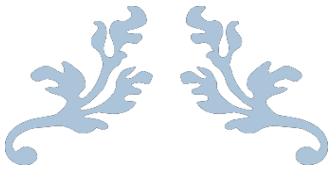
HENRY Nathan
MIRICA Constantin
VANDEN HERREWEGEN Guillaume

21.12.2018
2TL2-G14
E.P.H.E.C. Louvain-la-Neuve



RAPPORT PROJET JAVA

HENRY Nathan
MIRICA Constantin
VANDEN HERREWEGEN Guillaume

21.12.2018
2TL2-G14
E.P.H.E.C. Louvain-la-Neuve

Rapport projet – Démineur JAVA

Développement informatique avancé

Orienté Applications

Table des matières

HENRY Nathan
MIRICA Constantin
VANDEN HERREWEGEN Guillaume

21.12.2018
2TL2-G14
E.P.H.E.C. Louvain-la-Neuve

Introduction - cahier de charges

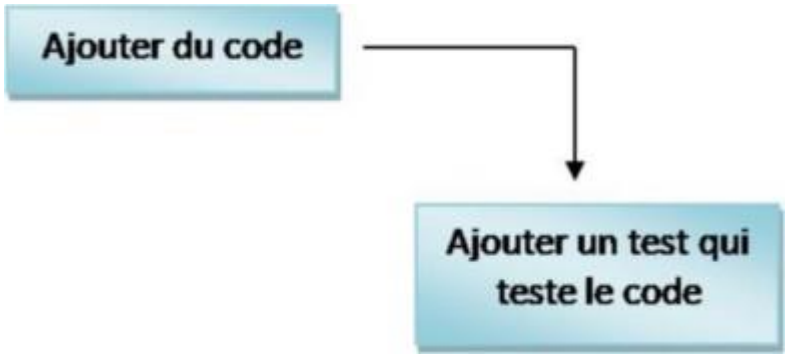
Contexte et définition du projet

Dans le cadre du cours de 2éme, « Développement informatique avancé : application », de la haute école EPHEC, nous avons dû concevoir une application en Java qui nous permettra de développer nos compétences dans la matière. Le projet a duré 3 mois, et pendant ce temps, notre code devait être en permanente évolution par rapport aux nouvelles informations acquises durant le cours.

Objectif du projet

Développer les compétences des membres du groupe pour la programmation dans le langage Java tout en suivant un démarche TDD.

Démarche TDD



http://igm.univ-mlv.fr/~dr/XPOSE2009/TDD/img/PresentationTDDPrincipeBaseSansTDD.jpg

Créer l'application Démineur pour qu'elle puisse être utilisée en réseau par deux utilisateurs, tout en respectant la structure MVC. Chaque joueur sera sur sa propre machine ayant comme simple objectif d'être le plus rapide des deux à finir la partie. La synchronisation se fera grâce aux notions de Socket. Tout le jeu sera visible en console mais aussi dans une interface graphique.

Les deux vont fonctionner en synchronisation grâce au model MVC et au principe de thread.

Description fonctionnelle des besoins

- Respecter l'architecture MVC avec deux interfaces utilisateurs (une interface console et une interface graphique)
- Comporter une communication Socket (les sockets seront vus au cours) OU une interaction JDBC avec une base de données (attention, les étudiants devront alors découvrir JDBC par eux-mêmes).
- Utiliser au moins une structure de données du framework Java Collection (HashMap, List, ...).

Des livrables

Une fois le projet fini, vous aurez accès à une archive contenant un fichier exécutable, un dossier avec l'intégralité du programme et un mode d'emploi.

Les outils

Le logiciel va être fait un Java, avec l'aide du logiciel Eclipse, donc comme cela est bien écrit dans le mode d'emploi, ces 2 logiciels sont indispensables pour le bon fonctionnement du fichier exécutable.

Délais de réalisation

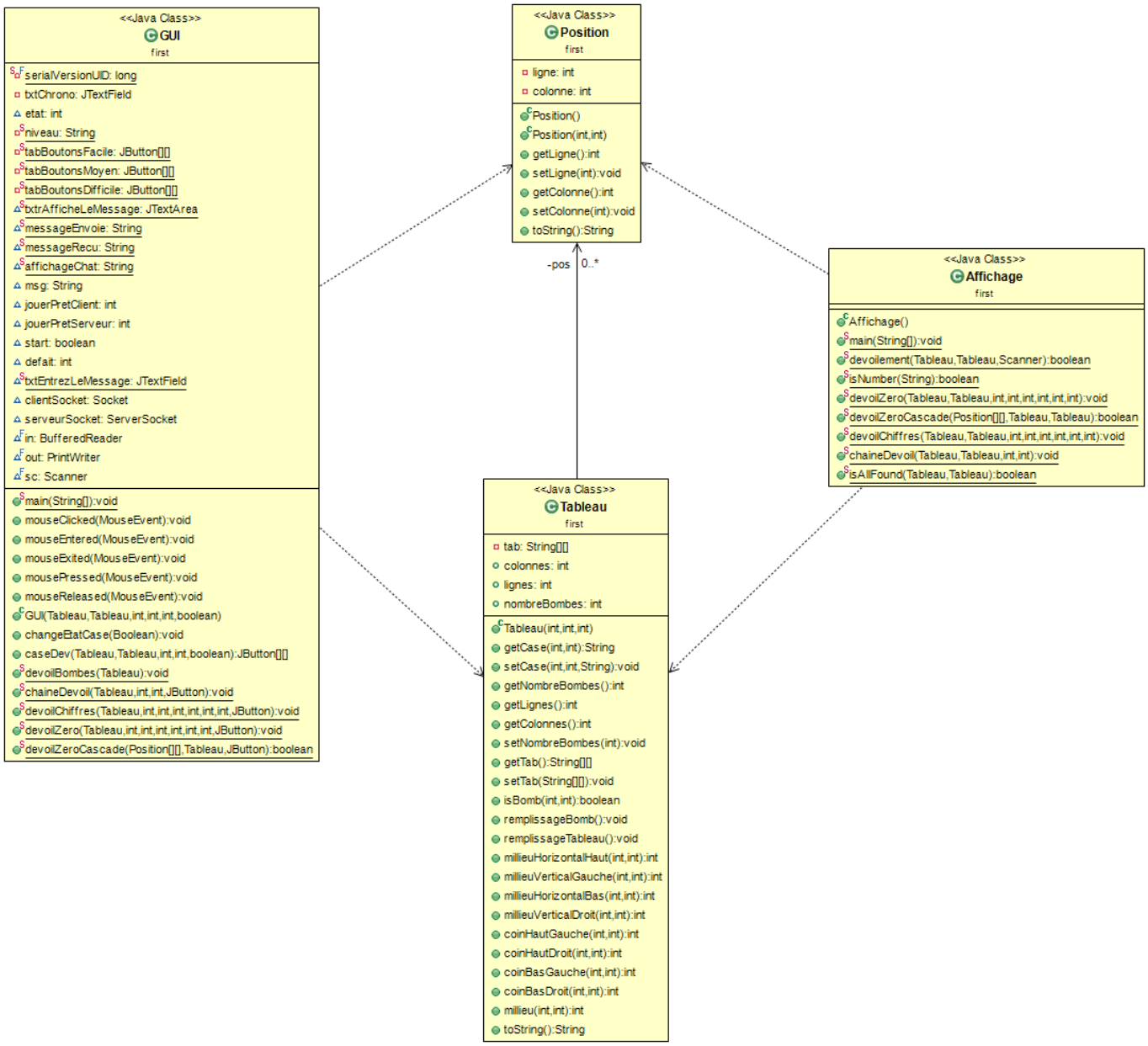
21 Décembre 2018

Le diagramme UML

Ce diagramme est la colonne vertébrale de notre projet car c'est à partir de lui que nous avons décidé la direction d'avancement du projet. C'est à cause

de cela que nous avons dû changer le diagramme trois-quatre fois. Les idées d'évolution du code avaient des répercussions directes sur notre diagramme, car ce dernier devait être fait avant le code.

Image de la diagramme UML



Explication de la diagramme UML

Le « model » de notre projet est composé des classe « position » et « tableau » et ensuite les deux vues qui viennent chercher les informations dans ces deux classes.

Les choix d'implémentation effectué

Explication MVC

Le MVC est un design pattern ce qui veut dire que si on suit le pattern, tous ceux qui le connaissent vont savoir travailler dessus et comprendre très vite.

MVC, signifie Modèle - Vue - Contrôleur. Le pattern MVC permet de bien organiser notre code source. Nous avons essayé de l'implémenter mais malheureusement sans succès. Pendant nos essaie nous avons bien modulé notre code mais malheureusement le lien entre les 2 interfaces ne s'est jamais achevé

Model

Cette partie de notre code s'occupe des donnés, car c'est elle qui va récupérer les informations et qui va les assembler avant de les envoyer au contrôleur.

C'est pour cela que les classes qui vont s'occuper de la gestion du tableau se trouvent dans le model.

Tableau

Dans cette classe nous calculons le nombre de bombe en fonction de la difficulté choisie, ensuite l'algorithme de remplissage mets le nombre autour et quand toutes les données sont finies, le tableau est créé.

Premièrement nous avons décidé qu'une fois le premier tableau créé, nous prenons un deuxième dans lequel on copie les données du premier au fur et à mesure du jeu. Cela sert à créer un tableau référence et un tableau caché qui sera utilisé dans la vue console.

Le problème était que pour faire le MVC cela compliqué énormément la vie, donc maintenant le code fonctionne avec un seul tableau qui est juste caché.

Position

Nous avons décidé de mettre les données de la grille et la position des bombes dans une classe appart car c'est notre point de départ avant de commencer à construire le tableau.

View

Comme son nom l'indique, la « view » s'occupe de ce que l'utilisateur voit, donc des interfaces. A cause des contraintes de notre projet nous devons avoir deux interfaces, une en console et une, graphique.

Les deux interfaces passent par une classe intermédiaire qui elle gère les deux, en leur passant les informations en faisant le lien entre le « model » et les « view ». Ça nous évite de faire chaque fois un model pour chacune des interfaces.

Légende pour la vue console :

+ -> case cachée

X -> bombe

^ -> drapeau

Controller

Le contrôleur est l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les informations, il va les analyser et il va les envoyer à la « view » qui va ensuite les partager plus loin vers chacun des 2 interfaces.

Dans notre cas le contrôleur à un rôle extrêmement important car c'est lui qui gère le model et les views.

Les difficultés rencontrées

Durant tout le long du projet nous avons rencontré des nombreux problèmes qui nous ont empêchaient d'avoir une évolution plus rapide et plus efficace. Ces soucis peuvent facilement être séparés et catégorisés en tant que des difficultés :

Software

1. Le plus gros problème rencontré est le MVC car même a présent, malgré le bon fonctionnement de notre code, nous ne savons pas synchroniser notre vue console avec la vue contrôleur.

2. La conception initiale était remplie de failles ce qui nous a retardé d'une semaine. Notre approche pour la création de la grille étant bien plus difficile à implémenter que l'actuelle.

3. Un deuxième souci a été la compréhension de certains aspects techniques comme pas exemple le MVC, son implémentation étant finie à la dernière minute.

4. L'avancement dans la matière nous ouvraient beaucoup de portes pour des améliorations presque obligatoires, mais cela impliquait aussi un changement parfois vraiment conséquent de notre code.

Organisation

1. Les horaires parfois chaotiques ont été au début un problème pour avoir des réunions de groupe bien productives.

2. Nous avons dû « chasser » des classes libres pendant ces 3 mois pour pouvoir avoir des endroits de réunion.

3. Le fait que deux tiers du groupe habitent très loin de l'école les réunions tardives ou trop matinales étaient presque impossibles à cause des transports en commun.

Les pistes d'amélioration éventuelles

Interface

L'amélioration de l'interface utilisateur est réalisable en mettant encore des informations diverses, comme par exemple :

- le nombre de parties gagnées par l'adversaire
- des informations sur le joueur adverse
- un menu pour mieux structurer la GUI

Bonus

Nous pouvons aussi ajouter des bonus supplémentaires comme par exemple :

- toutes les cinq parties gagnées l'utilisateur reçoit une vie
- si le jeu est fini sans aucun drapeau mis sur expert alors l'utilisateur reçois une vie

Faciliter la connexion

Ajouter un champ dans l'interface où l'adresse IP locale sera affichée dés le lancement du programme. Suite aux renseignements sur Stack Overflow, ça peut se faire grâce à : « InetAddress.getLocalHost() ».

Conclusion individuelle

Nathan

« Pour un premier projet complet fait en groupe, c'était une bonne expérience. Peut-être qu'on aurait pu avoir le cours sur le MVC plus tôt pour ne pas rencontrer le problème de devoir transformer le code sur lequel on a travaillé depuis quelques semaines.

Les pièces du puzzle se sont mises ensembles assez vite, ce qui a permis d'avoir une bonne vue d'ensemble sur le projet.

Ce projet m'a permis de réaliser l'importance de bien diviser le travail et de bien gérer le temps à notre disposition. »

Guillaume

« Pour un premier vrai projet en programmation, je suis satisfait de cette expérience. Les premières lignes de code étaient difficile à concevoir car nous

ne savions pas trop par où commencer. Une fois lancés, tous les objectifs se sont enchaînés sans trop de réelles difficultés, malgré une mauvaise coordination de l'adaptation de notre code avec le modèle MVC. «

Constantin

« Mon premier sentiment quand j'ai entendu parler du projet était la peur. Petit à petit nous avons su mettre les pièces du puzzle ensemble car je me suis mis à trouver des nouvelles idées d'implémentation pour notre projet, comme les débuts de découvertes de cases, algorithmes de découverte de bombes, grille GUI et ainsi de suite.

Je considère que c'était une expérience extrêmement enrichissante pendant laquelle j'ai pu voir monter un projet impressionnât, à mon niveau, mais aussi un groupe bien motivé. »

Conclusion générale

Malgré de nombreux problèmes rencontrés et la complexité du programme, nous avons réussi à proposer un code fonctionnel qui répond à la demande du projet et qui est capable d'accomplir toutes les demandes de base, voir même plus complexes, même si le pattern n'est pas respecté à 100%.

Ce projet nous a vraiment permis, à tous dans le groupe, d'approfondir les fonctionnalités offertes par le langage de programmation JAVA et d'apprendre à utiliser les outils tel que les threads, les GUI builder, mais aussi les API pour solutionner nos problèmes et mener à bonnes échéances ce projet.

Pour notre groupe, ce projet a donc été une excellente opportunité d'approfondir nos connaissances dans le langage JAVA, mais surtout un bon challenge et une excellente opportunité pour nous souder en tant qu'équipe.