École Polytechnique de Montréal

Département de génie informatique et génie logiciel

LOG1000 – Ingénierie logicielle

**Travail Pratique 5 : Restructuration de code source**

**Objectifs :**

* **Identifier** les mauvaises odeurs dans le code source (« bad smells »)nécessitant

une restructuration, sur plusieurs niveaux.

* **Restructurer** le code ayant des mauvaises odeurs en utilisant desrestructurations adéquates, et ce, en plusieurs phases consécutives.
* **Compiler et tester** le code restructuré après chaque phase pour valider laviabilité des nouvelles implémentations.

**Notes importantes :**

* Donnez des réponses courtes, claires et précises.
* Donnez votre réponse sous forme de liste quand cela est possible.
* Vérifier la lisibilité des captures d’écrans que vous allez mettre dans votre rapport.
* **Le rapport sera remis dans un dossier nommé TP5 dans votre répertoire Git le 12 avril avant midi (12h00). Il y aura une pénalité de 10% par jour de retard.**

**Répartition des points de l’évaluation [/80]:**

Respect des formats et consignes demandées pour la rédaction du rapport**. [/4]**

**Partie 1 :**

**E1) Une mauvaise odeur dans les attributs [/30]**

1. Il y a un groupe d’attributs qu’on pourrait regrouper sur une autre classe et on pourrait penser que la classe a trop de responsabilité, car son nombre d’attributs est élevé.

[/2]

1. Extraire la classe.

[/2]

3)string getChaineName();

string getChaineCodePostal();

string getChaineAddress();

Emission(string, string, string, string, string);

void associerChaine(string, string, string);

trouverEmission (string DB, string titre);

afficher();

[/3]

4) chaineName, chaineCodePostal, chaineAddress

[/6]

5)

|  |  |  |
| --- | --- | --- |
|  | Étape | Description |
| 1 | Créer une classe | Créer une nouvelle classe chaine. |
| 2 | Définir et implémenter la nouvelle classe | Déplacer les méthodes et les attributs pertinents dans  La nouvelle classe. |

[/4]

6) Chaine.h

#ifndef Chaine\_H

#define Chaine\_H

#include <string>

using namespace std;

class Chaine{

public:

// Constructeurs

Chaine() {};

Chaine(string, string, string);

string getChaineName();

string getChaineCodePostal();

string getChaineAddress();

private:

// Informations sur la chaine de l'emission

string chaineNom\_;

string chaineCodePostal\_;

string chaineAddresse\_;

};

#endif

Chaine.cpp

#include "Chaine.h"

Chaine::Chaine(string nom, string code, string adresse) {

chaineNom\_ = nom;

chaineCodePostal\_ = code;

chaineAddresse\_ = adresse;

}

string Chaine::getChaineName() {

return chaineNom\_;

}

string Chaine::getChaineCodePostal() {

return chaineCodePostal\_;

}

string Chaine::getChaineAddress() {

return chaineAddresse\_;

}

Emission.h

#ifndef Emission\_H

#define Emission\_H

#include "Chaine.h"

// Cette classe représente un Emission

class Emission {

public:

// Constructeurs

Emission () {}

Emission(string, string, string, string, string);

// Setters

void setTitre(string);

void setAnimateur(string);

void associerChaine(string, string, string);

// Getters

string getTitre();

string getAnimateur();

string getChaineName();

string getChaineCodePostal();

string getChaineAddress();

// Enregistrer l'Emission

void saveEmission(string);

// Afficher l'Emission

void afficher();

// Chercher un Emission dans une base de données par titre

Emission\* trouverEmission(string, string);

private:

// Information sur l'emission

string titre;

string animateur;

Chaine chaine;

};

#endif

Emission.cpp

#include "Emission.h"

#include <fstream>

#include <iostream>

// Constructeur

Emission::Emission (string titre,

string animateur,

string chaineName,

string chaineCodePostal,

string chaineAddress) {

// Emission information

this->titre = titre;

this->animateur = animateur;

// Chaine information

this->chaine.getChaineName() = chaineName;

this->chaine.getChaineCodePostal() = chaineCodePostal;

this->chaine.getChaineAddress() = chaineAddress;

}

// Setters

void Emission::setTitre(string titre) {

this->titre = titre;

}

void Emission::setAnimateur(string animateur) {

this->animateur = animateur;

}

// Associer un chaine à l'Emission

void Emission::associerChaine (string chaineName,

string chaineCodePostal,

string chaineAddress) {

this->chaine.getChaineName() = chaineName;

this->chaine.getChaineCodePostal() = chaineCodePostal;

this->chaine.getChaineAddress() = chaineAddress;

}

// Getters

string Emission::getTitre() {

return this->titre;

}

string Emission::getAnimateur() {

return this->animateur;

}

string Emission::getChaineName() {

return this->chaine.getChaineName();

}

string Emission::getChaineCodePostal() {

return this->chaine.getChaineCodePostal();

}

string Emission::getChaineAddress() {

return this->chaine.getChaineAddress();

}

// Enregistrer l'Emission dans un fichier

void Emission::saveEmission (string fileName) {

ofstream outfile (fileName.c\_str(), std::ofstream::binary | std::fstream::app);

// write to outfile

outfile<<this->titre <<","

<<this->animateur <<","

<<this->chaine.getChaineName() << ","

<< this->chaine.getChaineCodePostal() << ","

<< this->chaine.getChaineAddress() <<"\n";

outfile.close();

}

// Trouver un Emission avec son nom dans la base de données DB

Emission\* Emission::trouverEmission (string DB, string titre) {

ifstream fichier(DB.c\_str(), ios::in); // Ouvrir le fichier "DB.txt"

Emission\*tmp=NULL;

if(fichier)

{

string line;

// Lire les Emissions, un Emission par ligne dans la base de données (DB.txt)

while (getline(fichier, line)) {

string titreDB;

// Récupérer le nom de l'Emission

int i = 0;

for (i = 0 ; i < line.length() ; i++) {

if (line[i] != ',') {

titreDB += line[i];

} else {

break;

}

}

// Si l'Emission qu'on lit actuellement est celui qu'on cherche

if (titreDB == titre) {

// Récupérer le nom de l'animateur

string animateurDB;

for (i = i + 1; i < line.length() ; i++) {

if (line[i] != ',') {

animateurDB += line[i];

} else {

break;

}

}

// Récupérer le nom de l'éditeur

string chaineNameDB;

for (i = i + 1; i < line.length() ; i++) {

if (line[i] != ',') {

chaineNameDB += line[i];

} else {

break;

}

}

// Récupérer le code postale de l'éditeur

string chaineCodePostalDB;

for (i = i + 1; i < line.length() ; i++) {

if (line[i] != ',') {

chaineCodePostalDB += line[i];

} else {

break;

}

}

// Récupérer l'addresse de l'éditeur

string chaineAddressDB;

for (i = i + 1; i < line.length() ; i++) {

if (line[i] != ',') {

chaineAddressDB += line[i];

} else {

break;

}

}

// Créer un objet de type Emission avec les informations récupérées

Emission \*a = new Emission(titreDB, animateurDB, chaineNameDB, chaineCodePostalDB, chaineAddressDB);

// Fermer la base de données

fichier.close();

// Retourner l'Emission sélectionné

return a;

}

}

// Fermer la base de données

fichier.close();

}

// Si l'Emission est innexistant, on retourne NULL

return NULL;

}

// Afficher l'Emission

void Emission::afficher() {

std::cout << "Titre : " << this->titre << std::endl;

std::cout << "Animateur : " << this->animateur << std::endl;

std::cout << "Chaine name : " << this->chaine.getChaineName() << std::endl;

std::cout << "Chaine code postale : " << this->chaine.getChaineCodePostal() << std::endl;

std::cout << "Chaine address : " << this->chaine.getChaineAddress() << std::endl;

}

[/6]

7)

[/5]

8)

[/2]

**E2) Une mauvaise odeur dans les méthodes [/20]**

1) Les deux odeurs graves ici sont que le code est dupliqué et que qu’il y a du code mort. Le fait qu’il y a du code mort est une odeur grave, car il y a une variable « tmp » créé au début. Le fait que le code est dupliqué est une odeur grave, car il faudra faire plusieurs modifications. Identifiez le nom des deux odeurs graves et expliquez pourquoi ce sont des odeurs graves.

[/2]

2)

|  |  |  |
| --- | --- | --- |
|  | Étape | Description |
| 1 | Créer une méthode  récupérer() |  |
| 2 | Remplacer chaque  boucle for. |  |

[/6]

3) [/6]

4) [/4]

5) [/2]

**E3) Utilisation des variables [/26]**

1) [/6]

2) [/3]

3) [/5]

4) [/5]

5) [/5]

6) [/2]

**Énoncé :**

**Partie 1 :**

Dans cet énoncé, vous devez identifier et restructurer les mauvaises odeurs présentes dans un programme C++ (dans le dossier **source**). Le groupe TVA a un composant logiciel qui permet de gérer ses émissions. Ce groupe a jugé que le code nécessite certaines restructurations, dans le but qu’il soit maintenable et fiable à long terme, et ainsi ils font appel à vous pour le restructurer.

Cette restructuration du code consiste à modifier la structure interne du logiciel pour le rendre plus facile à comprendre et à modifier, évidemment SANS changer son comportement observable !

* + Dans le logiciel, le fichier « main.cpp » permet de proposer à un utilisateur les opérations possibles qu’il peut effectuer, par exemple, ajouter et enregistrer une émission.
  + Les émissions sont gérées par la classe « Emission » qui est définie dans le fichier «Emission.h » et implémentée dans le fichier « Emission.cpp ».
  + Pour effectuer des tests sur le programme, donne une base de données enregistrée dans le fichier «DB.txt ».

La classe «Emission» a besoin d’être restructurée dans E1 et E2, tandis que E3 vise à restructurer la méthode « main ».

Pour plus de détails sur la restructuration du code, veuillez consulter le lien suivant : http://www.professeurs.polymtl.ca/michel.gagnon/Smells/

**E1) Une mauvaise odeur dans les attributs [/27]**

En examinant les attributs de la classe « Emission », vous pouvez remarquer qu’elle est faiblement cohésive, parce qu’il y a des attributs représentant des informations sur la chaîne et d’autres attributs qui représentent des informations sur une émission.

1) Expliquez pourquoi c’est une odeur grave. **[/2]**

2) Identifiez le nom de la restructuration nécessaire pour enlever cette odeur du code. **[/2]**

3) Identifiez les méthodes qui seront impactées par ce changement. **[/3]**

4) Identifiez les attributs qu’il faut modifier ou déplacer. **[/6]**

5) Identifiez les étapes que vous allez suivre pour restructurer cette odeur. Utilisez le même format du tableau ci‐dessous, dans lequel vous décomposez la restructuration globale en étapes plus simples. Il est important que vos tests fonctionnent après vos restructurations **[/4]**

|  |  |
| --- | --- |
| Étape | Description |
|  |  |

6) Restructurez le code source en modifiant/déplaçant les attributs de la question 4 et en modifiant les méthodes impactées (question 3). (Notez que vous avez le droit de créer une nouvelle classe en cas de besoin). Copiez la/les classe(s) modifiée(s) dans le rapport (le header et le cpp files). **[/6]**

7) Compilez et exécutez les tests unitaires fournis (/tests/…), veuillez ajouter des captures d’écrans des résultats de vos tests dans le rapport. **[/5]**

8) Faites un commit de votre code source, et une capture d’écran de ce commit dans le rapport (Le résultat de la commande « git add, commit, et push »). **[/2]**

**E2) Une mauvaise odeur dans les méthodes [/18]**

En examinant la méthode « TrouverEmission» de la classe « Emission.cpp » :

1) Identifiez le nom des deux odeurs graves et expliquez pourquoi ce sont des odeurs graves. **[/2]**

2) Planifiez, étape par étape, comment restructurer cette odeur, dans le même format du tableau de l’exercice E1. **[/6]**

3) Restructurez le code source de cette méthode. Copiez dans le rapport le nouveau code de la méthode, ainsi que d’autres méthodes si vous en créez des nouvelles ou si vous modifiez d’autres méthodes dans cette restructuration. **[/6]**

4) Compilez et exécutez les tests unitaires fournis (/tests/…), veuillez ajouter des captures d’écrans des résultats de vos tests dans le rapport. **[/4]**

5) Faites un commit de votre code source, et une capture d’écran de ce commit dans le rapport. **[/2]**

**E3) Utilisation des variables [/26]**

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, la durée de vie et la portée des variables « DBFile », « choix » et « emission » dans la méthode « main » dans « main.cpp ». Les lignes vides ne comptent pas ! **[/6]**

2) Interprétez les résultats, et trouvez la variable (parmi les trois citées en dessus) qui bénéficiera le plus de la restructuration. **[/3]**

3) Proposez des restructurations pour améliorer l’utilisation de cette variable, en utilisant le même format du tableau de l’exercice E1. **[/5]**

4) Effectuez cette restructuration dans la méthode « main ». Faites une capture d’écran (ou copiez le code) de votre nouveau code source. **[/5]**

5) Compilez et testez manuellement les opérations (de l’opération 0 à 4) de la méthode « main », veuillez prendre des captures d’écrans de vos tests. **[/5]**

6) Faites un commit de votre code source, et une capture d’écran de ce commit dans le rapport. **[/2]**