Document de Conception du jeu Pacman_VYBA

Fait par :

AFKIR Mohammed
BOUGABA Mohamed
LIFSAL Younes
VINCENT Paul
YAHYAOUI Mehdi

***Promotion** : 2023*
***Groupe :** VYBA*
***Encadré par** : M. Hoai Diem Phuc Ngo*

Listes des fonctionnalités du jeu :

Classe “principale” :

Méthode **“principale()”** : lance le jeu avec la demande d’entrée des données du joueur
Méthode **“niveau()”** : demande la sélection du niveau
Méthode **“deplacement_heros()”** : Créé la fenêtre du labyrinthe et va s'actualiser
Méthode “**fin_partie()”** : Fenêtre de fin de partie lorsque le joueur gagne
Méthode **“perdue_partie()”** : Fenetre lorsque le joueur perd la partie (plus de point de vie)
Méthode **“KeyPressed(KeyEvent e)** : s’active lorsqu'une touche du clavier est pressé (flèches, barre espace), récupère le code de cette commande, et permet d’effectuer une des actions associé à la touche (déplacements des monstres/fantômes, du héros, et de l’attaque). vérifie aussi les points de vie du héros à la fin de l’action, et si on est arrivé.
Méthode **“keyTyped(Keyevent e)”** : s’active lorsqu’une touche du clavier est tapée (lettres et chiffres). Permet de récupérer le numéro du niveau auquel le joueur veut jouer.
Méthode **“keyReleased(KeyEvent e)”** : n’est pas utilisé dans notre cas, mais est obligatoire pour le bon fonctionnement du code, qu’il sache qu’il n’y a rien à faire lorsqu’une touche est relâché
Méthode “**actionPerformed(ActionEvent e)”** : Associe les boutons des différentes interfaces à une action, en récupérant le texte du bouton.

Classe “Hero”:

Méthode “Hero(String name, char c, int n, int m, int a, int b)” : permet la création du héros sur un plateau, avec une position initiale(a,b), une taille de plateau(n,m), un nom et un genre
Méthode **“getName()”** : retourne le nom du héros
Méthode “**setName(String name)”** : mets à jour le nom du héros
Méthode “**getc()”** : retourne le genre du héros
Méthode “**setGender(char c)”** : met à jour le genre du héros
Méthode “**getPosition()”** : retourne la position actuelle du héros
Méthode “**MoveRight()”** : déplace le héros vers la droite si possible
Méthode “**MoveUp()”** : déplace le héros vers le haut si possible
Méthode “**MoveLeft()”** : déplace le héros vers la gauche si possible
Méthode “**MoveDown()”** : déplace le héros vers le bas si possible
Méthode “**getPointVie()”** :retourne le nombre de point de vie du héros
Méthode **“setPointVie(int pointVie)”** : mets à jour les points de vie du héros
Méthode “**attaque_hero()”** : permet au héros d'attaquer un monstre/fantôme sur la case adjacente
Méthode **“teleportation()”** : téléporte le héros vers l’autre case de téléportation du niveau
Méthode “**case_piege()”** : enlève des points de vie lorsque le héros tombe sur une case piège
Méthode “**case_magique()”** : enlève ou ajoute aléatoirement des points de vie au héros

Classe Plateau :

Méthode **Plateau():** crée une matrice de taille (n*m) représentant le plateau, avec la position initiale du héros
Méthode "**getA()"** : retourne la position du héros sur la hauteur
Méthode "**setA(int a)"** : mets à jour la position du héros sur la hauteur
Méthode **"getB()"** : retourne la position du héros sur la largeur
Méthode "**setB(int b)"** : mets à jour la position du héros sur la largeur
Méthode **"getN()"** : retourne le nombre de ligne du plateau
Méthode **"setN(int n)"** : mets à jour le nombre de ligne du plateau
Méthode "**getM()"** : retourne le nombre de colonne du plateau
Méthode **"setM(int m)"** : mets à jour le nombre de colonne du plateau
Méthode **"gettableau()"** : permet d'obtenir le tableau sous forme de matrice
Méthode **"Affichage(int[][] tableau)"** : affiche le plateau sous forme de matrice dans la console

Pour le plateau, les numéros représente :

0 : case avec rien dessus (herbe)
1 le héros
2 un mur
3 : un monstre
4 : un fantôme
5: l'arrivée
6 : un fantôme dans un mur
7: case piège
8 : case de téléportation
9 : case magique
11 : le héros sur la case de téléportation

Classe Draw :

Méthode "**Draw**()" : récupère les informations du plateau, des monstres/fantômes, du héros
Méthode "**paint**(Graphics g)" : dessine le labyrinthe, les commandes, les information sur le héros et les monstres dans la fenêtre graphique

Classe labyrinthe:

Méthode labyrinthe : permet de récupérer les emplacements des murs des différents niveau grâce aux méthode "mur1()", "mur1_2()", "mur2_1()","mur2_2()","mur3_1()" et "mur3_2()" et de les mettre dans la matrice correspondant au plateau, on initialise aussi la position des cases spéciales

Classe mur1:
Méthode mur11(): retourne une liste des coordonnées sur la hauteur des murs du niveau 1

Classe mur1_2:
Méthode mur12(): retourne une liste des coordonnées sur la largeur des murs du niveau 1

Classe mur2_1:
Méthode mur21(): retourne une liste des coordonnées sur la hauteur des murs du niveau 2

Classe mur2_2:
Méthode mur22(): retourne une liste des coordonnées sur la largeur des murs du niveau 2

Classe mur3_1:
Méthode mur31(): retourne une liste des coordonnées sur la hauteur des murs du niveau 3

Classe mur2_2:
Méthode **mur32():** retourne une liste des coordonnées sur la largeur des murs du niveau 3

Classe Monster :

Méthode **"getNbr_monster()"** : Retourne le nombre des monstres
Méthode **"setNbr_monster(int nbr_monster)"** : Mets à jour le nombre des monstres
Méthode **"getId_monster()"** : Retourne L'ID du monstre
Méthode **"setId_monster(int id_monster)"** : Mets à jour l'ID du monstre
Méthode **"getPoint_de_vie()"** : Retourne le nombre de points de vie du monstre
Méthode **"setPoint_de_vie(int point_de_vie)** : Mets à jour les points de vie du monstre
Méthode **"getPosition_monster()"** : retourne la position actuelle du Monstre
Méthode **"MoveRightMon()"** : déplace le monstre vers la droite si possible
Méthode **"MoveUpMon()"** : déplace le monstre vers le haut si possible
Méthode **"MoveLeftMon()"** : déplace le monstre vers la gauche si possible
Méthode **"MoveDownMon():** déplace le monstre vers la gauche si possible

Classe Fantome

Méthode **"getNbr_fantome()"** : Retourne le nombre des fantômes
Méthode **"setNbr_fantome(int nbr_fantome)"** : Mets à jour le nombre des fantômes
Méthode **"getId_fantome()"** : Retourne L'ID du fantôme
Méthode **"setId_fantome(int id_fantome)"** : Mets à jour l'ID du fantôme
Méthode **"getPointVie()"** : Retourne le nombre de points de vie du fantôme
Méthode **"setPointVie(int pointVie)** : Mets à jour les points de vie du fantôme
Méthode **"getPosition()"** : retourne la position actuelle du fantôme
Méthode "**MoveRightFan()"** : déplace le fantôme vers la droite si possible
Méthode **"MoveUpFan()"** : déplace le fantôme vers le haut si possible
Méthode **"MoveLeftFan()"** : déplace le fantôme vers la gauche si possible
Méthode **"MoveDownFan():** déplace le fantôme vers la gauche si possible

Diagrammes de classes et de séquence:

Sprint 1:

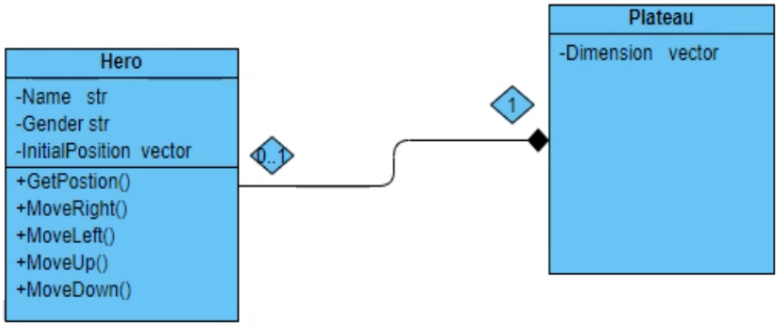

Figure1 :Diagramme de classe

Sprint 2 :

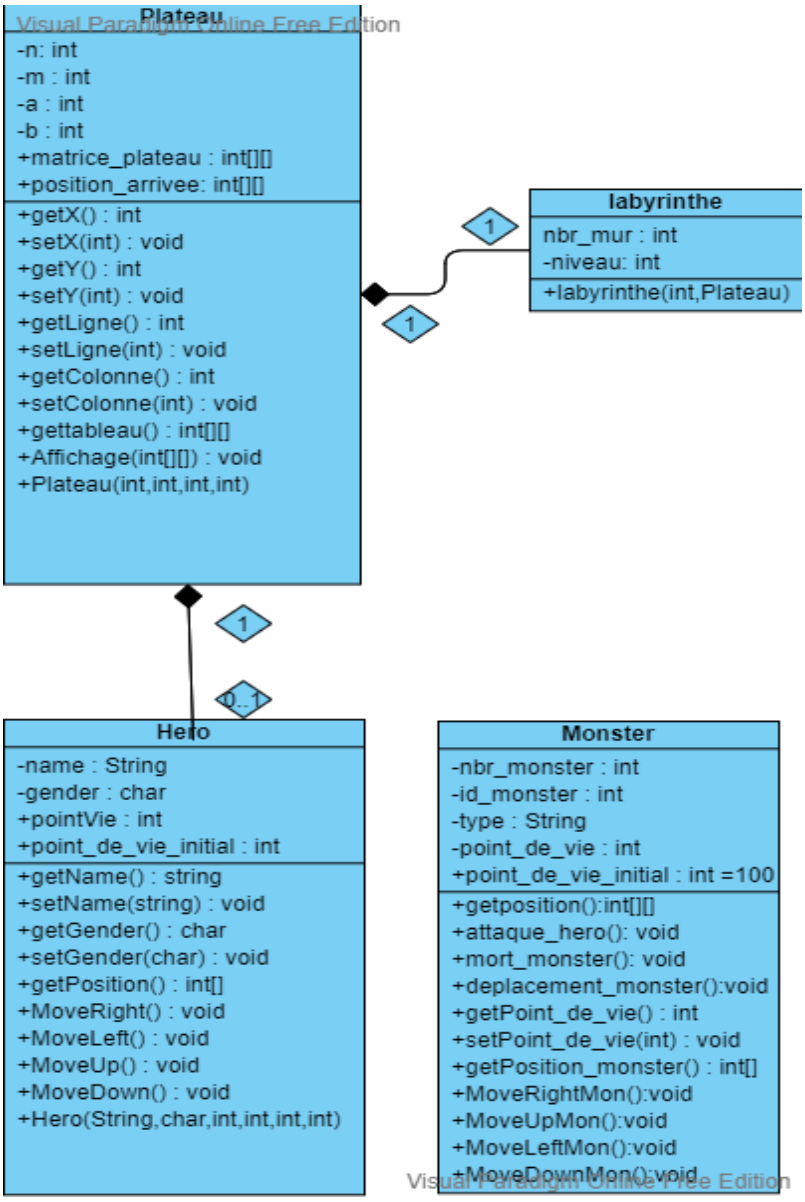

Figure2 : Diagramme de classe

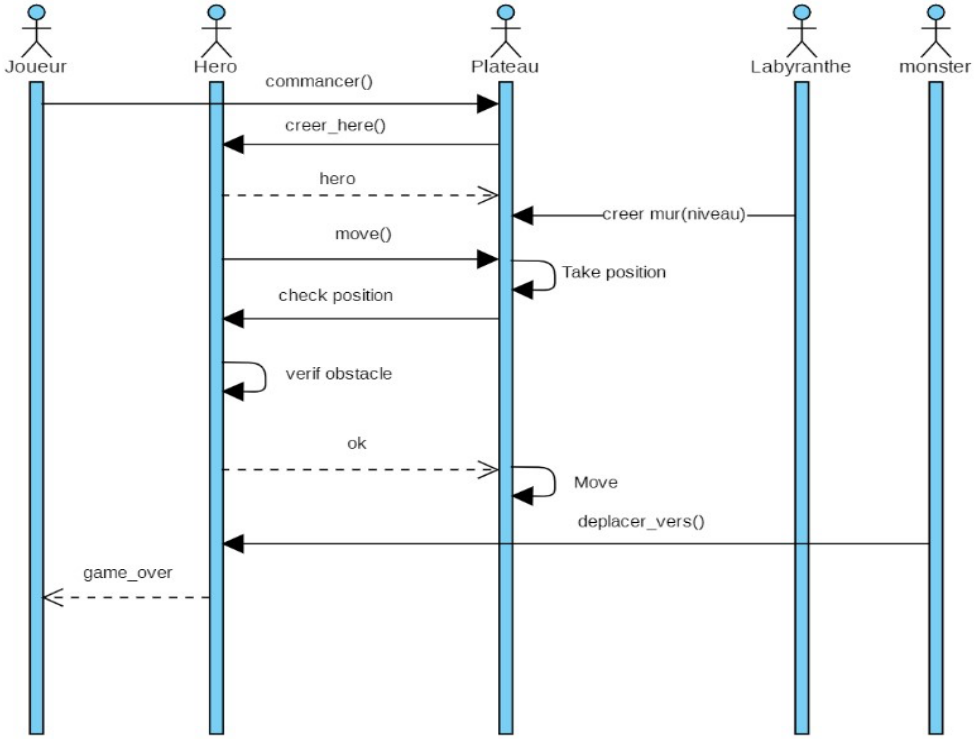

Figure3 :Diagramme de séquence

Sprint 3 :

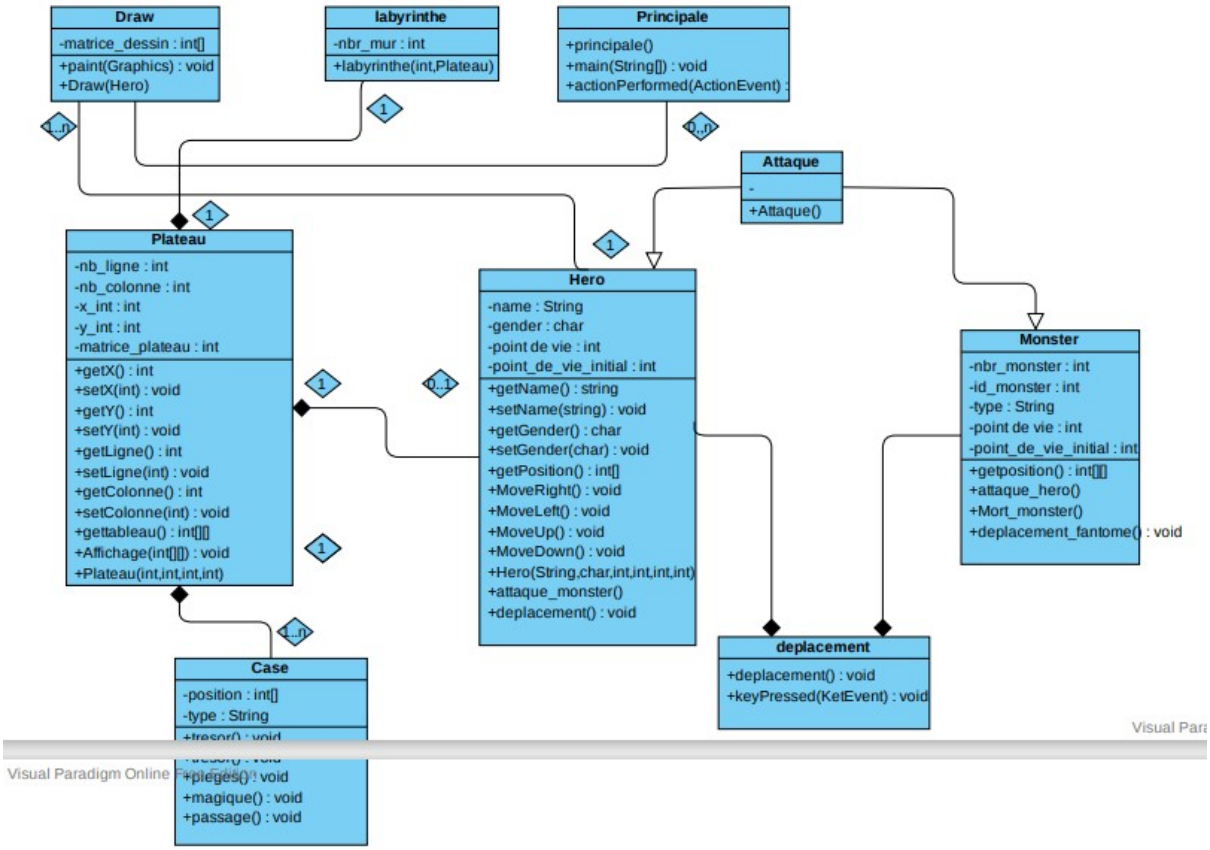

Figure4 :Diagramme de classe

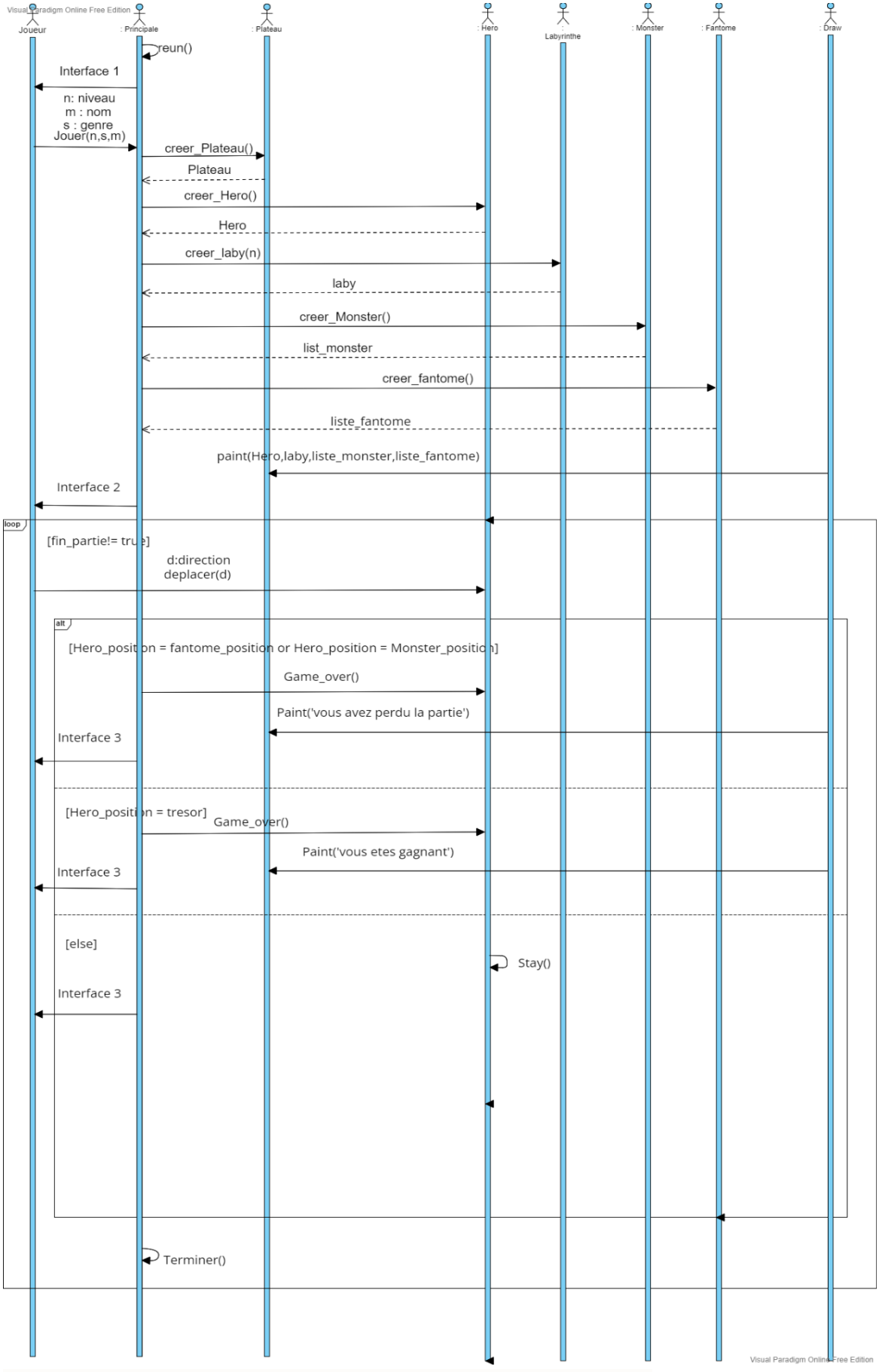

Figure5 : Diagramme de séquence

Sprint 4 diagrammes finaux:

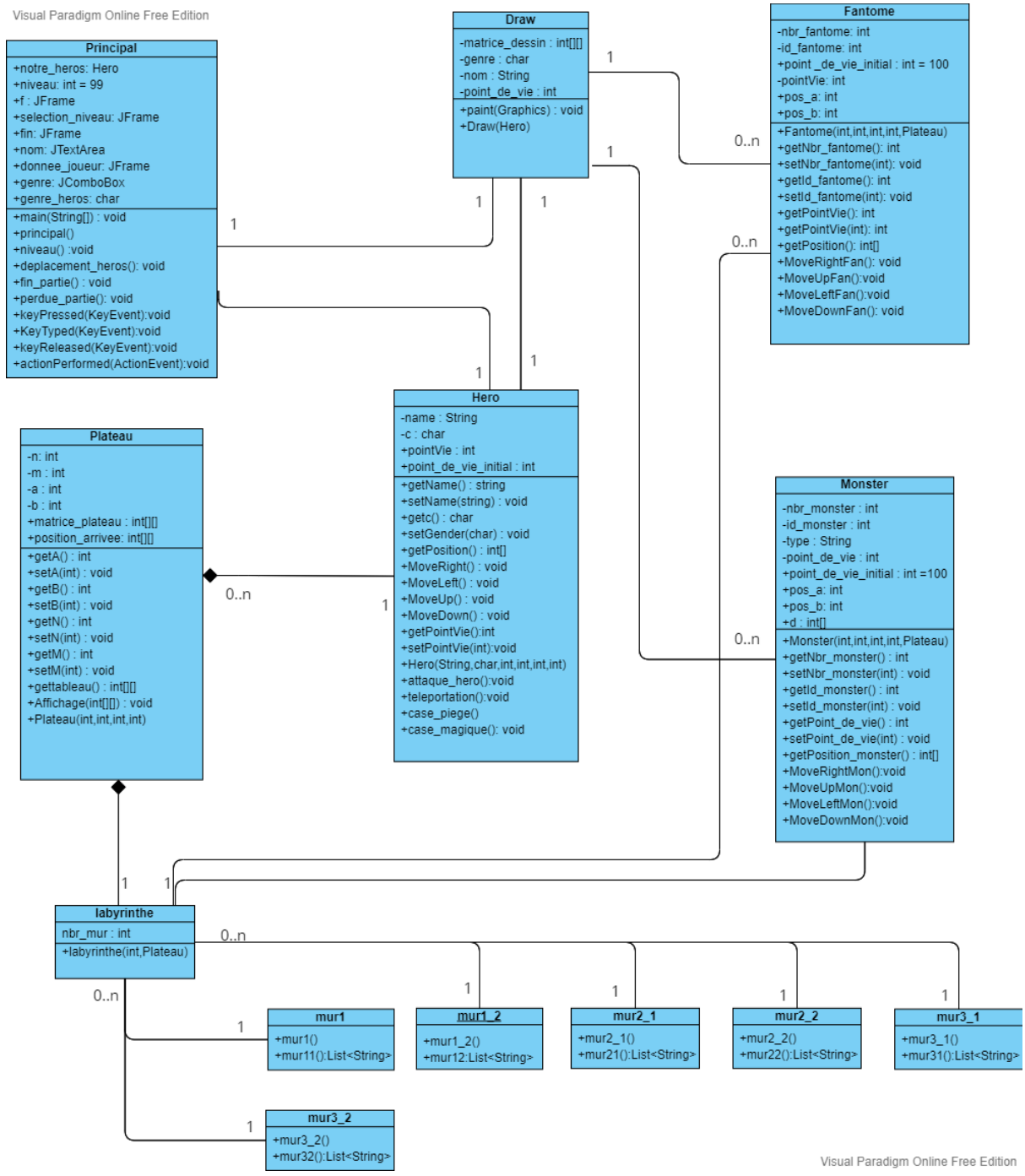

Figure6 : Diagramme de classe

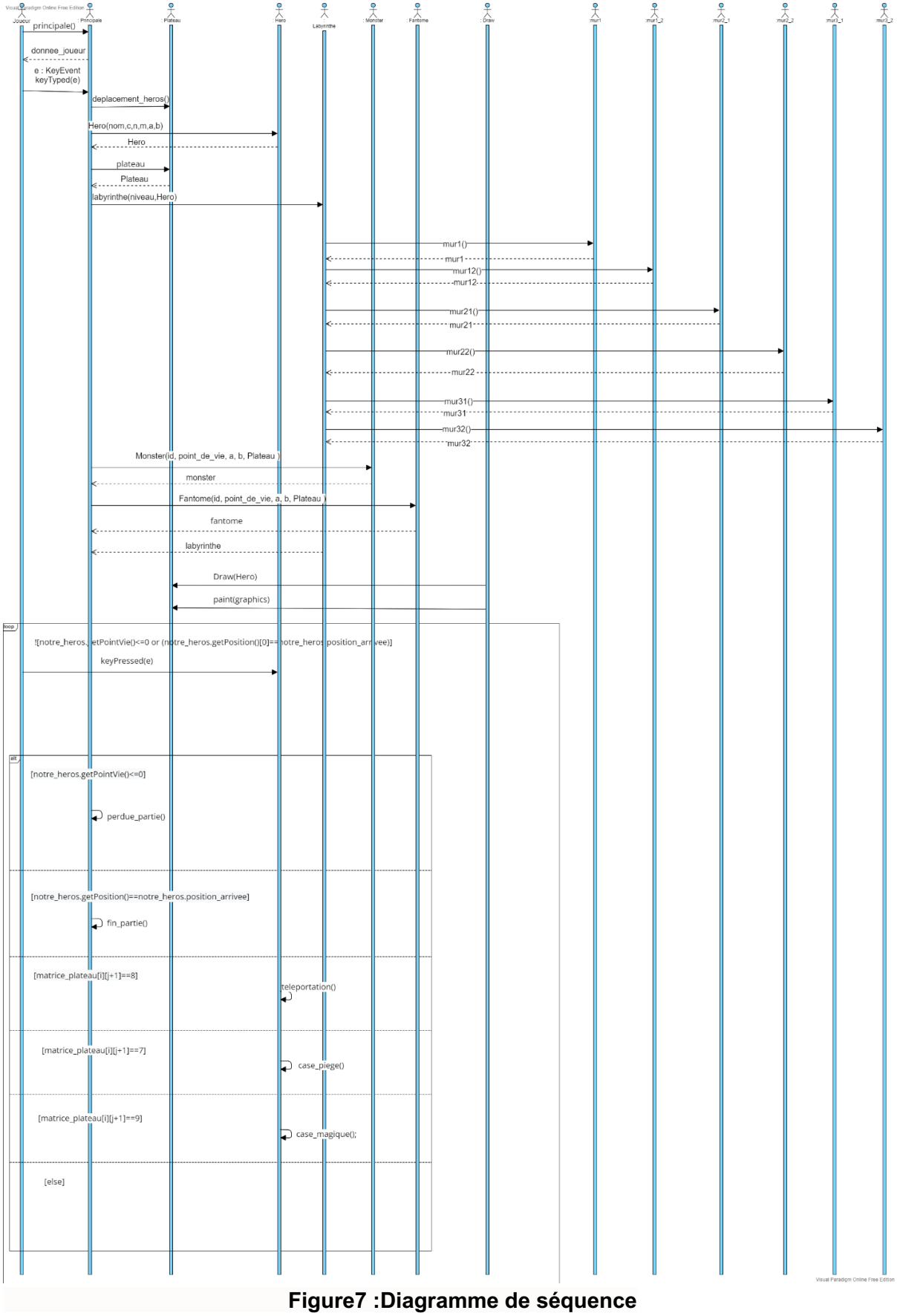

Figure7 :Diagramme de séquence

L’organisation du projet :

Répartition des Tâches entre les membres de l'équipe:

Après avoir créé le projet sur GitHub et faire une réunion entre les membres de l'équipe, nous nous sommes mis d'accord que sur le long du projet Paul Vincent va s'occuper de tout ce qui est relatif à l'implémentation de l'interface graphique, Mohamed Bougaba et Mehdi Yahyaoui vont s'occuper de créer tous les classes et implémenter les codes nécessaires et à partir du sprint3, Bougaba va s'occuper de tester les codes implémenter par Mehdi Yahyaoui. En ce qui concerne les diagrammes de classe et diagramme de séquence, Mohamed Afkir va s'en occuper.

Sprint1 :

Tout d'abord, afkir va créer le diagramme de classe.
Pour ce sprint, Mohamed Bougaba va commencer par créer la classe plateau, pour cela, il va créer une matrice de taille(n*m). Par la suite, Mehdi Yahyaoui va s'occuper de la création de la classe Héro et Mohammed Afkir va créer les méthodes MoveUp,Movedown, MoveRight et MoveLeft qui va permettre à notre héros de se déplacer dans les quatre directions et aussi la méthode GetPostion qui nous permettra de récupérer la positon du héros à chaque instant. Ensuite, Paul Vincent va créer l'interface graphique grâce à la bibliothèque Swing, cette interface contiendra le héros qui se déplace librement dans le plateau.

Sprint2 :

Tout d'abord, afkir va actualiser le diagramme de classe et élaborer le diagramme de séquence
Pour commencer, Mohamed Bougaba va mettre à jour la classe Héro et créer la classe fantôme qui contient les méthodes MoveUp,Movedown, MoveRight et MoveLeft qui va lui permettre de se déplacer dans les quatre directions et aussi la méthode GetPostion qui nous permettra de récupérer la positon du fantôme à chaque instant. Ce dernier contrairement à l'héros et le monstre est capable de franchir les murs. Tandis que Mohammed Afkir va créer la classe monstre de la même façon que la classe fantôme et Mehdi Yahyaoui va s’occuper de la classe Labyrinthe qui contient les 3 niveaux, le niveau1 va contenir 50 murs, le niveau2 va contenir 100 murs et le niveau3 150 murs. Enfin, Paul Vincent va afficher les monstres et les fantômes sur l'interface graphique et va aussi créer une page d'accueil qui permettra au joueur de saisir son nom et de choisir le niveau souhaité.

Sprint3 :

Après les remarques faites par la tutrice, Mohamed afkir va améliorer le diagramme de séquence, et Mehdi Yahyaoui va générer le labyrinthe à partir des fichiers texte. Il va aussi créer la méthode attaque qui permet au Héro d'attaquer les monstres et les fantômes. Comme Mehdi va s'occuper des codes, Bougaba va faire les tests Junit qui vérifiera que le héros, les monstres et les fantômes sont créés, se déplacent dans les quatre direction et ne sortent pas du plateau. En sus, Paul Vincent va mettre des images et les icônes du labyrinthe, du monstre, du fantôme, et du Héros ainsi que leur points de vie qui s'actualise au cours du jeu. Puis, il va

implémenter la méthode attaque faite par Mehdi Yahyaoui dans l'interface graphique qu'on pourra l'utiliser en appuyant sur la barre d'espace.

Sprint4 :

Pour le sprint final, Afkir va mettre la version finale des diagrammes de classe et de séquence. Mehdi va ajouter les cases pièges et téléportation et magique. Mohamed Bougaba va faire le test JUnit des cases ajouter par Mehdi, ainsi que la méthode attaque. Paul va finaliser à son tour l'interface graphique, en ajoutant les cases sur l'interface et en améliorant quelques bugs restants.