Compte rendu projet Programmation Objet

Alexandre RENOUX - Pierre-Emmanuel NOVAC

13 janvier 2016

1 Présentation

Le jeu se compose d'une grille et d'une liste de mots qui doivent y être placés. En mode graphique, l'utilisateur doit donc cliquer sur un mot pour le sélectionner puis cliquer sur un emplacement de la grille pour le placer. Un clic gauche effectue un placement horizontal, un clique droit effectue un placement vertical. Pour être placé, le mot ne doit pas dépasser de la grille et doit être placé sur des cases libres (c'est à dire contenant toujours un chiffre). Dans ce cas le score est calculé en sommant les chiffres présents sur les cases que le mot écrase. Le mot est alors affiché dans la grille et supprimé des choix, et le joueur passe donc au placement d'un autre mot.

Le jeu se termine lorsqu'il n'y a plus de mot à placer, où lorsque le placement d'un mot est impossible, c'est à dire quand il n'existe plus d'emplacement libre au moins aussi grand que le plus petit mot qu'il reste à placer. Le but est donc de placer un maximum de mots en cumulant un score le plus élevé possible.

Par défaut, le jeu démarrera en utilisant les mots du fichier *wordsList.txt* ainsi qu'une taille de grille égale au mot le plus long dans la liste. Des paramètres passés en ligne de commande permettent de modifier cette configuration. De plus le programme demandera à l'utilisateur s'il veut utiliser le terminal ou l'interface graphique ; le paramètre *-i* force l'utilisation du terminal. Il est donc possible de lancer le jeu dans le terminal avec le fichier *jeu1.txt* et une taille de grille de 10x10 caractères avec la commande suivante :

```
java Project -i -s 10 jeu1.txt
```

L'interface graphique propose un menu rappelant les règles du jeu ainsi qu'un bouton pour recommencer la partie.

2 Organisation du développement

En anglais : Au départ nous avions des noms de variables et de méthodes en français et d'autres en anglais. Dans un soucis d'homogénéité, nous avons fait le choix de n'utiliser que des noms anglais, les commentaires sont donc eux aussi rédigés en anglais. Cela permet d'autre part à un nombre plus important d'utilisateurs d'être en mesure de comprendre le code et les commentaires associés, l'anglais étant fortement privilégié en tant que langue internationale.

Javadoc : L'intégralité des classes et des méthodes possèdent des commentaires au format Javadoc. Un répertoire *javadoc/* est donc mis à disposition et contient la Javadoc générée à partir de l'ensemble du code et propose alors une vue globale du code du projet. En revanche certaines méthodes complexes contiennent des commentaires plus spécifiques pour certaines instructions et n'apparaissent donc pas dans la Javadoc.

Répartition du travail : Jusqu'à la création des classes Game et DisplayGame, nous travaillions ensemble. Cela comprend une première réalisation des classes *Project*, *Grid*, *WordsList*, *DisplayGrid*, *DisplayWordsList*, *Window*. Nous avons par la suite réparti plus distinctement le développement puisque nous ne pouvions plus travailler en permanence ensemble. Les classes *Game*, *DisplayGame* ont donc été réalisées par Pierre-Emmanuel. Alexandre à quant à lui créé *Terminal* et *MenuBar*. Les autres classes ont dû être complétées par chacun de nous au cours du développement.

3 Structuration du code

Nous avons essayé au maximum de procéder à une séparation cohérente du code. Notre première idée a été de séparer la grille et la liste des mots de leurs méthodes réservées à l'affichage. Cela a donné lieu à la création des classes *DisplayGrid* et *DisplayWordList*. Cela nous a permis de créer sans trop de difficultés une version réservée au terminal pendant le développement. (La première version du jeu était exclusivement graphique.)

Finalement, nous nous sommes retrouvés avec ces différentes classes :

- **Grid** qui s'occupe de gérer le stockage en mémoire de la grille.
- **DisplayGrid** qui dessine la grille, les nombres et les lettres dans les cases sur l'interface graphique.
- **WordsList** qui permet de récupérer les mots d'un fichier texte et de les insérer dans un tableau réutilisable.
- **DisplayWordsList** qui affiche un bouton pour chaque mot de la liste.
- **Window** qui hérite de *JFrame* pour créer la véritable fenêtre.
- **Project** qui contient la méthode *main()* et gère alors la reconnaissance des arguments passés en ligne de commande et le lancement de l'interface graphique ou terminal.
- **Terminal** qui propose des méthodes pour reconnaître et traiter les commandes entrées par l'utilisateur en mode terminal.
- **Game** s'occupant de l'interaction entre la liste des mots et la grille, ainsi que du lancement de la fin du jeu.
- **DisplayGame** qui gère le placement graphique des mots et repose sur *Game*.

Java 8 : Dans plusieurs méthodes, des nouveautés de Java 8 ont été utilisées, comme les références vers des méthodes, les expressions lambda ou les *Stream*. Toutes ces méthodes auraient pu être écrites sans utiliser ces fonctionnalités du langage, mais il fut très intéressant d'expérimenter cette façon de programmer.

4 Difficultés

Une des difficultés a été d'accéder aux différents attributs des classes *DisplayWordList*, *DisplayGrid*, *WordsList*, *Grid* afin de pouvoir mettre les mots de la liste de mots dans la grille. Cette difficulté était dû à la séparation de ces éléments en classes distinctes. Cela nous a mené à créer deux nouvelles classes : *DisplayGame* qui gère les évènements souris sur la grille pour lancer le placement d'un mot, qui, s'il réussit, mène à la désactivation du bouton concerné. *Game* qui permet de relier la liste de mots (classe *WordsList*) à la grille (classe *Grid*) et de traiter la fin du jeu. Ces 2 classes servent donc d'interface entre la liste de mots et la grille. Nous avions au début pensé à partager certains attributs, mais finalement nous avons conclu que cette solution serait plus adaptée.

L'organisation de l'interface graphique n'a pas été des plus simples. Nous avons mis du temps à trouver les bonnes combinaisons de layout et les bon paramètres pour obtenir une interface cohérente.

La méthode à exécuter à la fin du jeu étant appellée au même endroit mais différente pour les deux interfaces, notre solution de contournement à été d'ajouter un attribut *end* à *Game*. C'est en réalité une référence vers une méthode de *Window* ou *Terminal*, qui est affectée à l'aide d'un setter lors du choix du lancement de l'une des deux interfaces.

Pour des questions d'accès aux attributs, la réalisation des méthodes *reload()* et *end()* de *Window* n'a pas été facile. Notre solution de contournement a été d'insérer des expressions lambdas. Ce n'est peut être pas la solution la plus propre mais elle convient assez bien pour notre utilisation et permet d'exploiter ces nouveautés impressionantes de Java 8.

5 Améliorations

Nous aurions souhaité ajouter différentes fonctionnalités comme par exemple :

- Un mode multijoueur où deux joueurs s'affrontent sur la même grille. Celui remportant le plus de points gagne la partie. Un bonus serait attribué au joueur qui termine la partie.
- Un tableau de scores qui enregistre les meilleurs scores par joueur.
- Une configuration graphique de la taille de la grille ainsi que du fichier à utiliser pour la liste des mots.

Nous avons préféré nous garder un nombre de fonctionnalités réduit et plutôt nous concentrer sur la qualité du code. Cela réduit aussi le risque de laisser des bogues dans le programme.

6 Conclusion

Ce projet, bien qu'à première vu simple, a tout de même demandé un fort investissement. Nous avions pensé à un certain nombre de fonctionnalités que nous n'avons pas pu intégrer, mais nous pensons tout de même essayer de les programmer plus tard, puisque nous aurions aimé approfondir ce projet et ajouter des originalités par rapport au sujet. Nous avons essayé de faire au maximum un code clair et structuré ce qui nous a forcé à beaucoup réfléchir et nous nous sommes alors confrontés à plusieurs difficultés. Mais dans l'ensemble nous sommes arrivés au résultat que nous attendions. Cela nous a permis de gagner en expérience de programmation, qui nous sera très probablement utile dans le futur.