà été trié lors des insertions, il n'y a donc pas de tri à faire dans le cadre de cette fonction.
	Cette fonction affiche l'ensemble des id des lots de médicaments dans l'ordre de leur date d'expiration (et pas des descriptions de médicaments car deux lots peuvent correspondre à la même description mais avoir une date d'expiration différente).

getDrugsLot()	<p>Contrairement à la fonction précédente, elle ne fait pas appel à un vecteur déjà trié mais réalise elle-même le tri en utilisant une lambda indiquant comment effectuer le tri à partir de dates dans le format yyyy/mm/dd.</p> <ul style="list-style-type: none"> - L'ensemble des vecteurs de <i>Drug</i> contenus dans chaque <i>Drug_description</i> sont ajoutés dans un seul et même long vecteur. - Ce vecteur est trié en fonction des dates d'expiration. - Les lots sont affichés dans l'ordre à l'écran. On précise le nom du médicament auquel ils correspondent, leur ID ainsi que leur date d'expiration.
---------------	---

3.2 Classe *Drug_description*

Cette classe a, en plus des fonctions détaillées ci-dessous, un **constructeur** par défaut.

Signature de la fonction	Description du fonctionnement
sellDrug(int& number)	<p>Cette fonction est appelée par une autre fonction sellDrug de la classe Pharmacy. Elle s'occupe spécifiquement de la gestion des stocks relative à la vente de médicaments.</p> <ul style="list-style-type: none"> - Elle vérifie que le stock n'est pas vide et affiche un message "out of stock" le cas échéant. - Elle décrémente le nombre de médicaments d'un ou plusieurs lots en fonction du nombre de médicaments passés en paramètres, via une autre fonction de la classe Drug; decrementLotSize() - Si un lot est vidé par cette décrémentation, il est supprimé. - Si seulement une partie des médicaments demandés ont été délivrés, le nombre de médicaments restant à délivrer après la rupture de stock est affiché à l'écran.
addDrugLotDrug drug()	Un lot passé en paramètre est ajouté au vecteur de lots de l'instance de la description de médicaments.
removeDrugLotconst (std::string S_ID)	<p>Toutes les occurrences d'un id de lot passé en paramètre sont supprimées.</p> <ul style="list-style-type: none"> - Le vecteur de lots de l'instance est parcouru jusqu'au dernier. - Si le lot est trouvé, il est supprimé.

3.3 Classe Drug

Cette classe a, en plus des fonctions détaillées ci-dessous, un **constructeur** permettant d'ajouter une nouvelle instance de *Drug* (un nouveau lot) en demandant à l'utilisateur d'entrer au clavier chaque attribut de la classe, excepté le nom du médicament.

Signature de la fonction	Description du fonctionnement
incrementLotSize(const int& number)	Cette fonction incrémente le nombre de médicaments contenus dans l'instance du lot du nombre passé en paramètre.
decrementLotSize(const int& number)	Cette fonction décrémente le nombre de médicaments contenus dans l'instance du lot du nombre passé en paramètre. Elle retourne false si il reste des médicaments dans le lot, true si il n'en reste pas. Elle modifie par ailleurs le nombre de boîtes de médicaments à desservir.

4. Notice utilisateur et présentation des résultats du programme avec un jeu d'essai

Cette notice est aussi une présentation du programme avec des jeux d'essai. En effet, chaque instruction sera illustrée d'un exemple faisant office de jeu d'essai.

Le programme fourni dans le répertoire "programme" du fichier zip est déjà compilé. Il n'y a donc pas besoin de le recompiler pour qu'il fonctionne de manière à ce qu'il puisse être utilisable sans compétences particulières en informatique.

Néanmoins, si un utilisateur souhaitait tout de même le recompiler, il faudrait alors modifier le makefile en précisant le chemin du répertoire ou le fichier est enregistré, ligne 3 (qui commence par DIR).

```
poo > M Makefile
1 CC = g++
2 CFGLAS = -std=c++11
3 DIR = -I/home/simon/Bureau/gitNantes/poo
4
```

Le programme est lancé avec la commande shell `./main`. Avant d'accéder au menu, on demande à l'utilisateur s'il veut charger la base de données contenant les médicaments d'une précédente utilisation ou en créer une nouvelle. Néanmoins, si une base de données existe déjà et qu'on en crée une nouvelle que l'on sauvegarde, alors la nouvelle écrasera l'ancienne base de données existante.

```
$ ./main
Would you like to use the registered database (y/n) ? If the choice is 'no', the previous database will be overwritten if you save it when exiting the program.
```

Ensuite, l'utilisateur accède au menu avec les différentes options disponibles. A la fin de chaque option (sauf les options pour quitter le programme), l'utilisateur est renvoyé vers le menu principal.

```
*****
PHARMACY MANAGER

Menu:

(0) Print drug lists (by alphabetical order)
(1) Add a new drug
(2) Print all drug lots following expiration date
(3) Print drug lots of specific drug
(4) Print all drug from a specific laboratory
(5) Add a new drug lot to a specific drug
(6) Remove a drug
(7) Remove drug lot
(8) Sell drug
(9) Save
(10) Quit and save
(11) Quit without saving

*****
Please enter your choice here:
```

Il suffit de taper le chiffre de l'option choisie pour y accéder, par défaut l'option choisie est la (0).

(0) Print drug list (by alphabetical order:

Ce paramètre permet d'afficher les médicaments par ordre alphabétique. Un rappel du format d'affichage des descriptions de médicament est rappelé en amont.

```
Drug name
compositions
form -- indications -- drug administration -- contraindications -- laboratory name -- laboratory address -- autorisation number -- first autorisation -- renew autorisation

dafalgan
cristals -- 30
powder -- mouth uptake -- only mouth uptake! -- pfizer -- USA -- 42 -- 2000/02/02 -- 2000/03/03
```

(1) Add a new drug:

Cette option permet d'**ajouter un nouveau type de médicament** (une nouvelle instance de *Drug_description*) au catalogue du magasin. Attention, il ne s'agit pas ici de mettre à jour le stock (les quantités de médicaments disponibles ou non) mais simplement de préciser qu'un nouveau médicament doit être enregistré dans la base de données.

Chaque attribut doit être écrit à la suite, avec un format plus ou moins rigide imposé:

```
*****
Please enter your choice here:
0
Enter drug name
doliprane
Enter drug form
pills
```

Pour la composition, une liste de substances associée à une quantité est demandée en boucle jusqu'à ce qu'il n'y en ai plus à entrer. L'utilisateur peut alors quitter cette boucle en entrant "q" pour passer à l'attribut suivant:

```
enter q instead of name composition if you want to leave the loop of new composition element
Enter composition element :
paracetamol
Enter quantity element (integer only) :
10
enter q instead of name composition if you want to leave the loop of new composition element
Enter composition element :
water
Enter quantity element (integer only) :
20
enter q instead of name composition if you want to leave the loop of new composition element
Enter composition element :
q
```

Dans l'exemple ci-dessus, l'utilisateur a quitté la boucle après avoir entré deux substances ("paracetamol" et "water"). Il peut ensuite continuer à entrer d'autres attributs.

```
Enter drug indications
be carreful
Enter drug administration separated by ';'
drink it with water
Enter drug contraindications separated by ';'
don't take if you're pregnant
Enter laboratory name
pfizer
Enter laboratory address
1 plce de l'église, 75000, Paris
```

```
Enter autorisation number
757415
Please enter first autorisation date yyyy/mm/dd (example : 0010/01/01)
2006/05/02
Please enter renew autorisation date yyyy/mm/dd (example : 0010/01/01)
2008/05/03
```

(2) Print all drug lots following expiration date:

```
*****
Please enter your choice here:
2
Drug name -- expiration date -- manufacture date -- lot size -- id

levothyrox -- 2020/12/09 -- 2019/05/05 -- 56 -- RMR4521
dafalgan -- 2022/04/12 -- 2020/05/12 -- 40 -- MKDIR7854
levothyrox -- 2023/05/05 -- 2016/05/12 -- 5 -- MKDIR7458
dafalgan -- 2023/12/06 -- 2019/08/09 -- 25 -- PWD4521
```

Cette fonction permet l'affichage de tous les lots enregistrés dans la base de données en fonction de l'ordre d'expiration des dates. Le format est rappelé en amont de l'affichage des lots.

(3) Print all drug lots of specific drug:

```
*****
Please enter your choice here:
3
Please enter the drug name
levothyrox
Id -- expiration date -- manufacture date -- lot size
MKDIR7458 -- 2023/05/05 -- 2016/05/12 -- 5
RMR4521 -- 2020/12/09 -- 2019/05/05 -- 56

*****
```

Par cette fonction, il est possible d'afficher à l'écran tous les lots associés à une description de médicament. Le programme demandera alors le nom du médicament et se chargera d'afficher à l'écran comme ci-dessus. Le format est rappelé en amont de l'affichage des lots.

(4) Print all drug from a specific laboratory:

Pour un laboratoire donné en par l'utilisateur, le programme se chargera d'afficher tous les médicaments détenus par ce laboratoire, comme ceci:

```
Please enter laboratory name
pfizer
Drug name
compositions
form -- indications -- drug administration -- contraindications -- laboratory name -- laboratory address -- autorisation number -- first autorisation -- renew autorisation
dafalgan
cristals -- 30
powder -- mouth uptake -- only mouth uptake! -- pfizer -- USA -- 42 -- 2000/02/02 -- 2000/03/03

efferalgan
fromage -- 2thon hâché -- Spruneaux d'agen -- 6
pills -- eat it -- don't take before meals -- pfizer -- 16 rue des écoles, 44115 Haute Goulaine -- 654464567 -- 2009/12/12 --
```

(5) Add a new drug lot to a specific drug :

```
5
Please enter the drug name
levothyrox
Please enter id lot
GREP1524
Please enter the manufacture date, separated by yyyy/mm/dd
2018/05/05
Please enter the expiration date, separated by yyyy/mm/dd
2022/05/05
Please enter the lot size
10000
```

Pour une description de médicament donné, et seulement si elle existe dans la base de données, il est possible d'ajouter un lot de médicaments avec des paramètres entrés par l'utilisateur.

Le programme indique alors toutes les informations à rentrer à l'utilisateur, qui les entre alors un à un.

(6) Remove a drug:

```
*****  
Please enter your choice here:  
6  
Please enter a drug name  
levothyrox
```

Si une description de médicament est présente dans la base de données, il est alors possible de la supprimer en écrivant le nom de ce médicament. Si le médicament est présent il sera supprimé ainsi que les lots associés. Dans le cas contraire, un message indiquera à l'utilisateur que le nom entré ne correspond à aucun nom de description de médicament.

(7) Remove drug lot:

```
*****  
Please enter your choice here:  
7  
Please enter drug name  
dafalgan  
Please enter id lot  
PWD4521
```

En spécifiant un nom de description de médicament dans un premier temps puis un ID de lot dans un second temps, il est possible de supprimer un lot correspondant. Cette fonction avertira et cessera si un nom de description de médicament n'est pas présent dans la base de donnée, de même si l'id du lot n'est pas associé à cette description de médicament.

(8) Sell drug:

```
Please enter your choice here:
8
Please enter a drug name
dafalgan
How many boxes do you want to sell, should be an integer
3
```

Cette option permet de vendre un nombre spécifique d'un médicament. Les stocks seront alors mis à jour.

(9) Save:

```
Please enter your choice here:
9
```

Une fois cette option choisie, il est possible de sauvegarder la base de données. **Attention**, cette nouvelle sauvegarde supprimera tout autre sauvegarde présente auparavant et les changements sont irréversibles. De plus, un fichier out_of_stock est créé de la manière suivante minutes hours day month year dans le dossier out_of_stock.

(10) Quit and save:

```
Please enter your choice here:
10
Out of stock file has been set up
Closing application
```

Cette option permet la sauvegarde et la création d'un fichier out_of_stock, telle que l'option n°9, mais elle fait également quitter le logiciel. Ainsi, aucune perte de données n'arrivera.

(11) Quit without saving:

```
Please enter your choice here:
11
Closing application
```

Cette option permet de quitter sans sauvegarder les changements effectués depuis la dernière sauvegarde. Aucun fichier out_of_stock ne sera non plus créé. Cette option est donc à utiliser avec précaution.

5. Conclusion

5.1 Problèmes rencontrés:

Le principal problème rencontré a été l'enregistrement des médicaments de la pharmacie. L'enregistrement par **sérialisation** est vraiment adapté, mais la manière de l'implémenter était particulièrement complexe et peu intuitive. Toutefois, nous avons finalement réussi, mais c'est ce qui nous a pris le plus de temps dans le projet.

Un autre problème est l'utilisation de la fonction `year()` de la librairie `time` pour la création des fichiers `out_of_stock`. Nous aurions voulu que chaque nouveau fichier soit nommé avec l'heure et la date de sa génération. Concernant l'année, il y a un dysfonctionnement, cela affiche systématiquement "120".

5.2 Améliorations possibles

Plusieurs améliorations nous semblent possibles pour ce projet.

- Tout d'abord, il n'y a pas d'**interface graphique**, mais uniquement un menu textuel affiché à l'exécution du programme. Une interface graphique aurait rendu le logiciel plus agréable à utiliser, et même plus facile d'accès pour un utilisateur n'ayant aucune notion d'informatique.
- **Différents rôles d'utilisateurs**, avec des droits plus ou moins étendus (responsable, employé etc), avec une identification avec un mot de passe relié à une base de donnée auraient apporté une plus-value à notre travail
- Deux classes supplémentaires auraient aussi pu augmenter les possibilités offertes par le logiciel:
 - + **une classe client** pour archiver les clients et leurs commandes
 - + **une classe médecin** pour transférer automatiquement les prescriptions du médecin vers le pharmacien et ainsi éviter les falsifications de prescription par le patient/client.
 - + **une classe prescription** pour archiver indépendamment les prescriptions dont les médicaments ont été délivrés.
- Les entrées utilisateurs ne sont pas vérifiées, par exemple, l'entrée des dates se fait sur la bonne utilisation de l'utilisateur. Mais s'il le souhaite, il pourrait entrer la date suivante : 1111/20/88.
- La modification des attributs des classes `drug description` et `drug` ne sont pas modifiables a posteriori, il serait intéressant de pouvoir le faire,

notamment si l'utilisateur a fait une faute de frappe ou que des informations ont changé.

- Etant donné que la base de données est enregistrée, il est alors possible d'avoir plusieurs sauvegardes, il serait pertinent de faire une fonction qui précise en fonction de la date une nouvelle sauvegarde dans un dossier défini.