TP 5 : Étude et amélioration d'une application

1 Objectifs

L'objectif de ce TP est d'étudier et d'optimiser une application complète écrite en Java.

2 Matériels et logiciels nécessaires

Ces travaux pratiques se déroulent sur système GNU/Linux. Si vous travaillez sur votre machine personnelle, il est de votre responsabilité d'y installer et configurer tous les logiciels nécessaires. Les machines de l'université fonctionnent sous la distribution ubuntu. Vous pouvez travailler sur d'autres distribution si vous le souhaitez. La version du JDK que nous utilisons est la version 8.

3 Évaluation

L'évaluation de chaque TP s'effectue via compte-rendu écrit, seul ou en binôme, des travaux effectués durant la séance de TP. Ce compte-rendu doit être fourni au format PDF pour chacun des TP. Vous pouvez utiliser tous les outils que vous souhaitez pour mettre en forme votre compte-rendu : Word, Latex, LibreOffice... Il est conseillé de commencer à préparer le compte-rendu au fur et à mesure que vous avancez sur le TP. Quelques points rentrant en compte pour la notation :

- Maitrise des outils abordés en cours
- Analyses personnelles des travaux réalisés
- Présence de screenshots, de résultats des outils etc
- Qualité de la mise en page (Latex non obligatoire, mais une mise en page correcte n'est)
- Pas de plagiat
- Présence en cours/TP
- Respect des délais et des consignes pour rendre le compte-rendu

L'objectif du compte rendu est de montrer que vous avez compris ce que vous faisiez, et que vous êtes capables d'avoir un regard critique. Prendre du recul sur les différents outils en allant plus loin que la simple description de ceux-ci est très important.

Concernant les délais, le compte-rendu doit être rendu au plus tard une semaine après la dernière séance du TP sur la plateforme Celene dédiée à ce cours. Le mot de passe vous sera fourni lors des séances.

Exemple de date limite de rendu du compte-rendu : Ce sujet de TP s'étale sur 4 séances. Si la dernière séance est un mardi, vous avez jusqu'au mardi de la semaine suivante 23h59 pour envoyer votre rapport.

Bien entendu, je serai présent lors des TP, n'hésitez donc pas à me poser des questions si vous en avez besoin. Vous pouvez aussi me contacter par mail entre deux séances en cas de problème.

Il n'existe pas une seule et unique correction possible pour ce TP. De ce fait, aucune correction ne vous sera fournie. Cependant, si vous souhaitez avoir des informations/précisions sur votre compte rendu, n'hésitez pas à me contacter soit par mail soit directement pendant les séances suivantes pour que nous puissions en discuter.

TP 5 : Étude et amélioration d'une application

1.1 Présentation

Durant quatre séances, on vous propose d'étudier et améliorer un programme Java presque ordinaire. Ce programme consiste en la simulation d'une colonie de fourmis peintres. Les fourmis se déplacent sur une surface sans bord. Le déplacement d'une fourmi obéit à des règles simples : soit elle détecte à proximité une couleur qui l'intéresse et décide de la suivre ou pas, soit elle se déplace aléatoirement. A chaque déplacement, la fourmi dépose sa couleur sur la surface. Sur une exécution longue, une auto-organisation apparaît. L'application est actuellement réalisée sous la forme d'une applet. Il est possible de changer les paramètres (taille...) et de définir une image pour les couleurs initiales de la surface.

Vos objectifs sont :

- Établir une analyse des performances de l'application existante et des faiblesses du code actuel
- Après avoir établi un diagnostic, vous devez modifier/améliorer l'application afin de permettre de visualiser beaucoup plus rapidement l'auto-organisation à long terme.
- Établir une nouvelle analyse montrant les éventuelles améliorations/dégradations apportées.

L'ensemble de votre travail sera à rendre sous la forme d'une archive ZIP contenant : le rapport de pré/post analyse et les sources de votre programme. L'argumentation représente une part importante de votre évaluation de ce travail. L'amélioration ou non des performances ne représente qu'une partie des critères de notation.

Attention, ce TP représentera une part importante de la note finale dans cette matière.

1.2 Téléchargement du projet

Le code source du projet est disponible dans un zip sur la plateforme celene du TP.

1.3 Lancement du programme

Le code est sous la forme d'un projet maven standard prêt à l'emploi. Afin de faciliter les tests, j'ai ajouté au code java un script shell qui permet la compilation et le lancement du programme dans le sous dossier **src/main/sh** :

```
1 [florentclarret@imac-de-florent-clarret:~/Documents/Polytech/TP/paintingants/src/main/sh] : ./run.sh
```

Cette commande aura pour effet de :

1. Clean le projet maven
2. Recompiler le jar à partir des sources
3. Exécuter le programme à partir d'un fichier html

Les fichiers html permettant de lancer le programme se trouvent dans le répertoire **src/main/html**. Par défaut, le script utilise le fichier *ants_default.html*, qui génère les paramètres aléatoirement. Il existe un second fichier qui se nomme *ants_worst.html* et qui contient les paramètres pour tester le programme dans le pire des cas. Pour lancer un test sur un fichier html spécifique, vous pouvez ajouter le nom de ce fichier au script *run.sh* :

```
[florentclarret@imac-de-florent-clarret:~/Documents/Polytech/TP/paintingants/src/main/sh] : ./run.sh ants_worst.html
```

Vous pouvez rajouter autant de cas de tests que vous le souhaitez, avec plus ou moins de fourmis, des tailles de grilles différentes, etc... Vous avez aussi le droit de ne pas utiliser ce script ou de le modifier si besoin.

1.4 Les tâches à réaliser

1.4.1 Analyse préalable de l'application

Dans un premier temps, il vous est demandé d'effectuer une analyse complète de l'application. Pour cela, vous devez utiliser tous les outils pertinents et que nous avons vu au cours des TP précédents. L'objectif est de mettre en évidence les potentielles faiblesses de l'application et d'expliquer les différentes modifications que vous pensez y apporter afin de les résoudre. En parcourant le code, vous trouverez probablement des axes d'amélioration qui ne sont pas ou peu visible à l'aide des outils d'analyse. Vous pouvez aussi les inclure dans votre analyse.

1.4.2 Amélioration de l'application

Vous avez le droit d'effectuer toutes les modifications que vous jugez nécessaires. N'hésitez pas à expérimenter pour voir si une implémentation est meilleure qu'une autre. Attention toutefois, il ne faut pas modifier le comportement général de l'application, on souhaite uniquement en améliorer les performances, pas son fonctionnement nominal. Cependant, il est possible de faire certaines concessions sur le fonctionnement de base afin d'améliorer les performances. Nous discuterons de ces concessions au cours des séances de TP et elles devront toujours être validées par moi-même.

A chaque fois que vous effectuez des modifications, il vous est demandé d'effectuer une rapide analyse à la suite de celles-ci afin de mettre en évidence les améliorations que vous y avez apporté, si c'est bien le cas. Dans le cas contraire, si les performances n'ont pas évolué ou si elles sont moins bonnes, il est nécessaire d'y apporter toutes les explications nécessaires. Il faut aussi que vous expliquiez très clairement les raisons qui vous pousse à essayer de mettre en place cette amélioration. Il est très important d'être très précis dans les explications que vous fournissez. Une amélioration sans explication précise et détaillée ne sera pas prise en compte lors de l'évaluation, aussi bonne soit-elle.

1.4.3 Analyse finale de l'application

Une fois que vous jugez avoir apporté toutes les modifications possibles sur l'application (ou que vous n'avez plus de temps), il vous est demandé d'effectué un diagnostic final et de le confronter avec celui que vous avez effectué au tout début du TP pour faire le bilan d'une manière globale de votre travail. L'objectif est de prendre du recul et avoir un esprit critique.

Si vous le souhaitez, vous pouvez mentionner les différentes améliorations que vous pensiez mettre en place et pourquoi.

1.5 Quelques conseils avant de démarrer

Dans cette partie, je rajoute quelques conseils ainsi que des informations complémentaires sur le sujet. Vous être libre de les suivre ou non.

1.5.1 Refactoring de l'application

Vous allez le constater, l'application est composée de 4 classes. Avant d'essayer d'améliorer les performances de l'application, il est conseillé (mais pas obligatoire) de réorganiser le code afin d'y ajouter des classes et ainsi décharger les autres classes de certaines tâches. Cela vous permettra d'avoir des classes plus petites avec moins de fonctionnalités et devrait grandement vous faciliter la tâche pour améliorer les performances. Par exemple, vous pouvez avoir une classe qui ne s'occupe que de la gestion des paramètres, une autre qui ne s'occupe que de la gestion de la souris, etc.

Définir un rôle précis pour chacune des classes vous permettra de vous y retrouver facilement dans l'application. De plus, cela me facilitera aussi la tâche afin de comprendre l'architecture de votre code.

1.5.2 To git or not to git ?

Dans le cadre de ce TP, vous avez la possibilité d'utiliser ou non un gestionnaire de version (git de préférence pour ne pas le nommer). Ce n'est pas obligatoire mais fortement recommandé pour plusieurs raisons.

La première est que cela vous permettra de revenir en arrière si les modifications que vous avez faites ne sont pas satisfaisantes. Si c'est le cas, ne supprimez pas vos commits et créez une nouvelle branche. Toute modification, même sans résultat, peut-être intéressante si elle est expliquée correctement. La seconde est qu'il sera plus facile pour vous de relancer des tests sur des anciennes versions de votre code en utilisant git. Ensuite, lorsque j'évaluerai votre projet, cela me permettra de suivre le cheminement que vous avez eu au cours des séances en regardant la suite de vos commits. Enfin, un gestionnaire de version est un outil INDISPENSABLE lorsque l'on développe des applications. Il est très utile de versionner des projets, même si vous êtes le/la seul/seule à travailler dessus. Au plus tôt vous maitriserez ces outils, au mieux ce sera.

Dans tous les cas, je n'accepterai pas d'avoir plusieurs projets dans le zip final que vous m'enverrez car vous avez voulu revenir en arrière sur des modifications sans pour autant "perdre" ces modifications. Un seul et unique projet doit être rendu, versionné ou non.

1.5.3 De l'intérêt des tests unitaires

Le projet ne compte actuellement aucun test unitaire. Avant de vous lancez dans la modification de certaines fonctionnalités, il peut être utile de mettre en place des tests unitaires sur celles-ci afin de vérifier que :

1. Le code fonctionnait bien avant la modifications
2. Le code fonctionne toujours après la modification

De manière générale, je vous encourage à ajouter le plus de tests unitaires possibles pour vous assurer du bon comportement de votre application. Un code performant c'est bien, un code qui fonctionne vraiment c'est mieux.

Il n'est pas impossible que le projet qui vous est donné contienne des bugs ou des approximations. Un code sans aucun bug n'existe pas (ou presque). Mettre en place ces tests vous permettra de les détecter.

1.5.4 Vous avez dit Javadoc ?

La documentation d'une application est un aspect primordial. Même si cela n'améliore en rien les performances de votre application, cela vous permettra de comprendre plus facilement ce que chacune de vos méthodes/classes font, ainsi que le sens de chacun des paramètres, même si vous êtes vous-même l'auteur du code. Il n'est pas rare de revenir quelques semaines plus tard sur son propre code et de ne pas comprendre

ce qu'il fait. De plus, cela me permettra aussi de comprendre rapidement le sens d'une méthode si elle est correctement documentée.

1.5.5 De l'utilisation des logs

Il peut être utile de mettre en place un système de logs au sein de votre application afin de savoir ce qu'elle est en train de faire, ou loguer certaines informations importantes. Si vous le souhaitez, vous pouvez en ajouter. Pour cela, vous avez plusieurs solutions :

1. Utiliser simplement des System.out et System.err
2. Mettre en place un framework de loging tel que Log4j ou Logback

Attention toutefois, écrire des logs pertinents est une bonne chose, trop en écrire est contre-productif. De plus, mettre en place trop de logs ralentira de manière certaine votre application.

1.5.6 Concernant la mise en place de micro-benchmark

Si au cours de ce TP vous avez mis en place des tests en utilisant JMH par exemple, vous êtes autorisé à inclure les sources de ce projet dans le zip final. Cependant, vous n'avez pas le droit d'ajouter plusieurs projets JMH différents dans le zip. Si vous avez plusieurs tests portant sur différentes choses, créez plusieurs classes dans le même projet. Pour être pris en compte dans l'évaluation, ces micro-benchmarks ainsi que les résultats issus de ceux-ci doivent être présents dans votre rapport, accompagnés de toutes les explications nécessaires.