Makkes Malik
Khalfallah Anis

UNIVERSITE DE CLERMONT AUVERGNE

LICENCE 2 INFORMATIQUE 2021-2022

Rapport du Jeu Puissance 4

Avant de commencer l'explication et la démarche prise afin de coder le jeu voici le diagramme UML des classes, et plus tard en vert on aura les classes et interfaces et en rouge les bibliothèques :

Makkes Malik
Khalfallah Anis

D'abord il faut savoir que nous avons décidé de coder le jeu puissance 4 à l'aide de la bibliothèque swing et awt de Java. Puisque vu qu'au 1er semestre nous avons codé le jeu 2048 a l'aide de la bibliothèque SDL en C (similaire à la bibliothèque awt de Java) nous avons décidé de faire la même chose en Java, afin de rendre le jeu jouable et agréable.

Plongeons maintenant dans le choix des classes et l'implémentation du jeu puissance 4. Vu que nous avons fait un jeu avec une interface graphique il était impossible de choisir un modèle ou l'on faisait qu'hériter les classes grilles ou joueurs, vu que nous avons une interface graphique, et nous implémentons les interfaces Serializable pour la sauvegarde de la grille et ItemListener afin d'attendre que l'utilisateur click sur les boutons pour jouer et nous héritons aussi la classe JPanel pour organiser les boutons. C'est pour cela que nous avons choisi d'utiliser la composition à bon escient.

Maintenant expliquons ce que chaque classe fait exactement. Tout d'abord nous avons la classe Grille, avant toute chose on retrouve serialVersionUID, et on va la retrouver dans chaque classe car elle sert à indiquer le type de la chaine de bytes afin qu'on puisse sauvegarder la grille plus tard, et pour que ça fonctionne, serialVersionUID doit être égale à 1L(n'importe quel valeur unique) partout. Ensuite on initialise le tableau avec une EnumMap et on connecte chaque case avec les 6 cases qui sont à côté d'elle à l'aide d'une énumération Position contenant les 8 voisins dans tous les sens possibles. Par la suite on remplit chaque case par 'V' qui signifie le jeton vide. Après on a toutes les méthodes nécessaires à la manipulation du tableau pour le fonctionnement du jeu et il est important de comprendre que la méthode initJoueurs tire la couleur d'un jeton pseudo-aléatoirement (J = Jaune et R = Rouge) et fait la même chose pour qui sera le 1er joueur à jouer, car le 1er joueur qui joue à toujours un avantage donc on laisse le choix à la chance. Et enfin dans le 2ème constructeur on injecte le jeton dans la grille que le joueur choisit à chaque fois.

Ensuite nous avons la classe Jeu, qui servira a l'enregistrement d'une partie, elle stock la grille et le nom de la partie, juste après on a la classe JeuData qui stock les parties sous forme d'une pile, pour qu'on puisse y accéder plus tard après avoir sauvegarder une partie, et on a enfin la classe SauvegardeJeu qui utilise les deux classes précédentes afin d'enregistrer la partie en demandant a l'utilisateur le nom de cette dernière, mais pour l'enregistrement ca sera un fichier sérialisé, d'où l'utilisation de implements Seriazable dans toute les classes concerné, qui converti l'objet grille en un flux d'octets et le stock dans la mémoire en tant que fichier afin de pouvoir l'utiliser (restaurer une partie) en cas de besoin et le désérialisé.

Passons aux classes qui servent au graphisme maintenant, d'abord nous avons la classe GraphiqueAccueil ou l'on a décidé de donner une taille de 850px en largeur et 650px en hauteur pour la fenêtre d'accueil ainsi qu'une méthode pour ferme cette fenêtre. Par la suite nous avons la classe Menu qui contient les boutons cliquables par le joueur en cours de partie soit pour enregistrer la partie, accéder à l'aide, ouvrir une partie déjà enregistrer ou quitter, et bien évidemment il peut aussi change de mode entre sombre et clair ainsi qu'un bouton pour quitter la partie bien qu'on peut juste cliquer sur la croix pour quitter le jeu. Et enfin, la dernière classe pour le graphisme est la classe GraphiquePartie qui donne la taille de 700px et 600px pour la fenêtre de

jeu contenant la grille et contient une inner class FermetureFenetre. qui ferme complétement le jeu lorsque le joueur clique sur la croix ou s'il ne souhaite plus rejouer à une nouvelle partie.

Parlons maintenant des classes Accueil et Partie, commençons par la 1^{ére} étant la classe Accueil qui vient avant, elle contient les messages a afficher sur la fenêtre d'accueil afin que le joueur sache quoi choisir pour jouer, et dedans nous avons 2 inner classes qui sont InfosJoueur et ChargePartie, on a implémenté l'interface ItemListener de Java pour pouvoir attendre que l'utilisateur clique sur les choix qui lui sont proposé et lorsqu'il clique sur 1vs1 on lui affiche les cases pour saisir les noms des 2 joueurs mais s'il choisit de faire 1vsCPU, on l'autorise juste a saisir son nom et on remplit la case du 2^{éme} joueur par Bot, et enfin s'il clique sur ChargePartie alors on affiche un bouton pour charger une partie. Et c'est deux inner classes servent principalement à l'affichage ainsi que la validation des saisies de l'utilisateur. Maintenant passons à la classe qui nous sert pour la Partie de jeu, d'où son nom Partie, elle sert principalement à positionner les éléments sur la fenêtre correctement après avoir pris la taille de l'écran de l'ordinateur sur lequel le jeu tourne de la classe Main, mais pour l'affichage nous aurons 3 parties, d'abord le menu tout en haut qui contient les éléments vues précédemment, ensuite au milieu nous avons la grille ainsi que les boutons insérer les jetons et elle affiche si y a un gagnant ou si c'est une égalité et propose le choix de rejouer une nouvelle partie ou de quitter le jeu enfin en bas nous avons le message du joueur qui doit jouer. Et pour cela nous avons créé 2 inner classes, chaque partie de la fenêtre a sa propre classe, la classe principale Partie prend la fenetreEcran qui contient directement le menu en haut de la fenêtre crée précédemment avec la classe Menu, ensuite la classe fenetreGrille est la partie du milieu de la fenêtre de la partie, et enfin la classe MessageJoueur est la partie du bas de la fenêtre de la partie.

Pour l'avant dernière classe nous avons la classe Joueur qui contient toutes les méthodes nécessaires qualifiant un joueur, comme son numéro, son pseudo, la couleur de son jeton ainsi que le joueur qui est actif c'est-à-dire qui doit joueur.

Enfin nous allons finir par la dernière classe qui est le Main, dedans on commence par prendre les dimensions de l'écran sur lequel le joue va tourner. Ensuite on a les méthodes qui vont faire fonctionner le jeu, la première étant lancer Jeu, qui initialise la grille et la connecte avec toute les casses et lance la partie, ensuite on a les méthodes basiques pour n'importe quel jeu, du type fermerJeu, nettoyerJeuOuverture qui nettoie la place occupé par la grille dans la mémoire pour qu'on initialise une nouvelle partie, ou encore les méthodes sauvegarder et ouvrir qui permettent à l'utilisateur d'enregistrer la partie ou d'en ouvrir une qui a été déjà enregistré. Et enfin la méthode static main exécute le jeu.

Merci pour avoir tester notre jeu