



# Travail pratique # 5 : Restructuration du code source

École polytechnique de Montréal

**Trimestre :** automne 2017

Équipier1 : *Julien, Legault matricule 1847125*

Équipier2 : *Duc-Thien, Nguyen matricule 1878502*

**Equipe numéro : 25**

**Présenté à :** Khalil Bennani

Polytechnique Montréal  
Date de remise (27-11-2017)

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

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

La classe présente est beaucoup trop grosse puisque nous avons des méthodes et des attributs qui n'ont aucun lien ensemble. Le problème ici est donc que nous avons une classe en regroupant deux.

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

Pour régler cette odeur du code, il va falloir utiliser le principe d'extraction d'une classe afin d'avoir deux classes distinctes et par la suite transférer les méthodes et paramètres nécessaires.

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

Parmi les méthodes que nous avons actuellement, les méthodes qui iront dans la classe Chaîne seront :

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

Pour la classe Émission, les méthodes qui seront modifier afin d'utiliser le principe de composition sont :

- void associerChaîne(Chaîne\*);
- Chaîne\* getChaîne();

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

Les attributs qui seront déplacé dans la nouvelle classe sont :

- string chaîneName;
- string chaîneCodePostal;
- string chaîneAddress;

Pour la classe Émission, les attributs qui seront modifier afin d'utiliser le principe de composition sont :

- Chaîne\* chaîne;

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

Tableau 1 : Étapes nécessaire à la correction de la mauvaise odeur dans la classe Chaîne

Étape	Description
Création de la classe Chaîne	Créer un fichier .cpp et un fichier .h
Écriture de Chaîne.h	Transférer les définitions des méthodes ainsi que les attributs de la classe Chaîne.
Écriture de Chaîne.cpp	Transférer les implémentations des méthodes ainsi que les attributs de la classe Chaîne.
Modification de Emission.h	Enlever les définitions des classes faisant maintenant parti de la classe Chaîne.
Modification de Emission.cpp	Enlever les implémentations des classes faisant maintenant parti de la classe Chaîne.

**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

```
1  #ifndef Emission_H
2  #define Emission_H
3
4  #include <string>
5  #include "Chaine.h"
6
7  using namespace std;
8
9  // Cette classe représente un Emission
10 class Emission {
11 public:
12     // Constructeurs
13     Emission ();
14     Emission(string, string, string, string, string);
15
16     // Setters
17     void setTitre(string);
18     void setAnimateur(string);
19     void associerChaine(Chaine*);
20
21     // Getters
22     string getTitre();
23     string getAnimateur();
24
25     Chaine* getChaine();
26
27     // Enregistrer l'Emission
28     void saveEmission(string);
29     // Afficher l'Emission
30     void afficher();
31     // Chercher un Emission dans une base de données par titre
32     Emission* trouverEmission(string, string);
33
34 private:
35     // Information sur l'emission
36     string titre;
37     string animateur;
38     Chaine* chaine;
39
40
41 };
42
43 #endif
44
```

Figure 1 : définition de la classe Chaine

Emission.cpp

```

1  #include "Emission.h"
2
3  #include <fstream>
4  #include <iostream>
5
6  // Constructeur
7  Emission::Emission(){
8      titre = "";
9      animateur = "";
10     chaine = new Chaine();
11 }
12
13
14 Emission::Emission (string titre,
15                     string animateur,
16                     string chaineName,
17                     string chaineCodePostal,
18                     string chaineAddress) {
19     // Emission information
20     this->titre = titre;
21     this->animateur = animateur;
22     this->chaine = new Chaine (chaineName, chaineCodePostal, chaineAddress);
23 }
24
25 // Setters
26 void Emission::setTitre(string titre) {
27     this->titre = titre;
28 }
29 void Emission::setAnimateur(string animateur) {
30     this->animateur = animateur;
31 }
32
33 // Associer une Chaine à l'Emission
34 void Emission::associerChaine (Chaine* chaine) {
35     this -> chaine = chaine;
36 }
37
38 // Getters
39 string Emission::getTitre() {
40     return this->titre;
41 }
42
43 string Emission::getAnimateur() {
44     return this->animateur;
45 }
46
47 Chaine* Emission::getChaine(){
48     return chaine;
49 }
50
51 // Enregistrer l'Emission dans un fichier
52 void Emission::saveEmission (string fileName) {
53     ofstream outfile (fileName.c_str(), std::ofstream::binary | std::fstream::app);
54     // write to outfile
55     outfile<<this->titre <<","
56     <<this->animateur <<"\n";
57     outfile.close();
58 }
59
60 // Trouver un Emission avec son nom dans la base de données DB
61 Emission* Emission::trouverEmission (string DB, string titre) {
62     ifstream fichier(DB.c_str(), ios::in); // Ouvrir le fichier "DB.txt"
63     Emission*tmp=NULL;
64
65
66
67

```

Figure 2 : Implémentation de la classe Émission (partie 1)

```

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'adresse 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();
        }
    }
}

```

Figure 3 : Implémentation de la classe Émission (partie 2)

```

        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'adresse 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;
}

```

Figure 4 : Implémentation de la classe Émission (partie 3)

```

1  #ifndef Chaîne_H
2  #define Chaîne_H
3
4  #include <string>
5
6  using namespace std;
7
8  // Cette classe représente une chaîne
9  class Chaîne {
10 public:
11     // Constructeurs
12     Chaîne ();
13     Chaîne(string, string, string);
14
15     // Getters
16     string getChaîneName();
17     string getChaîneCodePostal();
18     string getChaîneAddress();
19
20 private:
21
22     // Informations sur la chaîne
23     string chaîneName;
24     string chaîneCodePostal;
25     string chaîneAddress;
26
27 };
28
29 #endif
30
31

```

Figure 5 Définition de la classe Chaîne

```

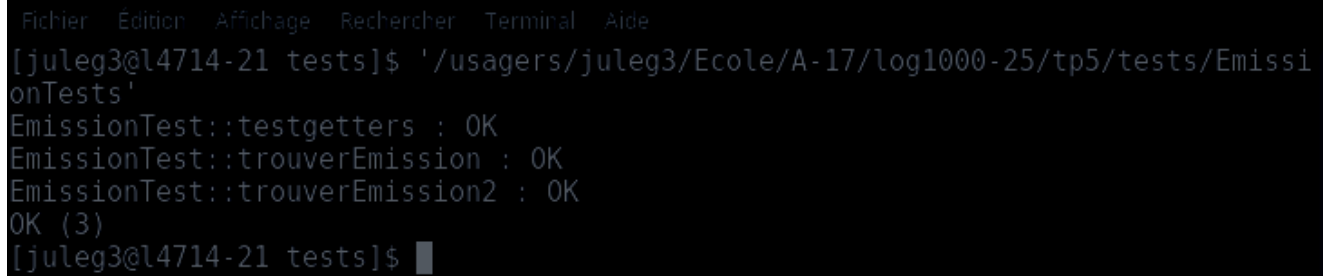
1  #include "Chaine.h"
2
3  #include <fstream>
4  #include <iostream>
5
6  // Constructeur
7
8  Chaine::Chaine(){
9      this->chaineName = "";
10     this->chaineCodePostal = "";
11     this->chaineAddress = "";
12 }
13
14 Chaine::Chaine (string chaineName,
15                 string chaineCodePostal,
16                 string chaineAddress) {
17     this->chaineName = chaineName;
18     this->chaineCodePostal = chaineCodePostal;
19     this->chaineAddress = chaineAddress;
20 }
21
22 // Getters
23
24 string Chaine::getChaineName() {
25     return this->chaineName;
26 }
27
28 string Chaine::getChaineCodePostal() {
29     return this->chaineCodePostal;
30 }
31
32 string Chaine::getChaineAddress() {
33     return this->chaineAddress;
34 }
35
36
37
38
39

```

Figure 6 Implémentation de la classe Chaine



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



```
Fichier  Edition  Affichage  Recherche  Terminal  Aide
[juleg3@l4714-21 tests]$ '/usagers/juleg3/Ecole/A-17/log1000-25/tp5/tests/EmissionTests'
EmissionTest::testgetters : OK
EmissionTest::trouverEmission : OK
EmissionTest::trouverEmission2 : OK
OK (3)
[juleg3@l4714-21 tests]$
```

Figure 7: Exécution des tests unitaires avec la nouvelle classe implémentée

**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]**

```
[juleg3@l4714-21 tp5]$ git add Emission.cpp
[juleg3@l4714-21 tp5]$ git add Emission.h
[juleg3@l4714-21 tp5]$ git add Chaine.h
[juleg3@l4714-21 tp5]$ git add Chaine.cpp
[juleg3@l4714-21 tp5]$ git commit -m "JL:modification des fichiers sources et leur definition"
[master 64a7305] JL:modification des fichiers sources et leur definition
Committer: Julien Legault <juleg3@l4714-21.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
```

```
[juleg3@l4714-21 tp5]$ git push
Username for 'https://github.com:': juleg3
Password for 'https://juleg3@github.com:':
Décompte des objets: 14, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (14/14), fait.
Écriture des objets: 100% (14/14), 1.36 KiB | 0 bytes/s, fait.
Total 14 (delta 12), reused 0 (delta 0)
To https://github.com:polymtl-ca/git/log1000-25/
40ec110..64a7305 master -> master
```

Figure 8 : Commit des fichiers des deux classes implémentées

## 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]**

La méthode de « TrouverEmission » a plusieurs parties du code qui sont dupliquées et la méthode est trop longue. Il est possible d'enlever ces duplications en créant une méthode.

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

Tableau 2 : Étapes nécessaire à la restructuration de la méthode TrouverEmission

Étape	Description
Déterminer les duplications	Trouver les répétitions de codes dans la méthode
Création d'une nouvelle méthode	Créer une nouvelle méthode pour la recherche des noms

**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]**

Ajout de la méthode «trouverInfo » pour enlever les duplications dans le code.

```
string Emission::trouverInfo(string line, int& index){  
  
    if(index != 0)  
        index++;  
  
    for (index ; index < line.length() ; index++) {  
        if (line[index] != ',') {  
            string info += line[index];  
        } else {  
            break;  
        }  
    }  
    return info;  
}
```

```

// 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 = trouverInfo(line,i);

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

                // Récupérer le nom de l'animateur
                string animateurDB = trouverInfo(line,i);

                // Récupérer le nom de l'éditeur
                string chaineNameDB = trouverInfo(line,i);

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

                // Récupérer l'adresse de l'éditeur
                string chaineAddressDB = trouverInfo(line,i);

                // 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;
}

```

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

```
(dunguf@l4714-11 tests)$ '/usagers/dunguf/Documents/log1000-25/tp5/tests/EmissionTests'
EmissionTest::testgetters : OK
EmissionTest::trouverEmission : OK
EmissionTest::trouverEmission2 : OK
OK (3)
```

Figure 8 : Exécution des tests unitaires avec la nouvelle classe implémentée

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

```
Thien@DESKTOP-76GGOHG MINGW64 ~/Documents/log1000/log1000-25/tp5 (master)
$ git add Emission.cpp
```

```
Thien@DESKTOP-76GGOHG MINGW64 ~/Documents/log1000/log1000-25/tp5 (master)
$ git add Emission.cpp Emission.h
```

```
Thien@DESKTOP-76GGOHG MINGW64 ~/Documents/log1000/log1000-25/tp5 (master)
$ git commit -m "Modification du Emission .cpp et .h pour la question E2"
[master d5234bd] Modification du Emission .cpp et .h pour la question E2
2 files changed, 1 insertion(+), 1 deletion(-)
```

```
Thien@DESKTOP-76GGOHG MINGW64 ~/Documents/log1000/log1000-25/tp5 (master)
$ git push
Counting objects: 5, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 469 bytes | 469.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
To https://github.com/polymtl.ca/git/log1000-25
8406fa8..d5234bd master -> master
```

Figure 10 : Commit des modifications appliquées à la classe Emission.

### E3) Utilisation des variables [/22]

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]

Tableau 3 : Tableau identifiant les réponses aux différents calculs fait avec les variables présentent dans la méthode main

Variable	Span	Durée de vie	Portée
DBFile	Total span : 2 $(34+14)/2 = 24$	$61 - 11 = 51$	Permanente 55
Choix	Total span : 3 $(9 + 0 + 40)/3 = 16,33$	$64 - 12 = 52$	Programme 54
emission	Total span : 8 $(28+0+5+1+5+0+4+0) / 8 = 5,375$	$62 - 12 = 52$	Dynamique 56

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

DBFile, car il a une très grande span, donc il faut la minimiser pour une meilleur compréhension et la lisibilité du code. Pour minimiser le span, on décide de changer les endroits où la variable est utilisée. Dans ce TP, il faut changer de place les conditions du « Switch case ».

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

Tableau 4 : Tableau identifiant les étapes à la restructurations de la variable DBFile

Étape	Description
Déterminer la variable à modifier	Avec le tableau fait ci-dessus, déterminer la variable qui a le plus grand Span.
Chercher une restructuration	Déterminer une restructuration qui diminuera le span de la variable.
Effectuer changement	Après avoir déterminer la restructuration, 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 choi 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 informatins 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 chaine : ";
    std::cin >> chaineName;
    std::cout << "Saisir le code postale de la chaine : ";
    std::cin >> chaineCodePostal;
    std::cout << "Saisir l'adresse de la chaine : ";
    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;  
}
```

Figure 11 : modifications de la méthode main



**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
-----
0
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$
```

Figure 12 : Test manuels de l'exécutable généré.

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

```
julien@julien-Inspiron-13-5368: ~/Bureau/Ecole/A-17/TP/log1000-25/tp5
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$ git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

    modifié :      DB.txt
    modifié :      main.cpp

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

    capture decran/testUnitaireV2.png

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$ git pull
Username for 'https://github.com: juleg3
Password for 'https://juleg3@github.com: juleg3
Already up-to-date.
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$ git add .
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$ git commit -m "JL: modification du code source"
[master 5f04ffa] JL: modification du code source
 3 files changed, 36 insertions(+), 31 deletions(-)
 create mode 100644 tp5/capture decran/testUnitaireV2.png
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$ git push
warning: push.default n'est pas défini ; sa valeur implicite a changé dans Git 2.0
de 'matching' vers 'simple'. Pour supprimer ce message et maintenir
le comportement actuel après la modification de la valeur de défaut, utilisez :

    git config --global push.default matching

Pour supprimer ce message et adopter le nouveau comportement maintenant, utilisez :

    git config --global push.default simple

Quand push.default vaudra 'matching', git poussera les branches locales
sur les branches distantes qui existent déjà avec le même nom.

Depuis Git 2.0, Git utilise par défaut le comportement plus conservatif 'simple'
qui ne pousse la branche courante que vers la branche distante correspondante
que 'git pull' utilise pour mettre à jour la branche courante.

Voir 'git help config' et chercher 'push.default' pour plus d'information.
(le mode 'simple' a été introduit dans Git 1.7.11. Utilisez le mode similaire
'current' au lieu de 'simple' si vous utilisez de temps en temps d'anciennes versions de Git)

Username for 'https://github.com: juleg3
Password for 'https://juleg3@github.com: juleg3
Décompte des objets: 7, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (7/7), fait.
Écriture des objets: 100% (7/7), 93.92 KiB | 0 bytes/s, fait.
Total 7 (delta 5), reused 0 (delta 0)
To https://github.com:juleg3/log1000-25
 337acf7..5f04ffa master -> master
julien@julien-Inspiron-13-5368:~/Bureau/Ecole/A-17/TP/log1000-25/tp5$
```

Figure 13 : Commit du code source nécessaire à l'exécution du programme

## 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]**

- 1) Avec les API, les ordinateurs peuvent générer le travail au lieu d'une personne. Grâce aux APIs, les agences peuvent mettre à jour les flux de travail pour une meilleure performance. Ça le rend plus rapide et plus productif.
- 2) Les APIs permettent d'intégrer le contenu à partir de n'importe quel site ou application plus facilement. Ça donne alors une meilleure intégrité.

3)

Si un développeur utilise un API non-stable, il va devoir peut-être remodifier son code pour pouvoir utiliser le API. Si le API n'est pas stable, ça pourrait être enlevé ou changé. Si ce développeur a travaillé sur un projet depuis longtemps et qu'un de ces problèmes se présente, il aura plusieurs problèmes d'incohérence.

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

STL : bibliothèque pour le c++

Python Standard Library : bibliothèque utilisée en python

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

Une des erreurs majeures en orientées objets est l'absence = new Chaine(); gestion des classes. Des fois, il y a l'utilisation des classes qui sont nécessaires ou sinon tout est regroupé dans une classe. Il faut rendre le code le plus lisible possible et le plus performant possible. Pour éviter ce type de problème, nous pourrions prendre plus de temps à évaluer la pertinence de certaines classes ou sinon créer des nouvelles classes au besoin.

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

Les critères nécessaires lors de l'implémentation sont : les noms utilisés pour la précision et la compréhension, la cohésion pour une implémentation idéal et performante, la longueur pour qu'il ne soit pas trop long ainsi que les paramètres nécessaires.

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

Premièrement, Il faut tout d'abord laisser plusieurs commentaires, par exemple mettre des entêtes sur des méthodes d'une classe. Deuxièmement, il faut bien indenter les codes. Troisièmement, il faut bien regrouper des lignes de codes ensemble qui sont inters reliés et laisser des commentaires de son fonctionnement. Quatrièmement, il faut écrire des codes qui ne se répètent pas trop, car ça augmente la difficulté de lecture et cause des mauvaises odeurs de codages. Cinquièmement, garder des bas niveaux d'imbrications.

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

La cohésion de communication est lorsque des parties d’un module sont groupées parce qu’elles fonctionnent sur les mêmes données. Par ailleurs, La cohésion fonctionnelle est lorsque les parties d’un module sont groupées, car elles contribuent toutes à une seule tâche bien définie.

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

D’un côté, les patrons de conception décrivent les liens entre les différentes composantes du logiciel. D’un autre côté, les anti-patrons décrivent des erreurs qui apparaissent fréquemment lors de la conception d’un logiciel dû à l’absence de l’usage d’un patron de conception ou à sa mauvaise application. Les mauvaises odeur du code sont dues à une mauvaise structure du code et reflètent alors des erreurs lors d’anti-patron de conception.

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

En faisant les TPs, mon coéquipier et moi avons remarqué que certaines questions sont un peu vagues. Il serait bien que ces questions aient plus d’informations et d’exigences afin de guider l’étudiant à comprendre ce que l’énoncé demande. Sinon, les questions des TPs couvrent bien l’ensemble de la matière de chaque TP et permettent de bien retenir la matière.