

Travail pratique # 5 : Restructuration du code source

École polytechnique de Montréal

Trimestre : automne 2017

Équipier1 : Georges Louis 1880098

Équipier2 : Mazigh Ouanes 1721035

Équipe numéro : 20

Date de remise (4-12-2017)

E1) Une mauvaise odeur dans les attributs

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

La Classe « Emission.h » est une mauvaise odeur parce qu'elle a un grand nombre d'attributs. C'est une classe qui agit comme 2 sous classes.

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

La restructuration nécessaire pour enlever cette odeur est l'extraction d'une classe. Cette dernière consiste à créer une nouvelle classe (Class Chaîne) et déplacer les attributs et méthodes pertinents vers la nouvelle classe.

3) Identifiez les méthodes qui seront impactées par ce changement. [/6]

Les méthodes qui seront impactées par ce changement sont celles que notre nouvelle Class Chaîne aura besoin et par la suite notre Class « Emission.h » va les perdre :

- Chaîne(string, string, string); méthode ajoutée
- Chaîne(); méthode ajoutée
- string getChaîneName();
- string getChaîneCodePostal();
- string getChaîneAddress();

Donc, il faut faire un changement dans quelques méthodes de la Class « Emission.h ». Les méthodes qui seront modifiées sont les suivantes :

- void associerChaîne(Chaîne*)
- Chaîne* getChaîne(); méthode ajoutée

Identifiez les attributs qu'il faut modifier ou déplacer. [/3]

Les attributs qui seront déplacé de la classe « Emission.h » vers la Class Chaîne sont :

- string chaineName;
- string chaineCodePostal;
- string chaineAddress;

Pour la classe Émission, il y a un seul attribut qui faut ajouter et qui est le suivant :

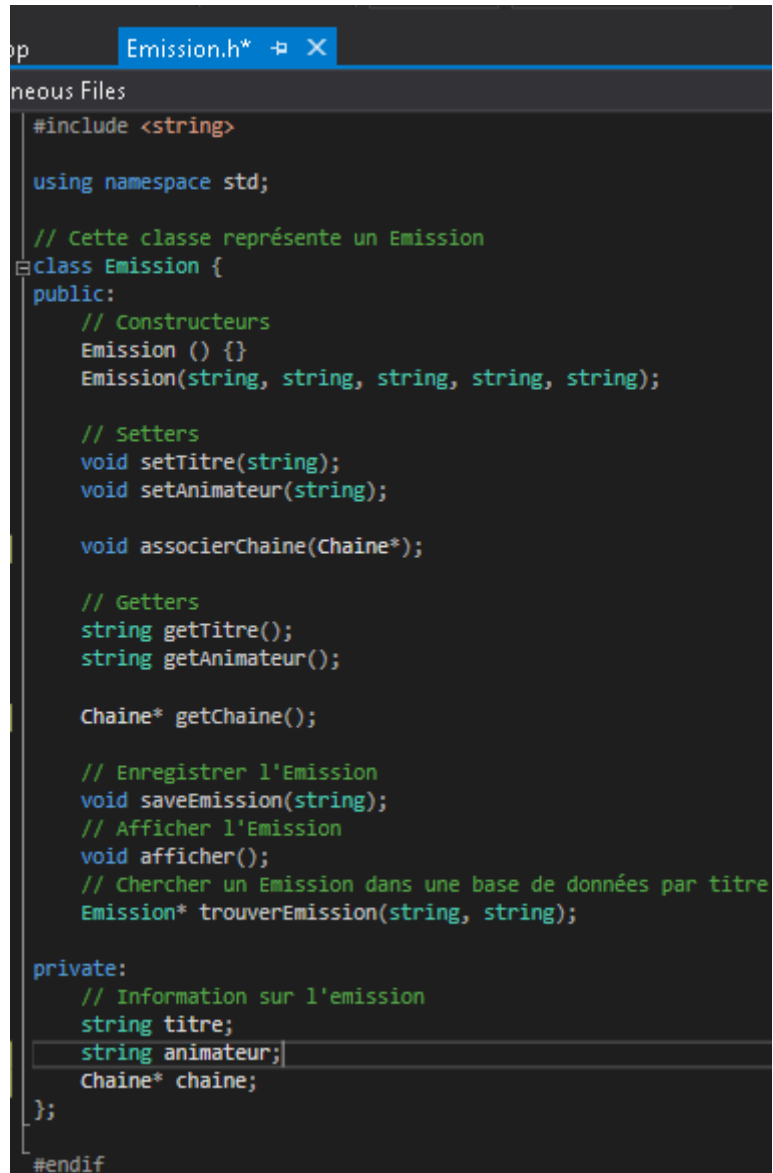
❏ Chaîne* chaine;

Identifiez les étapes à suivre pour restructurer cette odeur. Utilisez le même format du tableau cidessous, dans lequel vous décomposez la restructuration globale en étapes plus simples [/5]

Étape	Description
Créer une nouvelle classe Chaîne	Créer un fichier Chaîne.h et Chaîne.cpp
Écrire le fichier Chaîne.h	Transférer les définitions des méthodes et des attributs dans le fichier Chaîne.h.
Écrire le fichier Chaîne.cpp	Écrire les implémentations des méthodes et des attributs de la classe Chaîne.h
Modifier la classe Emission.h	Enlever les définitions des méthodes et des attributs qui font parties de la classe Chaîne.h
Modifier la classe Emission.cpp	Enlever les implémentations des méthodes et des attributs qui font parties de la classe Chaîne.cpp

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 Étape Description le droit de créer une nouvelle classe en cas de besoin). Copiez la/les classe(s) modifiée(s) dans le rapport (le fichier header et cpp). [/6]

Emission.h

The image shows a code editor window with a tab labeled 'Emission.h*'. The editor contains the C++ header file for the 'Emission' class. The code includes a preprocessor directive for the string library, a namespace declaration for 'std', and a class definition. The class has public methods for constructors, setters, getters, and database operations, as well as private attributes for title, animator, and channel. The code is syntax-highlighted with colors: blue for keywords, green for comments, and black for identifiers and literals. The editor interface includes a 'Recent Files' list on the left and a tab bar at the top.

```
#include <string>

using namespace std;

// 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(Chaine*);

    // Getters
    string getTitre();
    string getAnimateur();

    Chaine* getChaine();

    // 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

```
Chaine.cpp    Chaine.h    Emission.cpp  X  Emission.h
Miscellaneous Files  → Emission

1  #include "Emission.h"
2
3  #include <fstream>
4  #include <iostream>
5
6  // Constructeur
7  Emission::Emission (string titre,
8                      string animateur,
9                      string chaineName,
10                     string chaineCodePostal,
11                     string chaineAddress) {
12
13     // Emission information
14     this->titre = titre;
15     this->animateur = animateur;
16     this->chaine = new Chaine(chaineName, chaineCodePostal, chaineAddress);
17     // Chaine information
18     this->chaineName = chaineName;
19     this->chaineCodePostal = chaineCodePostal;
20     this->chaineAddress = chaineAddress;
21 }
22
23 // Setters
24 void Emission::setTitre(string titre) {
25     this->titre = titre;
26 }
27 void Emission::setAnimateur(string animateur) {
28     this->animateur = animateur;
29 }
30
31 // Associer un chaine à l'Emission
32 void Emission::associerChaine (Chaine* shaine) {
33     this->chaine = shaine;
34 }
35
36
37 // Getters
38 string Emission::getTitre() {
39     return this->titre;
40 }
41 string Emission::getAnimateur() {
42     return this->animateur;
43 }
44
45 string Emission::getChaine() {
46     return this->chaine;
47 }
48
49 // Enregistrer l'Emission dans un fichier
50 void Emission::saveEmission (string fileName) {
51
```

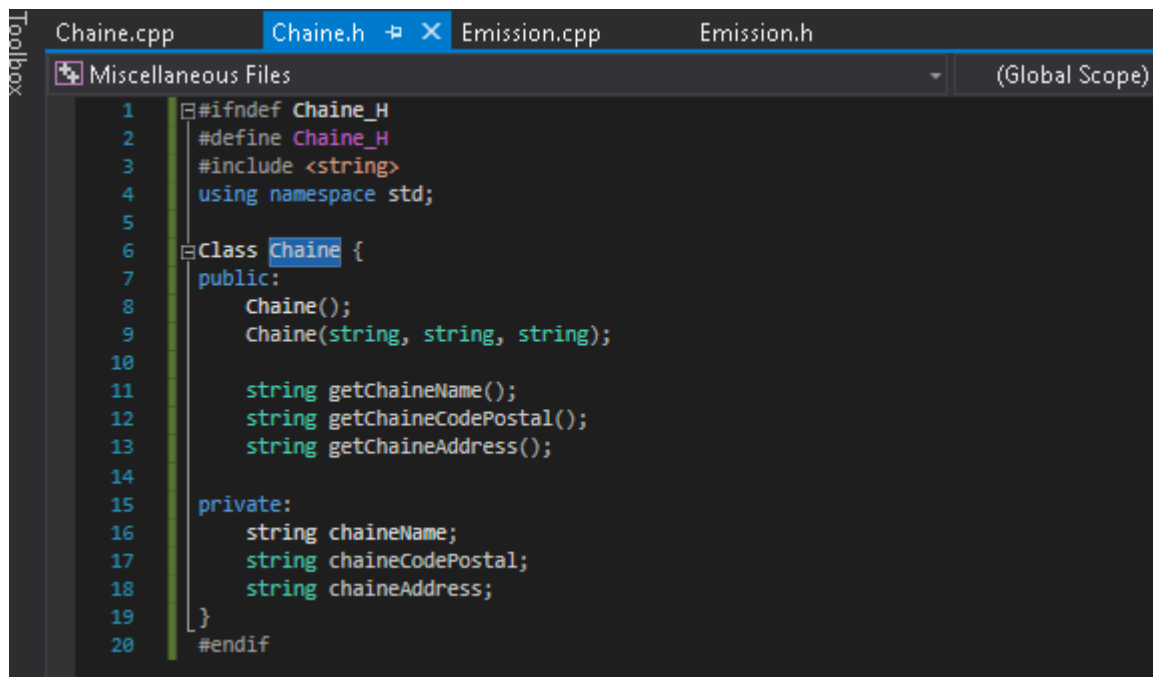
```
Chaine.cpp  Chaine.h  Emission.cpp  X  Emission.h
Miscellaneous Files  -  → En

49  // Enregistrer l'Emission dans un fichier
50  void Emission::saveEmission (string fileName) {
51  |
52  |     ofstream outfile (fileName.c_str(), std::ofstream::binary | std::fstream::app);
53  |     // write to outfile
54  |     outfile << this->titre << ","
55  |         << this->animateur << endl;
56  |
57  |
58  |     outfile.close();
59  | }
60  |
61  // Trouver un Emission avec son nom dans la base de données DB
62  Emission* Emission::trouverEmission (string DB, string titre) {
63  |
64  |     ifstream fichier(DB.c_str(), ios::in); // Ouvrir le fichier "DB.txt"
65  |     Emission*tmp=NULL;
66  |
67  |     if(fichier)
68  |     {
69  |         string line;
70  |         // Lire les Emissions, un Emission par ligne dans la base de données (DB.txt)
71  |         while (getline(fichier, line)) {
72  |             string titreDB;
73  |             // Récupérer le nom de l'Emission
74  |             int i = 0;
75  |             for (i = 0 ; i < line.length() ; i++) {
76  |                 if (line[i] != ',') {
77  |                     titreDB += line[i];
78  |                 } else {
79  |                     break;
80  |                 }
81  |             }
82  |
83  |             // Si l'Emission qu'on lit actuellement est celui qu'on cherche
84  |             if (titreDB == titre) {
85  |                 // Récupérer le nom de l'animateur
86  |                 string animateurDB;
87  |                 for (i = i + 1; i < line.length() ; i++) {
88  |                     if (line[i] != ',') {
89  |                         animateurDB += line[i];
90  |                     } else {
91  |                         break;
92  |                     }
93  |                 }
94  |
95  |                 // Récupérer le nom de l'éditeur
96  |                 string chaineNameDB;
97  |                 for (i = i + 1; i < line.length() ; i++) {
```

```
Chaine.cpp      Chaine.h      Emission.cpp  X Emission.h
Miscellaneous Files  → Emission

100         if (line[i] != ',') {
101             chaineNameDB += line[i];
102         } else {
103             break;
104         }
105     }
106
107     // Récupérer le code postale de l'éditeur
108     string chaineCodePostalDB;
109     for (i = i + 1; i < line.length() ; i++) {
110         if (line[i] != ',') {
111             chaineCodePostalDB += line[i];
112         } else {
113             break;
114         }
115     }
116
117     // Récupérer l'adresse de l'éditeur
118     string chaineAddressDB;
119     for (i = i + 1; i < line.length() ; i++) {
120         if (line[i] != ',') {
121             chaineAddressDB += line[i];
122         } else {
123             break;
124         }
125     }
126
127     // Créer un objet de type Emission avec les informations récupérées
128     Emission *a = new Emission(titreDB, animateurDB, chaineNameDB, chaineCodePostalDB, chaineAddressDB);
129     // Fermer la base de données
130     fichier.close();
131     // Retourner l'Emission sélectionné
132     return a;
133 }
134
135 // Fermer la base de données
136 fichier.close();
137 }
138 // Si l'Emission est inexistant, on retourne NULL
139 return NULL;
140
141 }
142
143 // Afficher l'Emission
144 void Emission::afficher() {
145     std::cout << "Titre : " << this->titre << std::endl;
146     std::cout << "Animateur : " << this->animateur << std::endl;
147     std::cout << "Chaine name : " << (this->chaine)->getChaineName() << std::endl;
148     std::cout << "Chaine code postale : " << (this->chaine)->getChaineCodePostal() << std::endl;
149     std::cout << "Chaine address : " << (this->chaine)->getChaineAddress() << std::endl;
150 }
```

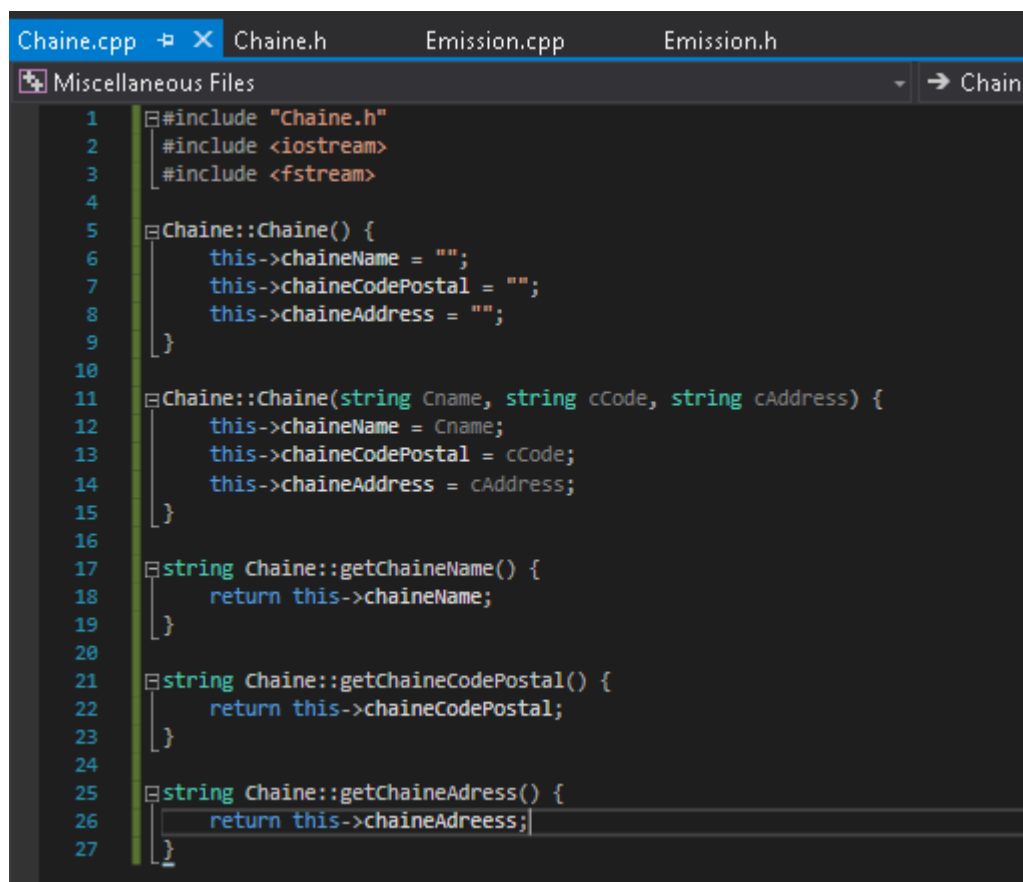
Chaine.h



The screenshot shows the Chaine.h header file in a code editor. The editor has tabs for Chaine.cpp, Chaine.h (active), Emission.cpp, and Emission.h. The left sidebar shows 'Miscellaneous Files' and '(Global Scope)'. The code defines the Chaine class with public methods and private attributes.

```
1  #ifndef Chaine_H
2  #define Chaine_H
3  #include <string>
4  using namespace std;
5
6  class Chaine {
7  public:
8      Chaine();
9      Chaine(string, string, string);
10
11     string getChaineName();
12     string getChaineCodePostal();
13     string getChaineAddress();
14
15 private:
16     string chaineName;
17     string chaineCodePostal;
18     string chaineAddress;
19 }
20 #endif
```

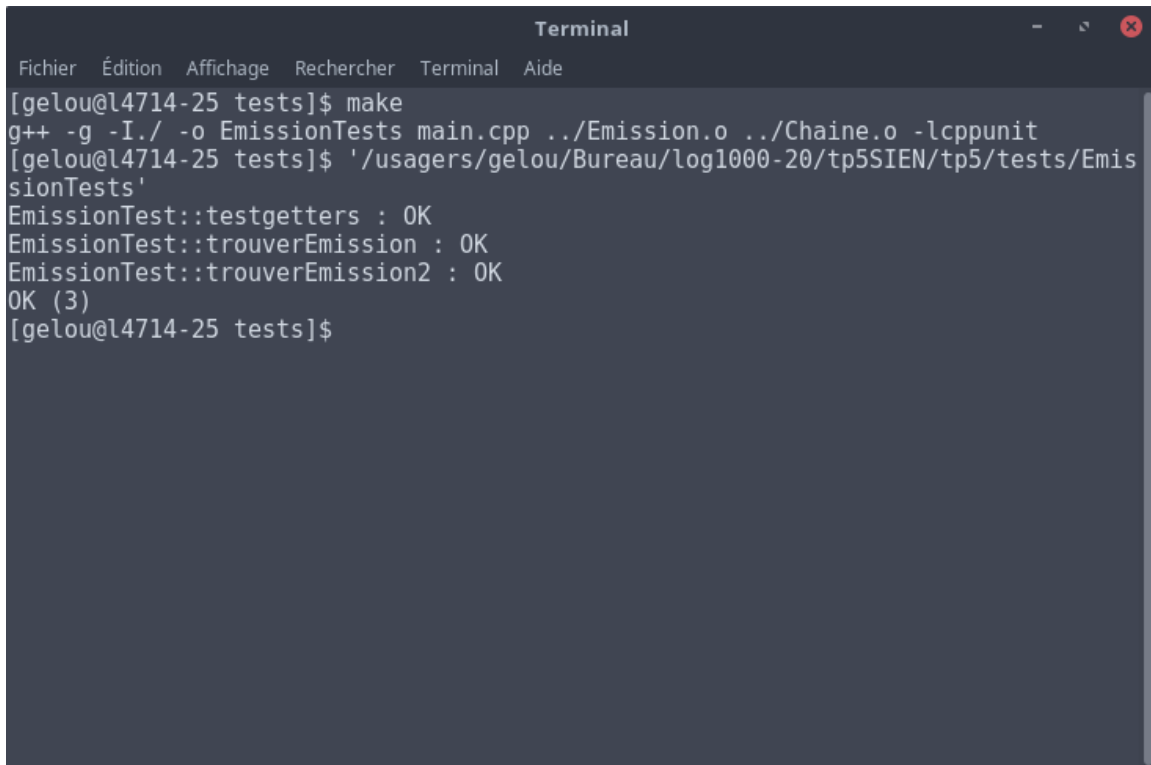
Chaine.cpp



The screenshot shows the Chaine.cpp implementation file in a code editor. The editor has tabs for Chaine.cpp (active), Chaine.h, Emission.cpp, and Emission.h. The left sidebar shows 'Miscellaneous Files' and a button to 'Chain'. The code implements the Chaine class methods.

```
1  #include "Chaine.h"
2  #include <iostream>
3  #include <fstream>
4
5  Chaine::Chaine() {
6      this->chaineName = "";
7      this->chaineCodePostal = "";
8      this->chaineAddress = "";
9  }
10
11  Chaine::Chaine(string Cname, string cCode, string cAddress) {
12      this->chaineName = Cname;
13      this->chaineCodePostal = cCode;
14      this->chaineAddress = cAddress;
15  }
16
17  string Chaine::getChaineName() {
18      return this->chaineName;
19  }
20
21  string Chaine::getChaineCodePostal() {
22      return this->chaineCodePostal;
23  }
24
25  string Chaine::getChaineAddress() {
26      return this->chaineAdreess;
27  }
```

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

A terminal window titled "Terminal" with a menu bar containing "Fichier", "Édition", "Affichage", "Rechercher", "Terminal", and "Aide". The terminal shows the following commands and output:

```
[gelou@l4714-25 tests]$ make
g++ -g -I./ -o EmissionTests main.cpp ../Emission.o ../Chaine.o -lcppunit
[gelou@l4714-25 tests]$ '/usagers/gelou/Bureau/log1000-20/tp5SIEN/tp5/tests/Emis
sionTests'
EmissionTest::testgetters : OK
EmissionTest::trouverEmission : OK
EmissionTest::trouverEmission2 : OK
OK (3)
[gelou@l4714-25 tests]$
```

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]


```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[gelou@l4714-25 tp5]$ ls
'capture decran'  DB.txt          main.cpp          tests
Chaine.cpp        Emission.cpp    main.o            TP5_1878502_1847125.pdf
Chaine.h          Emission.h      Makefile
Chaine.o          Emission.o      'Rapport TP5.odt'
[gelou@l4714-25 tp5]$ git add Emission.cpp
[gelou@l4714-25 tp5]$ git add Emission.h
[gelou@l4714-25 tp5]$ git add Chaine.h
[gelou@l4714-25 tp5]$ git add Chaine.cpp
[gelou@l4714-25 tp5]$ git commit -m "Modification des fichiers"
diff: --cached: No such file or directory
diff: ': No such file or directory
.git/hooks/pre-commit: ligne 5 : read: « -r » : identifiant non valable
[master 0325f69] Modification des fichiers
  Committer: Georges Louis <gelou@l4714-25.info.polymtl.ca>
Votre nom et votre adresse e-mail ont été configurés automatiquement en se
fondant sur votre nom d'utilisateur et le nom de votre machine. Veuillez
vérifier qu'ils sont corrects. Vous pouvez supprimer ce message en les
paramétrant explicitement. Lancez les commandes suivantes et suivez les
instruction dans votre éditeur pour éditer votre fichier de configuration :

    git config --global --edit

Après ceci, vous pouvez corriger l'identité utilisée pour cette validation avec :

    git commit --amend --reset-author

4 files changed, 248 insertions(+)
create mode 100644 tp5SIEN/tp5/Chaine.cpp
create mode 100644 tp5SIEN/tp5/Chaine.h
create mode 100644 tp5SIEN/tp5/Emission.cpp
create mode 100644 tp5SIEN/tp5/Emission.h
[gelou@l4714-25 tp5]$ git push
Username for 'https://github.com': gelou
Password for 'https://gelou@github.com':
Décompte des objets: 8, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (7/7), fait.
Écriture des objets: 100% (8/8), 2.37 KiB | 0 bytes/s, fait.
Total 8 (delta 1), reused 0 (delta 0)
To https://github.com:gelou/polymtl-ca/git/log1000-20
   c0e3f25..0325f69  master -> master
[gelou@l4714-25 tp5]$
```

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

En examinant la méthode « TrouverEmission » de la classe « Emission.cpp » : Identifiez le nom des deux odeurs graves et expliquez pourquoi ce sont des odeurs graves. [/2]

Les deux odeurs graves de la méthode « TrouverEmission » sont la duplication du même code et la méthode est trop longue. On peut créer une méthode qui fait le fonctionnement du code qui est dupliqué.

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

Étape	Description
Déterminer le code qui est dupliqué	Trouver les répétitions du même code dans la méthode "TrouverEmission"
Créer une nouvelle méthode	La nouvelle méthode créer doit faire le travail du code qui est dupliqué

Restrutrez 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]

Nouvelle méthode créer :

```
String Emission ::FindInfo(string line) {  
    String information = "";  
    For (int i = 0; i < line.length(); i++){  
        If(line[i] != ',') {  
            Information += line[i];  
        }  
        Else {  
            Break;  
        }  
    }  
    return information;  
}
```

```
// 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)) {
        // Récupérer le nom de l'Emission

        int i = 0;

        string titreDB = FindInfo(line);

        // Si l'Emission qu'on lit actuellement est celui qu'on cherche
        if (titreDB == titre) {

            // Récupérer le nom de l'animateur
            string animateurDB = FindInfo(line);

            // Récupérer le nom de l'éditeur
            string chaineNameDB = FindInfo(line);

            // Récupérer le code postale de l'éditeur
            string chaineCodePostalDB = FindInfo(line);

            // Récupérer l'adresse de l'éditeur
            string chaineAddressDB = FindInfo(line);

            // 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

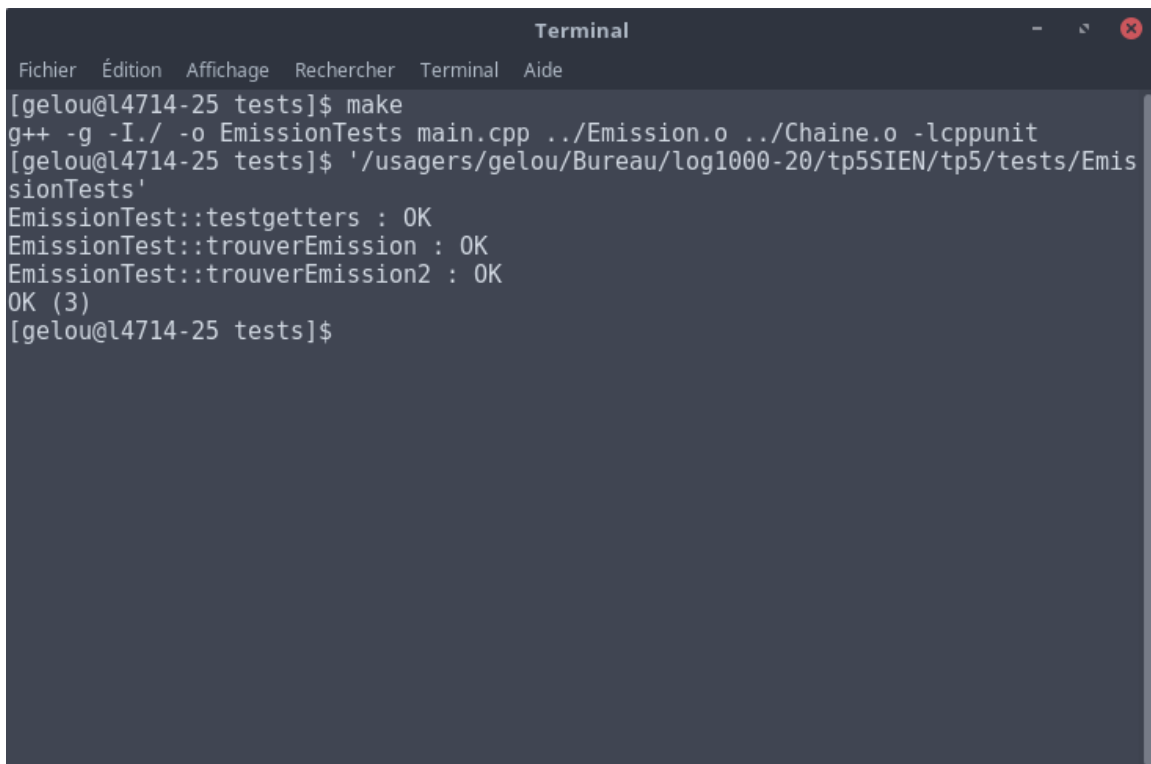
```

```
        fichier.close();
    }

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

    return NULL;
}
```

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

A terminal window titled "Terminal" with a menu bar containing "Fichier", "Édition", "Affichage", "Rechercher", "Terminal", and "Aide". The terminal shows the following commands and output:

```
[gelou@l4714-25 tests]$ make
g++ -g -I./ -o EmissionTests main.cpp ../Emission.o ../Chaine.o -lcppunit
[gelou@l4714-25 tests]$ './usagers/gelou/Bureau/log1000-20/tp5SIEN/tp5/tests/Emis
sionTests'
EmissionTest::testgetters : OK
EmissionTest::trouverEmission : OK
EmissionTest::trouverEmission2 : OK
OK (3)
[gelou@l4714-25 tests]$
```

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

```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[gelou@l4714-25 tp5]$ git add Emission.cpp
[gelou@l4714-25 tp5]$ git add Emission.h
[gelou@l4714-25 tp5]$ git commit -m "Modification des fichiers pour E2"
diff: --cached: No such file or directory
diff: ': No such file or directory
.git/hooks/pre-commit: ligne 5 : read: « -r » : identifiant non valable
[master 5384171] Modification des fichiers pour E2
  Committer: Georges Louis <gelou@l4714-25.info.polymtl.ca>
Votre nom et votre adresse e-mail ont été configurés automatiquement en se
fondant sur votre nom d'utilisateur et le nom de votre machine. Veuillez
vérifier qu'ils sont corrects. Vous pouvez supprimer ce message en les
paramétrant explicitement. Lancez les commandes suivantes et suivez les
instruction dans votre éditeur pour éditer votre fichier de configuration :

    git config --global --edit

Après ceci, vous pouvez corriger l'identité utilisée pour cette validation avec :

    git commit --amend --reset-author

2 files changed, 178 insertions(+)
 create mode 100644 tp5/Emission.cpp
 create mode 100644 tp5/Emission.h
[gelou@l4714-25 tp5]$ git push
Username for 'https://github.com': gelou
Password for 'https://gelou@github.com':
Décompte des objets: 3, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 355 bytes | 0 bytes/s, fait.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/polymtl-ca/git/log1000-20
   0325f69..5384171  master -> master
[gelou@l4714-25 tp5]$
```

E3) Utilisation des variables [/22]

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]

Variable	Span	Durée de vie	Portée
----------	------	--------------	--------

DBFile	Span= 2 $(34+14)/2 = 24$	51	Programme 55
emission	Span = 8 $(28+0+5+1+5+0+4+0) / 8 = 5,375$	52	Dynamique 56
Choix	Span = 3 $(9 + 0 + 40)/3 = 16,33$	52	Programme 54

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

DBFile, puisqu'il a un span qui est plus grand. Ainsi il faut donc minimiser cette variable pour une meilleur compréhension et lisibilité du code. Afin de pouvoir minimiser le span, on a changé les endroits où la variable est utilisée.

Proposez des restructurations pour améliorer l'utilisation de cette variable, en utilisant le même format du tableau de l'exercice E1. [/4]

Étape	Description
Déterminer la variable à modifier	Déterminer la variable qui a le plus grand Span.
Chercher une restructuration	Déterminer une restructuration qui va pouvoir diminuer le span de la variable.
Effectuer les changements	Une fois la restructuration complétée, effectuer les modifications.

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

```
#include <iostream>
```

```

#include "Emission.h"

using namespace std;

/*
 * fonction principale
 */
int main(int argc, char** argv) {

    Emission* emission = new Emission(); // Création d'un emission
    string DBFile = "DB.txt"; // Fichier qui contient une base des emissions
    int choix; // Opération sélectionnée par l'utilisateur

    do {
        // Afficher les opérations possibles
        std::cout << std::endl << "-----" << std::endl;
        std::cout << "0 - Quitter le programme " << std::endl;
        std::cout << "1 - Enregistrer l'émission " << std::endl;
        std::cout << "2 - Trouver une emission " << std::endl;
        std::cout << "3 - Afficher une emission " << std::endl;
        std::cout << "4 - Créer une emission " << std::endl;
        std::cout << "-----" << std::endl;

        // Lire le choix d'utilisateur
        std::cin >> choix;

        switch (choix) {

            case 1:

                {

                    // Enregistrer l'emission dans la base de données.
                    if (emission != NULL) {

```

```

        emission->saveEmission(DBFile);

        std::cout << "Emission enregistrée !" << std::endl;
    }
}

case 2:
{
    // Demander l'utilisateur de saisir le nom d'emission à chercher dans la base de données
    string titre;

    std::cout << "Saisir le titre de titre l'émission : ";

    std::cin >> titre;

    // Chercher l'emission
    Emission* tmp = emission->trouverEmission(DBFile, titre);

    if (tmp != NULL) { // Si l'emission est trouvé
        emission = tmp;

        std::cout << "Emission trouvée !" << std::endl;
    } else {
        std::cout << "Aucune émission trouvée !" << std::endl;
    }

    break;
}

case 3:
{
    // Afficher l'emission
    if (emission != NULL) {
        emission->afficher();
    }

    else {
        std::cout << "Aucune émission sélectionnée" << std::endl;
    }
}

```



```

        break;
    }

    case 4:
    {
        // Informations du nouvel emission
        string titre;
        string animateur;
        string chaineName;
        string chaineCodePostal;
        string chaineAddress;

        // Demander l'utilisateur de saisir les informations du nouvel emission
        std::cout << "Saisir le titre de l'émission : ";
        std::cin >> titre;
        std::cout << "Saisir l'animateur de l'émission : ";
        std::cin >> animateur;
        std::cout << "Saisir le nom de la chaîne : ";
        std::cin >> chaineName;
        std::cout << "Saisir le code postale de la chaîne : ";
        std::cin >> chaineCodePostal;
        std::cout << "Saisir l'adresse de la chaîne : ";
        std::cin >> chaineAddress;

        // Créer un nouvel emission
        delete emission;
        emission = new Emission(titre, animateur, chaineName, chaineCodePostal,
chaineAddress);

        break;
    }
}

} while (choix!= 0); // Tant que l'utilisateur ne décide pas de quitter le programme

```

```
return 0;
}
```

Compilez et testez manuellement (en exécutant le programme sur la ligne de commandes) les opérations (de l'opération 0 à 4) de la méthode « main », veuillez prendre des captures d'écrans de vos tests. [/4]

```
1
Saisir le titre de l'émission : XRA
Saisir l'animateur de l'émission : XRA
Saisir le nom de la chaîne : PFFR
Saisir le code postale de la chaîne : 123ABC
Saisir l'adresse de la chaîne : 123ABCDEFG
-----
0 - Quitter le programme
1 - Créer une émission
2 - Trouver une émission
3 - Afficher une émission
4 - Enregistrer l'émission
-----
2
Saisir le titre de titre l'émission : titreTest4
Emission trouvée !
-----
0 - Quitter le programme
1 - Créer une émission
2 - Trouver une émission
3 - Afficher une émission
4 - Enregistrer l'émission
-----
2
Saisir le titre de titre l'émission : abcdefg
Aucune émission trouvée !
-----
0 - Quitter le programme
1 - Créer une émission
2 - Trouver une émission
3 - Afficher une émission
4 - Enregistrer l'émission
-----
3
Titre : titreTest4
Animateur : animateurTest4
Chaîne name : chaîneNameTest4
Chaîne code postale : chaîneCodePostalTest4
Chaîne address : chaîneAddressTest4
-----
0 - Quitter le programme
1 - Créer une émission
2 - Trouver une émission
3 - Afficher une émission
4 - Enregistrer l'émission
-----
4
Emission enregistrée !
-----
0 - Quitter le programme
1 - Créer une émission
2 - Trouver une émission
3 - Afficher une émission
4 - Enregistrer l'émission
-----
```

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

```

[gelou@l4714-25 tp5]$ clear
[gelou@l4714-25 tp5]$ git pull
Username for 'https://github.com': gelou
Password for 'https://gelou@github.com':
Already up-to-date.
[gelou@l4714-25 tp5]$ git add .
[gelou@l4714-25 tp5]$ git commit -m "Modification E3"
diff: --cached: No such file or directory
diff: ': No such file or directory
.git/hooks/pre-commit: ligne 5 : read: « -r » : identifiant non valable
[master 7f83e14] Modification E3
  Committer: Georges Louis <gelou@l4714-25.info.polymtl.ca>
  Votre nom et votre adresse e-mail ont été configurés automatiquement en se
  fondant sur votre nom d'utilisateur et le nom de votre machine. Veuillez
  vérifier qu'ils sont corrects. Vous pouvez supprimer ce message en les
  paramétrant explicitement. Lancez les commandes suivantes et suivez les
  instructions dans votre éditeur pour éditer votre fichier de configuration :

      git config --global --edit

Après ceci, vous pouvez corriger l'identité utilisée pour cette validation avec :

      git commit --amend --reset-author

15 files changed, 452 insertions(+)
create mode 100644 tp5/.~lock.TP5LOG1000.docx#
create mode 100644 tp5/.~lock.TP5LOG1000.odt#
create mode 100644 tp5/Chaine.cpp
create mode 100644 tp5/Chaine.h
create mode 100644 tp5/DB.txt
create mode 100644 "tp5/LOG1000 \342\200\223 TP5.pdf"
create mode 100644 tp5/Makefile
create mode 100644 tp5/Makefile (1)
create mode 100644 tp5/TP5LOG1000.odt
create mode 100644 tp5/main (1).cpp
create mode 100644 tp5/main.cpp
create mode 100644 tp5/tests/EmissionTest.h
create mode 100755 tp5/tests/EmissionTests
create mode 100644 tp5/tests/Makefile
create mode 100644 tp5/tests/main.cpp
[gelou@l4714-25 tp5]$ git push
Username for 'https://github.com': gelou
Password for 'https://gelou@github.com':
Décompte des objets: 17, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (17/17), fait.
Écriture des objets: 100% (17/17), 815.34 KiB | 0 bytes/s, fait.
Total 17 (delta 3), reused 0 (delta 0)
To https://github.com:gelou/tp5
  5384171..7f83e14  master -> master

```

E4) Questions de cours [/10]

- i. Citez deux avantages de l'utilisation des APIs. Quel problème un développeur peut avoir en utilisant une API non stable ? [/1]

Les API permettent aux ordinateurs de générer le travail au lieu que ce soit un individu qui le fait. Ceci permet donc aux agences d'avoir des mises à jour continues sur les flux de travail ainsi assurant une performance de production.

Grâce aux APIs, l'intégration de contenu est mise à jour à partir plusieurs plateformes de manière plus facile: Site Web, application etc.. Ceci permet d'avoir une meilleure intégrité.

En utilisant un API instable, le développeur se doit de modifier le son code afin d'utiliser l'API. Ses mises à jour peuvent même être changé ou enlevé. L'API non stable occasionne plusieurs problèmes d'incohérence.

ii. **Citez deux librairies qui fournissent une API de bonne qualité. [/1]**

Python Standard Library (librairie utilisée en python) et STL Librairie de c++.

iii. **Comme la plupart de vous ont déjà touché à la programmation orienté objet, citez des erreurs que vous avez faites. Et comment allez vous les éviter après la lecture de "les APIs et les classes" ? [/1]**

Des erreurs qui sont souvent présente en programmation objet orientée sont liées à comment séparer ses classes. Quel code devrait être présent dans quelle classe? Quel est son utilité à être utilisé dans cette classe au lieu d'une autre? Afin d'éviter ce genre de problème, une solution serait de prendre d'avantage de temps pour évaluer la pertinence de ces classes durant la phase de conception.

iv. **Quelles sont les critères qu'on doit prendre en considération lors de l'implémentation d'une méthode ? [/1]**

Critères : La longueur de la méthode, les paramètres nécessaire, les noms utilisé, la cohésion.

v. **Pour faciliter la lecture de votre code, quels sont les pratiques que vous ne devez pas oublier ? [/1]**

S'assurer de laisser des commentaires clairs et précis, s'assurent d'indenter son code, regrouper les lignes de codes de manière cohérente et conserver de bas niveaux d'imbrications.

- vi. **Quelle est la différence entre une “Cohésion fonctionnelle” et une “Cohésion de communication” ? [/1]**

La cohésion de communication a lieu quand des fragments de modules sont groupés puisqu'ils fonctionnent sur les mêmes données tandis que la cohésion fonctionnelle est lorsque des fragments d'un module sont regroupés car ils contribuent à la même tâche.

- vii. **Définissez “patron de conception”, “anti-patron de conception” et “mauvaise odeur du code” et le lien entre eux ? [/2]**

Les patrons de conception présentent les liens entre les différentes composantes du logiciel. Les anti-patron de conception décrivent les erreurs qui ont lieu de manière fréquente durant la conception du logiciel puisqu'on n'a pas utilisé les patrons de conception correctement. Les mauvaises odeurs du code sont causées par une mauvaise structure du code. Ceux-ci reflètent donc des erreurs d'anti patron de conception.

- viii. **(BONUS) Nous sommes rendus à la fin des TPs du LOG1000. Comment avez vous trouvé les TPs ? et d'après vous quels sont les changements qu'on doit apporter aux TPs ? [/2]**

Les consignes des TPs ne sont pas toujours très claires. Nous considérons que celles-ci peuvent être un peu plus précises. Néanmoins, nous avons beaucoup appris quant aux systèmes de gestion de versions. Les TPs couvrent très bien la matière vue en classe.