Langage avancé de programmation :

Projet de Java 2012 : Démineur.

Sommaire

[1. Introduction 3](#_Toc343513628)

[2. Avant de programmer, il faut un svn… 4](#_Toc343513629)

[3. Présentation du projet 5](#_Toc343513630)

[a) Objectifs 5](#_Toc343513631)

[b) Concrètement 5](#_Toc343513632)

[4. Diagrammes uml du projet 6](#_Toc343513633)

[a) Séparation entre interface graphique et code 6](#_Toc343513634)

[b) Code du projet 6](#_Toc343513635)

[c) Interface graphique 7](#_Toc343513636)

[d) Socket 7](#_Toc343513637)

[5. Fonctionnement du projet 8](#_Toc343513638)

[a) La classe « Fenetre » 8](#_Toc343513639)

[b) 8](#_Toc343513640)

# Introduction

Après le pascal qui fut relativement facile, le C qui était un peu plus compliqué mais assez similaire en terme de programmation. Nous attaquons le projet en java qui, il faut bien l’avouer, nous fais prendre une toute autre direction que les deux précédents langages évoqués ci-dessus.

La logique de programmation en Java est très différente de ce qu’on a pu faire par le passé. La façon de penser, la façon de programmer,… demande un certain temps d’adaptation.

# Avant de programmer, il faut un svn…

Avant même de commencer à réellement travailler sur le projet proprement dit, il fallait mettre en place une solution de gestion du code ainsi que de ses différentes version postées au fur et à mesure de l’avancement des travaux.

UN SVN semblait la solution appropriée pour un projet en groupe bien avant même la « conférence » des deux anciens élèves. En effet, lors du projet en C de première, Tortoise SVN a été utilisé pour mener à bien le projet entre tous les membres du groupe dont nous faisions tous deux partie (nous étions trois au total). Cela a parfaitement fonctionné et nous avons pu tester toutes les fonctions intéressantes des SVN.

Cependant, n’étant pas personnellement en possession d’un serveur privé, il fallait trouver une solution pour déposer notre code sur un serveur distant et de pouvoir le cloner sur nos deux machines. C’est à ce moment que l’exposé des deux anciens de l’Ephec entre en jeu. Ils ont parlé de Github comme étant une solution qui fonctionnait et donc nous avons essayé cela… Tout simplement.

Nous avons donc commencé par créer une organisation qui porte ne nom de « Ephec » dans laquelle nous aurons accès à notre code en lecture et écriture. Cela signifie que d’autres personne peuvent bien entendu voir le code mais ne peuvent pas le modifier directement. Ils doivent passer par des requêtes PUSH qui seront, au cas où il y en a, refusées car il s’agit d’un projet en groupe dans le cadre d’un examen.

Ensuite nous avons synchronisé la totalité du dossier Eclipse contenant le dossier **src** avec les fichiers « .java », avec le dossier **bin** contenant les fichiers « .class » ainsi que les paramètres du projet tel que l’encodage en iso,… Ce qui permet de conserver les mêmes paramètres sur toutes les machines utilisant le projet. Et enfin nous avons ajouté le dossier **document** dans lequel se trouvent ce rapport, l’uml, la javadoc, etc… Il n’est alors plus nécessaire d’utiliser un système de dropbox en plus ce qui facilite d’autant plus la tâche. Noter que dans le dossier bin, les fichiers .class ne sont pas synchronisé car ils se modifient lors de l’exécution de l’application et non lors de la modification du code source. Cela devient vite gênant et nous avons choisi d’exclure ces fichiers de la synchronisation.

Une fois le répertoire Github synchronisé, il nous suffit de créer un nouveau projet Eclipse à l’emplacement du répertoire en question. Les modifications qui sont effectuées sur Eclipse apparaissent instantanément sur Github et inversement. C’est le seul moyen efficace que nous avons trouvé pour avoir un projet identique sur toutes les machines aussi bien sous Windows que sous Mac OS.

# Présentation du projet

## Objectifs

Au départ, nous partions pour un petit utilitaire permettant par exemple de changer les icones des disques dur en allant modifier le registre par exemple. Cependant nous avons vite pris une autre direction. En effet, l’application doit être autonome et utiliser un socket. Nous sommes donc partis sur un jeu avec une possibilité de faire une partie en réseau.

Notre choix s’est donc porté sur le démineur. A première vue, rien de bien original mais en y regardant de plus près, certaines choses varient par rapport au démineur Windows. Voici en quelques points les éléments phares de notre démineur :

* Sélection de différents niveaux prédéfinis ;
* Possibilité de jouer sur une grille personnalisée ;
* Possibilité d’enregistrer les scores de plusieurs joueurs et donc d’effectuer un classement ;
* Possibilité de jouer un réseau. C’est-à-dire que la partie commence en même temps pour les deux joueurs et le premier qui termine à gagner ;
* Pas de mine lors du premier clic afin de « lancer » la partie ;
* Règles élémentaires d’un démineur classique (inutile de toutes vous les citer mais la précédente n’est pas si facile à mettre en place).
* Partie en réseau simple. Les deux joueurs commencent une partie du même niveau en même temps et le premier qui termine gagne la partie.

L’ensemble de ces fonctions sont présentes dans le démineur de Windows avec bien entendu toutes les petites subtilités au niveau interface graphique. Le but de notre démineur est d’être fonctionnel. Le coté graphique pourra être amélioré par après.

## Concrètement

Vu le retard pris lors de la réalisation du projet, certaines fonctions ne sont pas présentes et nous en sommes conscients. Ainsi il n’est à l’heure actuelle pas possible de jouer en réseau bien qu’une tentative de socket figure bien dans le projet sous le package **RESEAU**. Nous expliquerons par la suite pourquoi ces fonctions ne sont pas présentes.

En attendant, notre démineur est totalement fonctionnel et les fonctions manquantes ne gênent pas la jouabilité. Comme dis précédemment, des modifications ultérieures pourront être opérées.

# Diagrammes uml du projet

## Séparation entre interface graphique et code



## Code du projet



## Interface graphique



## Socket



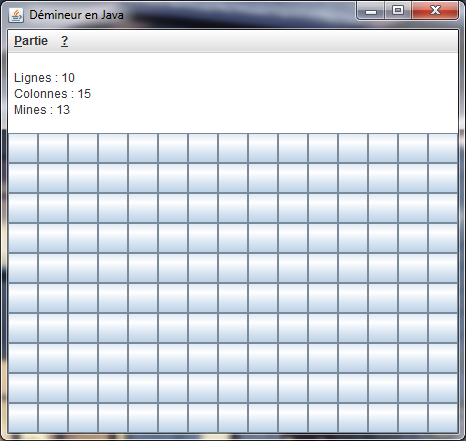
# Fonctionnement du projet

Pour plus d’informations, vous pouvez consulter la javadoc générée par Eclipse se trouvant dans le dossier du projet. La classe **Main** est LA classe principale du projet, elle crée la fenêtre de jeu qui commande les autres éléments de l’interface graphique (**GUI**) et les éléments de code (**DEMINEUR**).

## La classe « Fenetre »

La classe « Fenetre » est la classe principale du package GUI. C’est notre fenêtre de jeu qui crée le menu, crée un grille de jeu par défaut et qui permet de jouer. Elle contient 3 éléments distinctifs :

* Le menu
* Les détails concernant la partie en cours
* Le plateau de jeu



Plateau de jeu

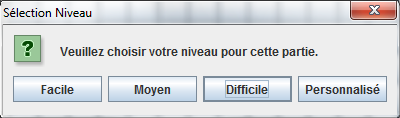
Détails de la partie

Menu

La classe commence par initialiser le layout général du démineur, à savoir, une taille minimale qui s’adapte aux nombres de cases du niveau sélectionné (par défaut : facile). Ensuite le menu est initialisé. Ce menu ne change jamais et reste toujours visible. Enfin la grille est créée avec les valeurs par défaut et est fonctionnelle dès le premier lancement du jeu sans aucune intervention de l’utilisateur.

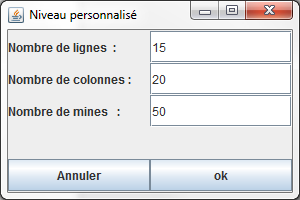
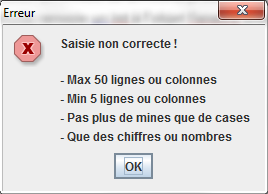
C’est également cette classe qui gère les actions de la souris pour découvrir la grille (clic gauche), poser un drapeau (clic droit) ou encore lancer le chronomètre. A chaque clic, une méthode vérifie si la partie est gagnée ou perdue.

## La classe Niveau



A chaque lancement d’une nouvelle partie, il faut choisir un niveau de difficulté prédéfini ou un niveau personnalisé si elles ne conviennent pas. Cette méthode renvoie un int à l’objet Fenetre avec le niveau choisi.

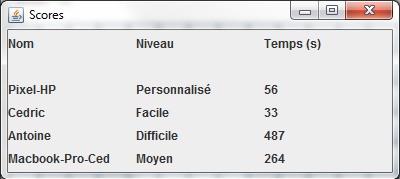
## La classe NiveauPerso

Cette boite de dialogue est appelée lors de la sélection de l’entrée « Personnalisé » dans la fenêtre précédente. Elle permet d’entrer le nombre de lignes, de colonnes ainsi que de mines selon les désirs de l’utilisateur en prenant soin de respecter certaines limites.

Ainsi, il n’est pas possible de laisser les champs vide, d’entrer plus de 50 lignes et 50 colonnes et d’entrer plus de mines que de cases. Dans ces cas, la fenêtre d’erreurs (en haut à droite) apparait.

## La classe ScoresVue

Cette classe affiche les scores en provenance du fichier texte. Avec dans l’ordre, le nom du joueur, le niveau de jeu ainsi que le temps. Si le nombre de lignes dans le fichier est supérieur à 10, seuls les 10 derniers scores enregistrés sont affichés dans la fenêtre pour des raisons évidentes de lisibilités.