

# Travail pratique #5: Retructuration de code source

École Polytechnique de Montréal

Trimestre: Automne 2020

Équipier1: Sami Bourai #2041659

Équipier2: Fanilotiana Guy Randriamanalinarivo#2040625

**Équipe:312** 

Présenté à : William Harvey et Nicolas Verbaere

Polytechnique Montréal Remis le 1 décembre 2020

# E1) Une mauvaise odeur dans les attributs [/5]

La classe Appointments permet de gérer les informations afférentes aux différentsrendezvous. Elle possède un attribut de type AppointmentsData.

Examinez la classe « AppointmentsData ».

1) Identifiez le/les nom(s) des odeurs qui se cachent derrière celle-ci et expliquez pourquoi elles sont graves. [/0.5]

L'odeur qui s'y cache est l'odeur data classe. On juge que l'ajout de la classe AppointementData n'est absolument pas nécessaires, puisque les attributs de cette dernière aurait pu se trouver dans la classe Appointement. Le programmeur fait d'ailleurs appel à cette classe dans la classe Appointement.

- 2) Identifiez le/les nom(s) des restructurations nécessaires pour enlever ce/ces odeur(s)
- Déplacer les attributs et les méthodes de cette dernière dans la classe Appointements. De plus, comme ApointementData est inclut seulement dans Appointement cela ne devrait pas causer de problèmes lors de la modification. du code. [/0.5]
- 3) Identifiez les étapes que vous allez suivre pour restructurer ce/ces odeur(s). Utilisez

le même format que le tableau ci-dessous, dans lequel vous devez décomposer la restructuration globale en étapes plus simples. [/1]

5)Suppression de AppointementData

Etape Description	
Étapes	Description
1) Déplacer les attributs	On remplace _appointmentsData par -data
	dans le Appointement.h
2) Déplacer les méthodes	On remplace getData() et setDate
	dans le Appointements.h
3) Implémentation des nouvelles méthodes	On implimente adéquatement getData() et
dans Appointement.cp	setDate() dans le Appointements.cpp
	notamment en remplaçant l'opérateur de
	résolution de portée.
4)Changements d'attributs et de méthodes	On parcours tous le cpp afin de changer ou
dans Appointemet.cpp	supprimer appointmentsData car on a plus
	besoin de ce dernier pour avoir accès à nos
	setters et getters.

makefile

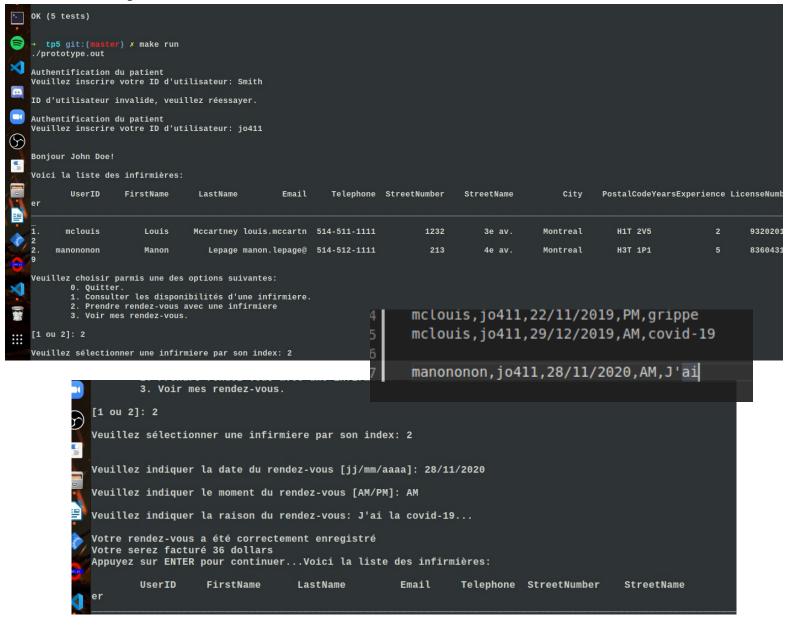
On supprime enfin le .h et le .cpp de cette

classe et on ajuste du même coup notre

4) Restructurez le code source et expliquez le raisonnement de vos changements dans le rapport. N'oubliez pas de soumettre les modifications sur Git. [/2]

une data classe est une odeur qui est décrite comme étant une classe à qui les attributs appartiennes à une autre classe. C'est ainsi le cas de la classe AppointementData qui aurait pu être implémenter dans la classe Appointement. Nous avons donc corrigé cela en implémentant les attributs et les méthodes dans la classe Appointement.cpp.

5) Compilez et essayez de créer un nouveau rendez-vous. Assurez-vous que ce nouveau rendez-vous soit dans la liste des rendez-vous. Veuillez ajouter des captures d'écrans des résultats de ces tests fonctionnels dans le rapport, afin de prouver la validité de vos modifications. [/1]



#### E2) Une mauvaise odeur dans les classes [/4]

qui sera facile a utiliser

Dans la structure actuelle, un « rendez-vous » est représenté comme étant un vecteur de string. La signification associée à chaque index de ce vecteur est définie dans le fichier Constants.hpp.

- 1) Est-ce la meilleure façon de définir un rendez-vous ? Justifiez votre réponse. [/1.5] Non, ce n'est pas la meilleure manière de definir un rendez-vous car on peut facilement se perdre avec le vecteur, et surtout il faudra toujours se référer au fichier Constant.hpp pour se repérer dans le vecteur.Le code est difficile a comprendre actuellement
- 2) Proposez, sans coder, une restructuration et justifier en quoi celle-ci est pertinente. Cette restructuration peut-elle s'appliquer dans d'autres fichiers ? [/1.5] la restructuration à ce problème est de regrouper les données du rendez-vous dans une classe ou l'on pourra avoir acces aux variables et permettra ainsi une lisibilité plus facile. Cette restructuration permettra alors d'éviter le vecteur de vecteur de string et laissera la place a un vecteur d'Appointment directement ce
- 3) Quel effet aura cette restructuration sur le fichier « Constant.hpp ». [1] Cette restructuration rendra alors le fichier Constants.hpp inutile, on n'aura plus besoin de ces références.

#### E3) Une mauvaise odeur dans les méthodes [/9]

- a) Examinez la methode void Appointments::display() /5
- 1) Identifier le/les nom(s) du/des odeurs qui se cache(nt) derrière celle-ci et expliquez en quoi ce sont des odeurs graves. [/0.5]v

Longueur de la methode : la methode est trop longue et fait beaucoup de choses trop de responsabilité on peut aussi dire une méthode Swiss Army Knife.

2) Identifiez le/les nom(s) des restructurations nécessaires pour enlever ce(s) odeur(s) du code. [/0.5]

Diviser le methode en plusieurs sous-methodes

3) Planifiez, étape par étape, comment restructurer ces odeurs, dans le même format que le tableau de l'exercice E1. [/1]

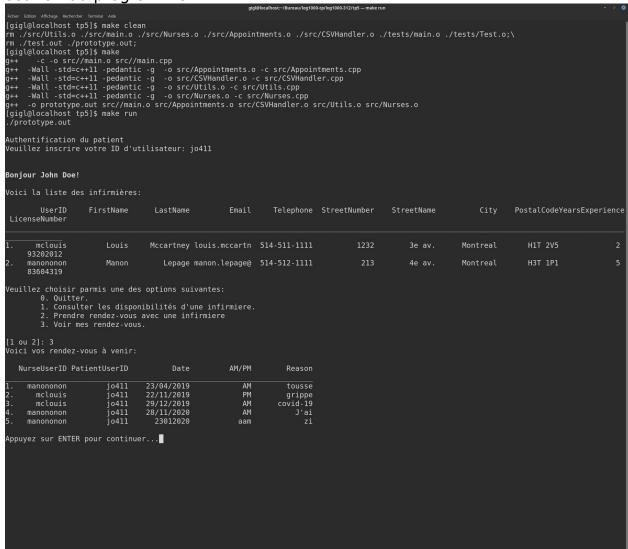
Étapes Description 1) Déclarer les nouvelles sous-méthodes On declare 2 nouvelles methodes: displayHeader et filterAppointments 2) Implémenter les nouvelles sous-Dans Appointment.cpp, on prend la partie 1 méthodes et la partie 2 de display pour displayHeader pour afficher l'entete, puis la partie 3 pour **filterAppointments** 3) Supprimer les lignes de la méthode Remplacer la partie 1 et 2 par l'appel de la display() fonction displayHeader et la partie 3 par **filterAppointments** 4)Deplacer les declarations des varibles Declarer le vecteur filteredAppointments juste avant leur utilisation juste avant l'appel de la methode filterAppointments ainsi que la variable AppointmentsIdx apres l'appel des 2 fonctions

4) Restructurez le code source de cette méthode et expliquez le raisonnement de vos changements dans le rapport. Mentionnez si vous avez créé des nouvelles méthodes ou modifié des méthodes existantes autre que display(). N'oubliez pas de soumettre les modifications sur Git. [/2]

On a cree 2 nouvelles methodes: displayHeader() et filterAppointments(). On a mis ces 2 methodes en private puisqu'elles seront seulement utiliser par la methode display qui etait tres complexe au depart. Effectivement, la methode display faisait beaucoup de choses à elle toutes seule, pour l'alleger, la methode displayHeader s'occupe de remplir le vecteur displayedColumns ainsi que l'affichage de l'entete tandis que la methode filterAppointments prend en charge le filtrage des rendez-vous.

5) Compilez et exécutez votre logiciel. Veuillez ajouter des captures d'écrans des résultats de tests fonctionnels dans le rapport qui prouve la validité des modifications que vous avez effectués. [/1]

Execution du programme :



#### Test:

```
[gigl@localhost tp5]$ make test
g++ -c -o tests/main.o tests/main.cpp
g++ -wall -std=c++11 -pedantic -g -o tests/Test.o -c tests/Test.cpp
g++ -o test.out tests/main.o tests/Test.o src/Appointments.o src/CSVHandler.o src/Utils.o src/Nurses.o -I./ -lcppunit
[gigl@localhost tp5]$ make run_test
./test.out
.....

OK (5 tests)
[gigl@localhost tp5]$
```

- b) En examinant la méthode int Appointments::getHoursRate(), vous remarquez probablement certaines odeurs dans le code. /4
- 1) Identifier les odeurs [/1]
- -Manque de clarte sur le nom de varable : years
- -Trop de conditions if qui s'entremelent
- -condition if inutile (if years<=2) puisque cela ne change pas rate
- variable years non initialisé avant utilisation
- 2) Restructurez le code source de cette méthode et expliquez le raisonnement de vos changements dans le rapport. Mentionnez si vous avez créé des nouvelles méthodes ou modifié des méthodes existantes autre que getHoursRate(). N'oubliez pas de soumettre les modifications sur Git. [/2]
- Renommer years comme yearsExperience ce qui sera plus explicite sur l'utilite de cette variable
- Si patientDisease== « covid-19 », on va mettre return 2 comme quoi, le reste de la methode sera consideree comme une condition else
- -On initialise yearsExperience avant de l'utiliser
- On retire la condition if years<=2, on va mettre yearsExperience>2 && yearsExperience<5
- on ajoute une else if(yearsExperience>5) pour modifier le rate
- 3) Compilez et exécutez votre logiciel. Veuillez ajouter des captures d'écrans des résultats de tests fonctionnels dans le rapport qui prouve la validité des modifications que vous avez effectués. [/1]

[gigl@localhost tp5]\$ make test
g++ -c -o tests/main.o tests/main.cpp
g++ -c -o tests/main.o tests/main.cpp
g++ -wall -std=c++11 -pedantic -g -o tests/Test.o -c tests/Test.cpp
g++ -o test.out tests/main.o tests/Test.o src/Appointments.o src/CSVHandler.o src/Utils.o src/Nurses.o -I./ -lcppunit
[gigl@localhost tp5]\$ make run\_test
./test.out
.....

OK (5 tests)

[gigl@localhost tp5]\$

#### E3) Utilisation des variables [/5]

Afin d'améliorer la compréhension et la lisibilité du code, il est important de minimiser le span, la durée de vie et la portée des différentes variables :

1) Calculez le span moyen, la durée de vie et la portée des variables choice et loggedInUser dans la méthode main() dans main.cpp. Ne comptez pas les lignes vides. Les accolades et les commentaires comptent comme des lignes. [/0.5] choice

span: 24.3

durée de vie :74 lignes

portée: 79

#### loggedInUser

span : 11

durée de vie :74 lignes

portée: 79

- 2) Interprétez les résultats, et trouvez la variable (parmi les deux citées en dessus) qui bénéficiera le plus de la restructuration. Justifiez [/0.5] en somme, la durée de vie et la portée des deux objets sont équivalent. Quant au span choice est plus dipersée que loggedInUser. Donc, c'est choice qui en bénéficiera le plus.
- 3) Proposez des restructurations de votre fonction main pour améliorer l'utilisation de cette variable, en utilisant le même format du tableau de l'exercice E1. Notez qu'il faut tout autant minimiser le span, que la durée de vie et la portée. [/1]

# Étapes

- 1) Initialisation juste avant utilisation
- 2) Utiliser do while en while

#### Description

On va déclarer et initialiser choice juste avant son utilisation de sorte à réduire span, durée de vie et portée. En transformant le while en do while on gagne énormément et cela au niveau de span, durée de vie et portée. 4) Effectuez cette restructuration dans la méthode « main » et recalculez les nouvelles métriques. N'oubliez pas de soumettre les modifications sur Git. [/2]

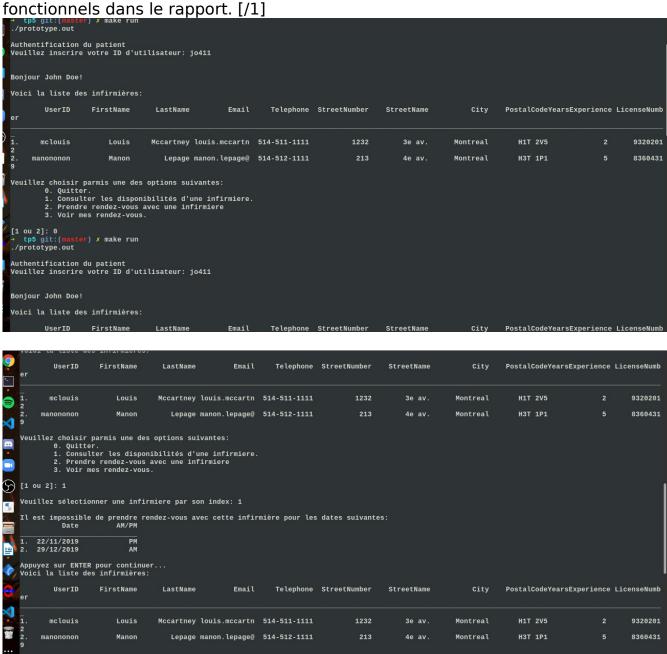
# Recalculer : <a href="mailto:choice">choice</a>

span: 3.6

durée de vie :14 lignes

portée: 79 main

5) Compilez et testez manuellement les opérations (de l'opération 0 à 3) de la méthode « main ». Veuillez ajouter des captures d'écrans de vos tests



Veuillez choisir parmis une des options suivantes:

0. Quitter. 1. Consulter les disponibilités d'une infirmiere.

```
[1 ou 2]: 2
Veuillez sélectionner une infirmiere par son index: 1
Veuillez indiquer la date du rendez-vous [jj/mm/aaaa]: 20/02/1997
Veuillez indiquer le moment du rendez-vous [AM/PM]: am
Veuillez indiquer la raison du rendez-vous: moula
Votre rendez-vous a été correctement enregistré
Votre serez facturé 20 dollars
Appuyez sur ENTER pour continuer...
Voici la liste des infirmières:
        UserID FirstName LastName
                                                               Email
                                                                        Telephone StreetNumber StreetName
                                                                                                                                    City PostalCodeYearsExperience LicenseNumb
                                                                                                                                                                                       9320201
     manononon
                          Manon
                                         Lepage manon.lepage@ 514-512-1111
                                                                                                                 4e av. Montreal
                                                                                                                                                   H3T 1P1
                                                                                                                                                                                        8360431
Veuillez choisir parmis une des options suivantes:
         0. Quitter.
1. Consulter les disponibilités d'une infirmiere.
2. Prendre rendez-vous avec une infirmiere
3. Voir mes rendez-vous.
[1 ou 2]: 3
Voici vos rendez-vous à venir:
   NurseUserID PatientUserID
                                                                                Reason
                           j0411 23/64/2019
j0411 22/11/2019
j0411 29/12/2019
j0411 28/11/2020
j0411 23012829
       mclouis
```

# **E4)** Tests [/2]

À la remise, assurez-vous que les commandes make et make test fonctionnent correctement. Assurez-vous que tous les tests unitaires soient valides et effectuez des tests d'acceptation sur le programme, afin de vous assurer que vous n'ayez introduit aucunerégression logicielle. Il n'est pas nécessaire de fournir de capture d'écran, mais le tout sera testé lors de la correction et devra être fonctionnel.