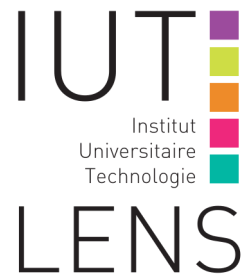


Projet Tutoré 2021

Rapport Projet "Labyrinthe"

par l'équipe Tockada



Equipe Tockada : Albéric Simon

Boubeker Ibrahim

Hoest Edouard

Ricquet Alicia

Introduction	2
I/ Explication algorithmique :	2
Méthode genereEntier :	2
Méthode calculeChemin :	2
Méthode lancer :	3
II/ Utilisation de connaissances :	4
Module COO :	4
Module POO :	4
III/ Répartition du travail et méthodes :	5
répartition du travail :	5
Methodologie :	5
Rôle du tuteur :	6
IV/ Retours :	6
Retour global :	6
Retours personnels :	7
Conclusion :	8
Annexes :	9

Introduction

Les consignes du projet de ce semestre sont de réaliser un jeu en se basant sur un jeu de plateau qui existe réellement.

Pour ce projet chaque équipe devait se trouver un nom, nous avons décidé de nous nommer "Tockada" en référence à un jeu de société qui existe au Canada car tous les membres de l'équipe souhaitent effectuer un semestre à Chicoutimi.

I/ Explication algorithmique :

a) Méthode genereEntier :

Cette méthode est assez simple, on rentre en paramètre la valeur maximum que l'on souhaite et on utilise un générateur de nombre appelé random qui va nous sortir un nombre compris entre 0 et max+1. Nous devons mettre max+1 sinon la valeur maximum n'apparaîtra jamais. Le générateur de nombre aléatoire utilise une fonction déjà présente dans la classe utils qui nous a été donnée. La fonction utilisée s'appelle nextInt et permet de renvoyer un nombre de manière pseudo-aléatoire entre 0 (inclus) et max (exclu) (cf: annexe 1).

b) Méthode calculeChemin :

Dans la méthode 'calculeChemin' on commence par initialiser un tableau de booléen à 2 dimensions représentant le plateau (la valeur à l'index [0][0] correspond à la première pièce en haut à gauche sur le plateau).

La valeur de la case de départ placée en paramètre passe à true pour indiquer que la case a été visitée.

un Vecteur est initialisé pour le chemin retourné à la fin de la méthode

La case de notre position actuelle est définie comme étant la case de départ que l'on ajoute au vecteur chemin.

Ensuite la boucle while commence et continue tant que le vecteur chemin n'est pas vide (ce qui signifie qu'il n'y a pas de chemin vers la case d'arrivée choisi par le joueur) la boucle s'arrête donc à une seconde condition qui est le passage à la valeur true de la case d'arrivée.

Nous testons ensuite toutes les possibilités de chemin en commençant par le haut donc pour savoir si il y a un chemin possible vers une case à côté et que la case n'est pas déjà visité si cette condition est remplie la case passe passe donc à true dans le tableau de booléen la position actuelle change et la case est ajoutée au chemin.

Si aucun chemin n'est possible, on retire la case actuelle du chemin et on revient sur la précédente.

Pour finir si un chemin est possible on met ce chemin dans un tableau qui est retourné null si le chemin n'existe pas. (*cf: annexe 2*)

c) Méthode lancer :

Pour ne pas écrire à chaque fois 'elementPartie.get'..., des variables sont initialisées ainsi qu'une autre variable pour le gagnant et deux tableaux pour le chemin.

Une boucle while tourne tant qu'un joueur n'a pas gagné, à l'intérieur de celle-ci une boucle for avec un tour de boucle qui correspond donc au tour d'un joueur.

Dans ce tour si le joueur est un joueur Humain un message est affiché avec le nom du joueur et un clic est demandé pour commencer à jouer.

Nous demandons ensuite au joueur de choisir une orientation pour la pièce libre en cliquant dessus il sélectionne une flèche sur laquelle sur le plateau.

Ensuite l'insertion est faite à l'endroit voulu avec l'orientation souhaitée grâce à une variable choix correspondant au numéro de la flèche sélectionnée.

Les modifications du plateau sont faites avec les nouvelles coordonnées des objets, joueurs et pièces sur le plateau.

Un message demandant au joueur de sélectionner une case d'arrivée est affiché. Il suffit donc qu'il clique sur une case, si le chemin vers cette case est impossible, nous demandons de nouveau de sélectionner une case .

Sinon des billes sont placées sur le plateau avec le chemin fait par le joueur, nous affichons à présent sa nouvelle position.

Nous avons ensuite globalement la même chose mais en ce qui concerne un joueur Ordinateur de n'importe quel type.

Une indication du joueur à qui c'est le tours est affichée, le choix de la flèche est fait selon la méthode 'choisirOrientationEntree' avec en plus une indication sur la flèche sélectionnée pour les joueurs Humain si il y en a.

Une fois que le tour du joueur est fait, peu importe son type, les billes sont effacées du plateau ainsi que les objets des joueurs qui deviennent transparents et sont enlevés du plateau si le joueur est sur la case de l'objet qu'il doit récupérer.

Pour la fin de la boucle on regarde si un des joueur a gagné si c'est le cas pour le joueur numéro 0 (ou 1 si il y a 3 joueur) on sort de la boucle while car le jeu est terminé si un joueur récupère tous ces objets sans donc attendre les tours des autres joueurs. (cf : annexe 3)

II/ Utilisation de connaissances :

a) Module COO :

Les cours de COO qui signifie Conception Orientée Objet nous ont été dispensés à partir de la deuxième partie du semestre 2. Durant ceux-ci nous avons appris à manipuler des diagrammes et à les réaliser nous-même à partir d'un énoncé.

Durant le projet nous avons dû joindre dans un dossier plusieurs diagramme de classes à un instant T de notre projet, même si cela pouvait être fait directement par un IDE, le diagramme permettait d'avoir une vision plus globale de notre projet et de comprendre comment les différents éléments codés jusqu'à présent interagissent entre-eux. (cf : annexe 4)

b) Module POO :

Le module de POO (qui signifie Programmation Orientée Objet) est peut-être celui qui dans un sens nous a été le plus utile. Effectivement, dans ces cours nous avons appris à utiliser le langage de programmation java, qui nous a servis tout au long de ce projet.

Durant les CM, Td et Tp, nous avons pu découvrir comment était construite une classe, comment initialiser des variables ou encore comment s'en servir à l'aide de méthodes.

III/ Répartition du travail et méthodes :

a) répartition du travail :

Lors de la première semaine, nous nous sommes mis au point sur les forces et faiblesses de chacun en programmation.

Edouard étant le plus doué parmi nous s'est occupé des classes et méthodes les plus complexes. Alicia s'est occupée de la partie graphique lorsqu'elle a fait les sprites pour notre thème. Ibrahim et Simon se sont occupés des petites parties à programmer ou à modifier.

Cependant pour le projet est un intérêt pour nous tous Édouard a décidé de nous réexpliquer la construction de main en détails et nous a permis de nous améliorer en programmation java.

b) Méthodologie :

Nous avons organisé une réunion au lancement du projet pour se mettre d'accord sur les logiciels que nous allions utilisés et la manière de rester en contact entre nous et notre tuteur.

Pour la programmation nous avons convenu sur l'utilisation de VisualStudio Code pour une homogénéité sur git (*cf : annexe 5*).

Pour rester en contact nous avons utilisé Discord (*cf : annexe 6*) et nous avons créé un groupe nommé projet tutoré S2. Nous nous mettons au point toutes les semaines pour nous répartir le travail, tout le monde a toujours quelque chose à faire à chaque étape.

c) Rôle du tuteur :

Pour le premier contact, notre tuteur M.Roussel nous a contacté en premier via nos boîtes mail (cf : annexe 7). Le protocole aurait voulu que le premier mail soit envoyé par notre chef de projet mais nous nous sommes vite laissé déborder au début du projet.

Chaque semaine avant le rendu nous avons essayé de faire une réunion sur l'application Zoom avec notre tuteur pour se mettre au point sur les avancées ou difficultés rencontrées. Lorsque nous avions un problème dans notre code ou incompréhension les réunions étaient d'une grande aide.

IV/ Retours :

a) Retour global :

De manière générale, la réalisation de ce projet nous a permis à tous de faire des progrès en java, pour certains plus considérables que d'autres.

Le projet nous a permis aussi de voir que l'on pouvait compter les uns sur les autres, à chaque étape les forces de chacun étaient mises en avant et ceux qui possédaient des difficultés pouvaient avoir le soutien des autres pour avancer sur sa partie.

La création du jeu Labyrinthe nous a aussi permis d'en savoir un peu plus sur nous-même, ce qui nous intéresse ou non en informatique.

b) Retours personnels :

Simon :

Cette année j'ai pu découvrir le java au global, et sous des formes diverses via les différents TP et TD qu'on a pu faire. Bien que j'ai eu pas mal de difficultés pour comprendre et y arriver, j'ai pu, notamment grâce au projet, comprendre et mieux approfondir mes connaissances en java, notamment pour ne pas pouvoir quoi que ce soit au projet et ne pas apporter ma participation.

Que ce soit l'amélioration des connaissances qu'on a vus en cours, l'optimisation de code, les méthodes et constructeur, les boucles et autres, tous ces éléments font partie de ce que j'ai appris et mieux compris grâce au projet, que je n'aurais pas pu réussir si il y avait simplement eu les cours.

Ibrahim :

Cette année, j'ai appris comment coder en java, durant les CM, les TD et les TP. Ce projet labyrinthe m'a permis d'approfondir mes connaissances mais également de mieux comprendre comment le système java fonctionne. Je me suis amélioré sur l'utilisation des tableaux, comment les classes échangent ensemble, l'utilisation d'une librairie et aussi qu'il faut rétrécir un maximum un code pour éviter les actions inutiles. De plus, il faut privilégier un code compréhensible pour tout le monde car c'est un projet collaboratif. Nous étions bien guidés durant l'avancée de ce projet, soit par les consignes de M. Condotta ou les conseils de notre professeur référent M. Roussel.

Edouard :

Lors de cette réalisation du labyrinthe en code j'ai appris et compris beaucoup de choses en java que j'avais eu des difficultés à assimiler avec les cours ou les Tp/Td. Cela m'a permis d'apprendre que la réalisation d'un petit jeu peu difficile sur le papier pouvait être assez compliqué à coder et je vois un peu mieux ce que pourrait être la réalisation d'un jeu plus compliqué et avec beaucoup moins d'aide.

Alicia :

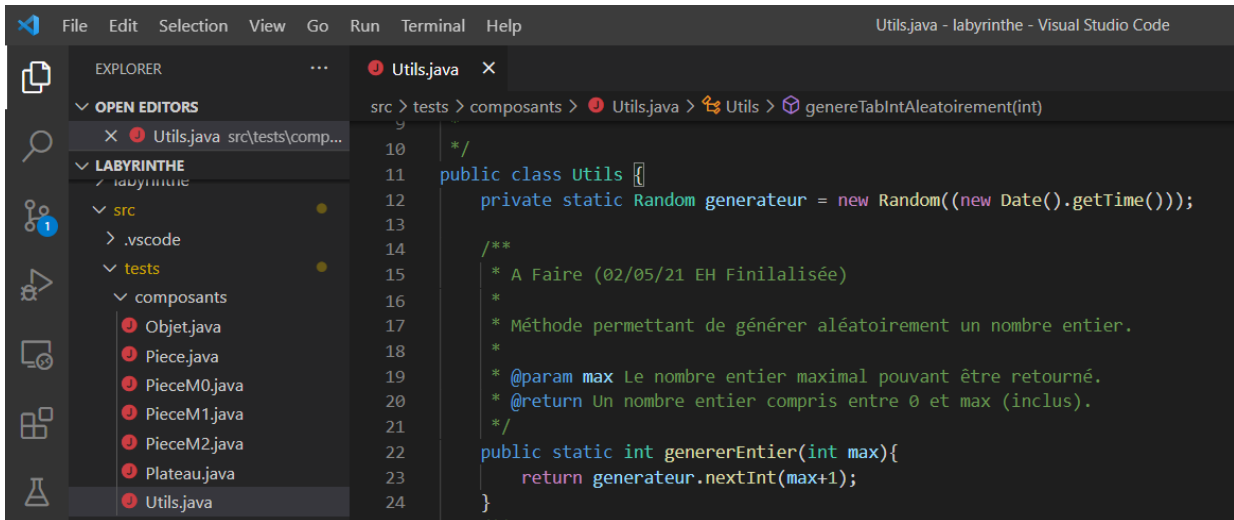
Pendant toute la durée du projet, j'ai pu apprendre plus de choses, surtout en programmation car j'ai des difficultés en POO mais Edouard et M.Roussel ont passé beaucoup de temps à me réexpliquer certains points qui m'ont beaucoup servi par la suite pendant les cours. J'avais des doutes au début car le projet ne plaisait pas et je pensais que je n'aurais jamais le temps de m'y consacrer pleinement. Cependant à ma grande surprise même si parfois c'était compliqué j'ai adoré passé du temps sur le projet et je suis fière de ce que l'on a pu accomplir tous ensemble.

Conclusion :

Pour conclure, ce projet nous a été bénéfique sur plusieurs points, que cela soit sur nos capacités à programmer en java ou la relation au sein du groupe car nous avons pu durant l'entièreté de cette partie de semestre compter les uns sur les autres lorsque nous étions face à des difficultés. Nous avons pu découvrir une autre facette de nos études grâce à la conception de ce labyrinthe.

Annexes :

Annexe 1 : Code méthode genererEntier



```
File Edit Selection View Go Run Terminal Help
Utils.java - labyrinthe - Visual Studio Code

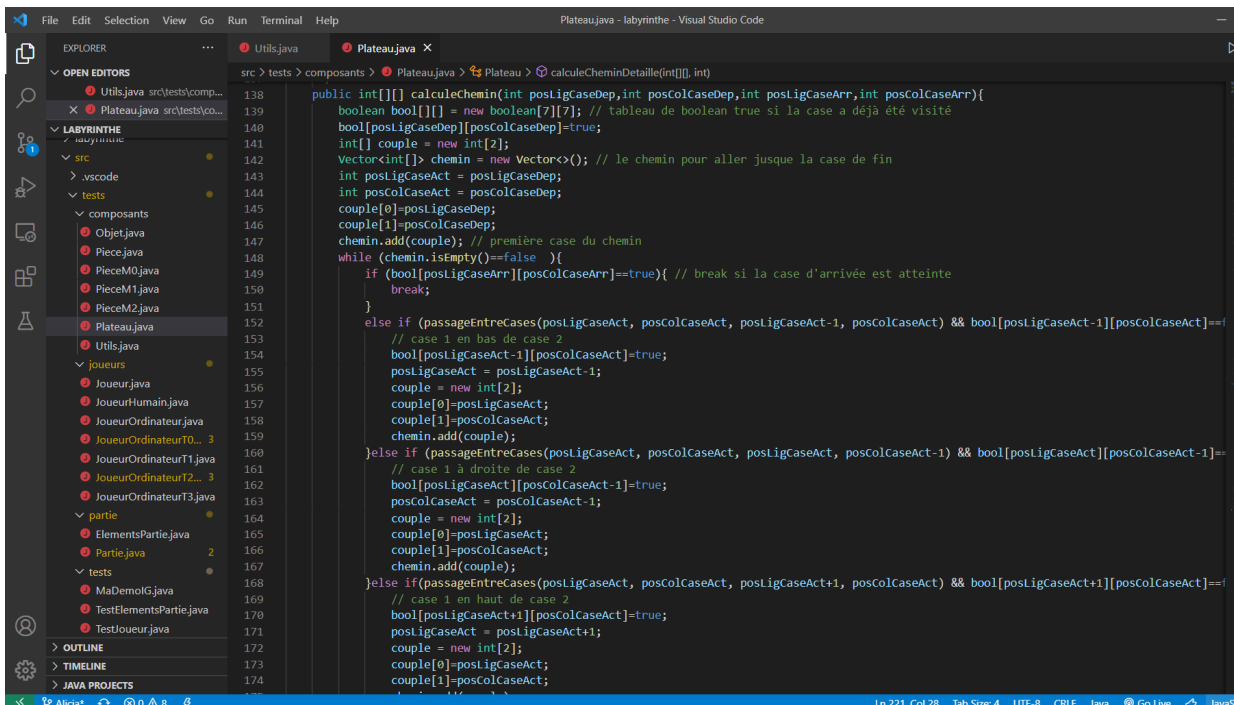
EXPLORER
OPEN EDITORS
  x Utils.java src\tests\comp...
LABYRINTHE
  labyrinthe
    src
      .vscode
      tests
        composants
          Objet.java
          Piece.java
          PieceM0.java
          PieceM1.java
          PieceM2.java
          Plateau.java
          Utils.java

src > tests > composants > Utils.java > Utils > genereTabIntAleatoirement(int)
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

/**
 * A Faire (02/05/21 EH Finalisée)
 * Méthode permettant de générer aléatoirement un nombre entier.
 * @param max Le nombre entier maximal pouvant être retourné.
 * @return Un nombre entier compris entre 0 et max (inclus).
 */
public class Utils {
    private static Random generateur = new Random((new Date()).getTime());

    /**
     * Méthode permettant de générer aléatoirement un nombre entier.
     * @param max Le nombre entier maximal pouvant être retourné.
     * @return Un nombre entier compris entre 0 et max (inclus).
     */
    public static int genererEntier(int max){
        return generateur.nextInt(max+1);
    }
}
```

Annexe 2 : code méthode calculeChemin



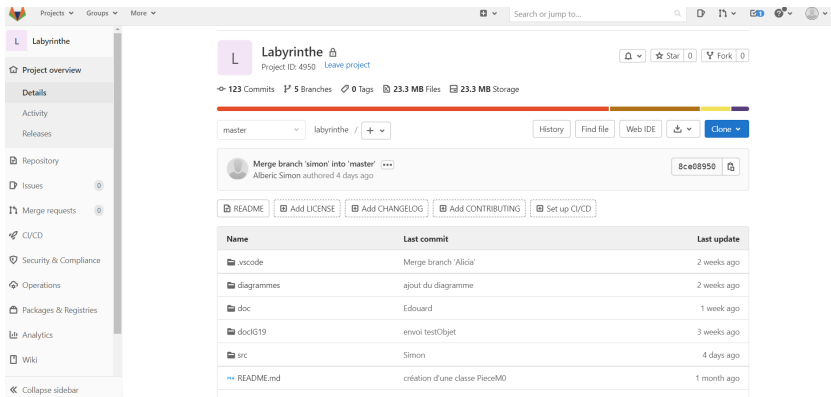
```
File Edit Selection View Go Run Terminal Help
Plateau.java - labyrinthe - Visual Studio Code

EXPLORER
OPEN EDITORS
  x Utils.java src\tests\comp...
  x Plateau.java src\tests\co...
LABYRINTHE
  labyrinthe
    src
      .vscode
      tests
        composants
          Objet.java
          Piece.java
          PieceM0.java
          PieceM1.java
          PieceM2.java
          Plateau.java
          Utils.java
        joueurs
          Joueur.java
          JoueurHumain.java
          JoueurOrdinateur.java
          JoueurOrdinateurT0... 3
          JoueurOrdinateurT1.java
          JoueurOrdinateurT2... 3
          JoueurOrdinateurT3.java
        partie
          ElementsPartie.java
          Partie.java
        tests
          MaDemolG.java
          TestElementsPartie.java
          TestJoueur.java
    OUTLINE
    TIMELINE
    JAVA PROJECTS

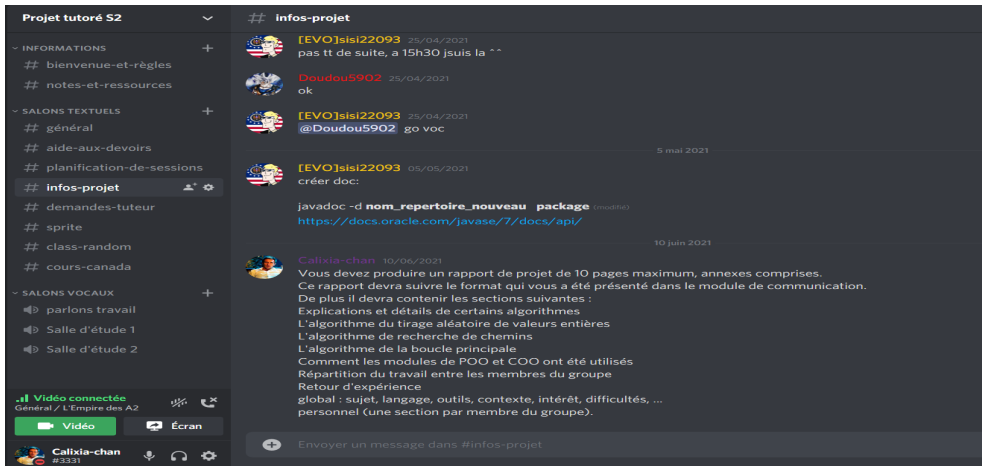
src > tests > composants > Plateau.java > Plateau > calculeCheminDetaille(int[][], int)
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

public int[][] calculeChemin(int posLigCaseDep, int posColCaseDep, int posLigCaseArr, int posColCaseArr){
    boolean bool[][] = new boolean[7][7]; // tableau de boolean true si la case a déjà été visité
    bool[posLigCaseDep][posColCaseDep]=true;
    int[] couple = new int[2];
    Vector<int[]> chemin = new Vector<>(); // le chemin pour aller jusque la case de fin
    int posLigCaseAct = posLigCaseDep;
    int posColCaseAct = posColCaseDep;
    couple[0]=posLigCaseDep;
    couple[1]=posColCaseDep;
    chemin.add(couple); // première case du chemin
    while (chemin.isEmpty()==false){
        if (bool[posLigCaseArr][posColCaseArr]==true){ // break si la case d'arrivée est atteinte
            break;
        }
        else if (passageEntreCases(posLigCaseAct, posColCaseAct, posLigCaseAct-1, posColCaseAct) && bool[posLigCaseAct-1][posColCaseAct]==false){
            // case 1 en bas de case 2
            bool[posLigCaseAct-1][posColCaseAct]=true;
            posLigCaseAct = posLigCaseAct-1;
            couple = new int[2];
            couple[0]=posLigCaseAct;
            couple[1]=posColCaseAct;
            chemin.add(couple);
        }
        else if (passageEntreCases(posLigCaseAct, posColCaseAct, posLigCaseAct, posColCaseAct-1) && bool[posLigCaseAct][posColCaseAct-1]==false){
            // case 1 à droite de case 2
            bool[posLigCaseAct][posColCaseAct-1]=true;
            posColCaseAct = posColCaseAct-1;
            couple = new int[2];
            couple[0]=posLigCaseAct;
            couple[1]=posColCaseAct;
            chemin.add(couple);
        }
        else if (passageEntreCases(posLigCaseAct, posColCaseAct, posLigCaseAct+1, posColCaseAct) && bool[posLigCaseAct+1][posColCaseAct]==false){
            // case 1 en haut de case 2
            bool[posLigCaseAct+1][posColCaseAct]=true;
            posLigCaseAct = posLigCaseAct+1;
            couple = new int[2];
            couple[0]=posLigCaseAct;
            couple[1]=posColCaseAct;
        }
    }
}
```


Annexe 5 : Capture d'écran du git



Annexe 6 : Capture d'écran du discord



Annexe 6 : Capture d'écran des échanges de mail

