

LOG1000 – Ingénierie Logicielle TP5

Restructuration du code source
Hiver 2017

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 des restructurations adéquates, et ce, en plusieurs phases consécutives.
- **Compiler et tester** le code restructuré après chaque phase pour valider la viabilité 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 écrans que vous allez mettre dans votre rapport.
- Les ressources qui seront consultées en ligne seront en anglais. Si vous voulez de l'aide de traduction pour certains termes techniques, n'hésitez pas à en faire part au chargé de laboratoire.

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) [/2]
- 2) [/2]
- 3) [/3]
- 4) [/6]
- 5) [/5]
- 6) [/6]
- 7) [/4]
- 8) [/2]

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

- 1) [/2]
- 2) [/6]
- 3) [/6]
- 4) [/4]
- 5) [/2]

E3) Utilisation des variables [/22]

- 1) [/6]
- 2) [/2]
- 3) [/4]
- 4) [/4]
- 5) [/4]
- 6) [/2]

Bonus : [/4]

Énoncé :

Partie 1:

Dans cet énoncé, vous devez identifier et restructurer les mauvaises odeurs présentes dans un programme C++. 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 évoluer, 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 [/30]

En examinant les attributs de la classe « Emission », vous pouvez remarquer qu'elle joue deux rôles différents, et ainsi 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. [/6]
- 4) Identifiez les attributs qu'il faut modifier ou déplacer. [/3]
- 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 [/5]

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. [/4]
- 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 [/20]

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 [/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]
- 2) Interprétez les résultats, et trouvez la variable (parmi les trois citées en dessus) qui bénéficiera le plus de la restructuration. [/2]
- 3) Proposez des restructurations pour améliorer l'utilisation de cette variable, en utilisant le même format du tableau de l'exercice E1. [/4]
- 4) Effectuez cette restructuration dans la méthode « main ». Faites une capture d'écran (ou copiez le code) de votre nouveau code source. [/4]
- 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. [/4]
- 6) Faites un commit de votre code source, et une capture d'écran de ce commit dans le rapport. [/2]

Bonus : [/4]

Nous sommes rendus à la fin des TP5 du LOG1000. Vous avez eu dans les 4 premiers TP5 une partie qui concerne le projet open source Ring. On vous demande de citer **trois avantages et trois inconvénients** de travailler sur ce genre de projet en se basant sur vos propres expériences durant cette session. Comme vos commentaires seront utilisés pour évaluer l'inclusion des projets open source dans les TP5, svp donnez des commentaires concrets et constructifs.

3 Avantages	3 inconvénients

DATE LIMITE DE REMISE

19 AVRIL À MIDI

Pénalité pour retard 10% par jour