# 1. Presentation

Durant quatre seances, on vous propose d’etudier et ameliorer un programme Java presque ordinaire. Ce programme consiste en la simulation d’une colonie de fourmis peintres. Les fourmis se deplacent sur une surface sans bord. Le deplacement d’une fourmi obeit a des regles simples : soit elle detecte a proximite une couleur qui l’interesse et decide de la suivre ou pas, soit elle se deplace aleatoirement. A chaque deplace- ment, la fourmi depose sa couleur sur la surface. Sur une execution longue, une auto-organisation apparaıt. L’application est actuellement realisee sous la forme d’une applet. Il est possible de changer les parametres (taille...) et de definir une image pour les couleurs initiales de la surface.

Vos objectifs sont :

* Etablir une analyse des performances de l’application existante et des faiblesses du code actuel
* Apres avoir etabli un diagnostic, vous devez modifier/ameliorer l’application afin de permettre de visualiser beaucoup plus rapidement l’auto-organisation a long terme.
* Etablir une nouvelle analyse montrant les eventuelles ameliorations/degradations apportees.

L’ensemble de votre travail sera a rendre sous la forme d’une archive ZIP contenant : le rapport de pre/post analyse et les sources de votre programme. L’argumentation represente une part importante de votre evalua- tion de ce travail. L’amelioration ou non des performances ne represente qu’une partie des criteres de notation.

Attention, ce TP representera une part importante de la note finale dans cette matiere.

# 2. Telechargement du projet

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

# 3. Lancement du programme

Le code est sous la forme d’un projet maven standard pret a l’emploi. Afin de faciliter les tests, j’ai ajoute 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 a partir des sources
3. Executer le programme a partir d’un fichier html

Les fichiers html permettant de lancer le programme se trouvent dans le repertoire **src/main/html**. Par defaut, le script utilise le fichier *ants default.html*, qui genere les parametres aleatoirement. Il existe un second fichier qui se nomme *ants worst.html* et qui contient les parametres pour tester le programme dans le pire des cas. Pour lancer un test sur un fichier html specifique, vous pouvez ajouter le nom de ce fichier au script *run.sh* :

1

[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 differentes, etc... Vous avez aussi le droit de ne pas utiliser ce script ou de le modifier si besoin.

# 4. Les taches a realiser

## 4.1 prealable de l’application

Dans un premier temps, il vous est demande d’effectuer une analyse complete de l’application. Pour cela, vous devez utiliser tous les outils pertinents et que nous avons vu au cours des TP precedents. L’objectif est de mettre en evidence les potentielles faiblesses de l’application et d’expliquer les differentes modifications que vous pensez y apporter afin de les resoudre. En parcourant le code, vous trouverez probablement des axes d’amelioration qui ne sont pas ou peu visible a l’aide des outils d’analyse. Vous pouvez aussi les inclure dans votre analyse.

## 4.2 Amelioration de l’application

Vous avez le droit d’effectuer toutes les modifications que vous jugez necessaires. N’hesitez pas a experimen- ter pour voir si une implementation est meilleure qu’une autre. Attention toutefois, il ne faut pas modifier le comportement general de l’application, on souhaite uniquement en ameliorer les performances, pas son fonctionnement nominal. Cependant, il est possible de faire certaines concessions sur le fonctionnement de base afin d’ameliorer les performances. Nous discuterons de ces concessions au cours des seances de TP et elles devront toujours etre validees par moi-meme.

A chaque fois que vous effectuez des modifications, il vous est demande d’effectuer une rapide analyse a la suite de celles-ci afin de mettre en evidence les ameliorations que vous y avez apporte, si c’est bien le cas. Dans le cas contraire, si les performances n’ont pas evolue ou si elles sont moins bonnes, il est necessaire d’y apporter toutes les explications necessaires. Il faut aussi que vous expliquiez tres clairement les raisons qui vous pousse a essayer de mettre en place cette amelioration. Il est tres important d’etre tres precis dans les explications que vous fournissez. Une amelioration sans explication precise et detaillee ne sera pas prise en compte lors de l’evaluation, aussi bonne soit-elle.

## 4.3 Analyse finale de l’application

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

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

# 5. Quelques conseils avant de demarrer

Dans cette partie, je rajoute quelques conseils ainsi que des informations complementaires sur le sujet. Vous

etre libre de les suivre ou non.

## 5.1 Refactoring de l’application

Vous allez le constater, l’application est composee de 4 classes. Avant d’essayer d’ameliorer les performances de l’application, il est conseille (mais pas obligatoire) de reorganiser le code afin d’y ajouter des classes et ainsi decharger les autres classes de certaines taches. Cela vous permettra d’avoir des classes plus petites avec moins de fonctionnalites et devrait grandement vous faciliter la tache pour ameliorer les performances. Par exemple, vous pouvez avoir une classe qui ne s’occupe que de la gestion des parametres, une autre qui ne s’occupe que de la gestion de la souris, etc.

Definir un role precis pour chacune des classes vous permettra de vous y retrouver facilement dans l’appli- cation. De plus, cela me facilitera aussi la tache afin de comprendre l’architecture de votre code.

## 5.2 To git or not to git ?

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

La premiere est que cela vous permettra de revenir en arriere si les modifications que vous avez faites ne sont pas satisfaisantes. Si c’est le cas, ne supprimez pas vos commits et creez une nouvelle branche. Toute modification, meme sans resultat, peut-etre interessante si elle est expliquee 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’evaluerai votre projet, cela me permettra de suivre le cheminement que vous avez eu au cours des seances en regardant la suite de vos commits. Enfin, un gestionnaire de version est un outil INDISPENSABLE lorsque l’on developpe des applications. Il est tres utile de versionner des projets, meme si vous etes le/la seul/seule a travailler dessus. Au plus tot 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 arriere sur des modifications sans pour autant ”perdre” ces modifications. Un seul et unique projet doit etre rendu, versionne ou non.

## 5.3 De l’interet des tests unitaires

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

* + 1. Le code fonctionnait bien avant la modifications
    2. Le code fonctionne toujours apres la modification

De maniere generale, je vous encourage a 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 donne 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 detecter.

## 5.4 Vous avez dit Javadoc ?

La documentation d’une application est un aspect primordial. Meme si cela n’ameliore en rien les performances de votre application, cela vous permettra de comprendre plus facilement ce que chacune de vos methodes/classes font, ainsi que le sens de chacun des parametres, meme si vous etes vous-meme 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 methode si elle est correctement documentee.

## 5.5 De l’utilisation des logs

Il peut etre utile de mettre en place un systeme 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, ecrire des logs pertinents est une bonne chose, trop en ecrire est contre-productif. De plus, mettre en place trop de logs ralentira de maniere certaine votre application.

## 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 etes autorise

a inclure les sources de ce projet dans le zip final. Cependant, vous n’avez pas le droit d’ajouter plusieurs projets JMH differents dans le zip. Si vous avez plusieurs tests portant sur differentes choses, creez plusieurs classes dans le meme projet. Pour etre pris en compte dans l’evaluation, ces micro-benchmarks ainsi que les resultats issus de ceux-ci doivent etre presents dans votre rapport, accompagnes de toutes les explications necessaires.