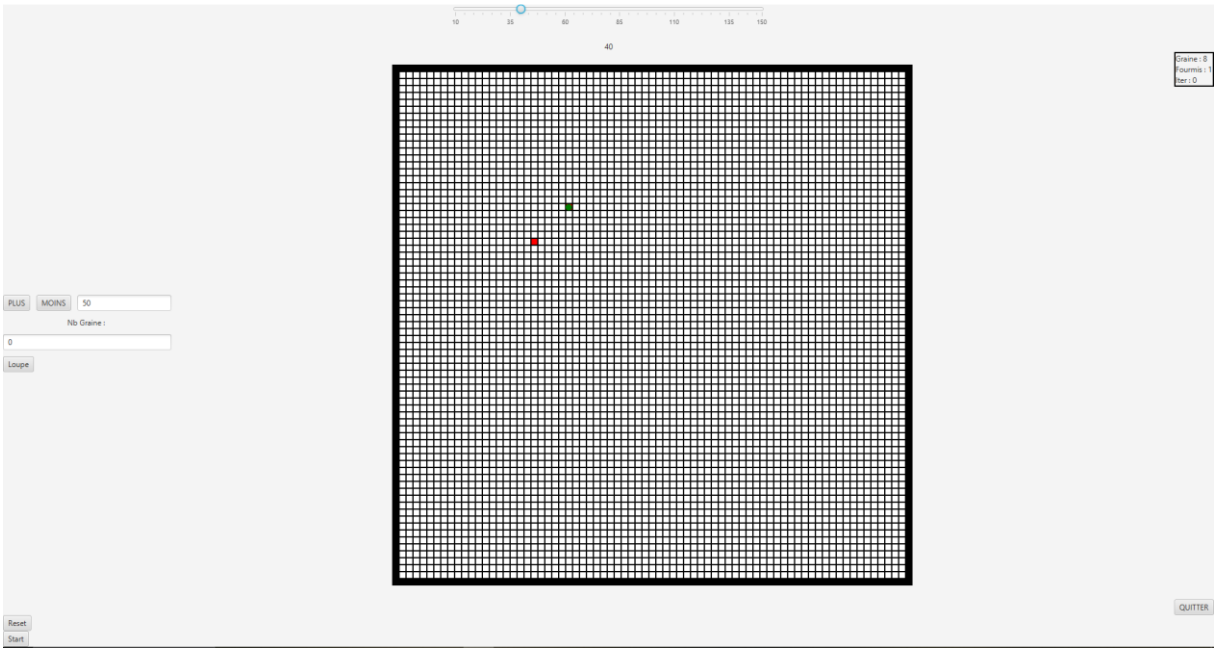
Arthur Haye

Baptiste Lacroix

L3 info – Projet

POO-IMH 2



SEMESTRE 6 2022/2023

Résumé :

Ce projet consiste à développer une interface de simulation de fourmilière, dans laquelle des fourmis peuvent ramasser ou éparpiller des graines sur une grille de cellules carrées représentant le sol de la fourmilière. Chaque cellule peut contenir un mur, une fourmi ou un tas de graines (qMax). Les fourmis peuvent porter une seule graine et se déplacent aléatoirement sur les cellules voisines en prenant ou en laissant des graines selon certaines conditions.

L'interface devra comporter un plateau de jeu initial de 50x50 cellules représenté par des carrés de 10 pixels, ainsi qu'une grille de paramètres permettant de modifier la taille de la grille, la vitesse de la simulation et les probabilités d'apparition de graines, de fourmis et de murs. La simulation peut être mise en pause ou en lecture avec le bouton "Play/Pause" et il est possible d'ouvrir une fenêtre zoom de 11x11 cellules avec le bouton "Loupe". La simulation peut être réinitialisée avec le bouton correspondant et les composants affichent le nombre de graines et de fourmis dans la fourmilière ainsi que le nombre d'itérations depuis le début de la simulation.

Structure du code :

Le code de notre application est divisé dans 3 packages. Le premier est le packages contenant la classe JeuDesFourmi.java qui permet de lancer l'application.

Le second est le package correspondant au modèle dans notre MVC. Il contient les données principales de l'application ainsi que les méthodes pour les manipuler. Dans notre cas nous avons les classes Fourmi.java et Foumilière.java où les noms sont explicites concernant les données contenues par chacune.

Le dernier correspond donc à la vue du MVC. Ce package regroupe tous les composants nécessaires au bon fonctionnement de notre application.

Explication des Classes :

JeuDesFourmis.java : Cette classe est une classe dépendant de la Classe Application et possède un unique attribut qui est une Interface que nous avons définie. Cette classe possède une méthode Start pour pouvoir lancer notre application.

Foumi.java : Cette classe comme son nom l'indique représente les fourmis de notre application. Elle contient les méthodes permettant à la fourmi d'effectuer les actions citées précédemment.

Fourmilère.java : Cette représente notre fourmilière. Elle est construite à partir d'une taille que nous pourront modifier ainsi qu'un nombre de graine et un nombre de graine maximum par case. Elle contient également la liste des fourmis présentes. Les bordures de la fourmilière sont stockées dans une matrice de coordonnées. Il en est de même pour les fourmis et les farines.

ButtonLoupe.java : Cette classe définit un bouton personnalisé appelé "ButtonLoupe" qui hérite de la classe de base "Button". Ce bouton est utilisé pour zoomer sur une image du "Plateau", qui est une classe de type "GridPane".

Le bouton "ButtonLoupe" contient une référence à la classe "Plateau" via la variable "gridPane". Lorsque le bouton est cliqué, la méthode "handleZoom()" est appelée.

Dans cette méthode, un "ImageView" est créé à partir d'une image de la "GridPane". Ensuite, un "StackPane" est créé pour contenir l'image zoomée, qui est créée en appliquant une échelle à ce "StackPane". Une "ScrollPane" est également créée pour permettre le défilement de la zone de zoom.

Un "Slider" est ajouté pour régler l'échelle du zoomPane, et un "VBox" est créé pour contenir le "Slider" et la "ScrollPane". Finalement, une nouvelle fenêtre est créée pour afficher la "VBox" contenant le "Slider" et la "ScrollPane", qui affichent tous deux l'image zoomée.

Enfin, la variable "firstClick" est réinitialisée à "true" pour permettre un nouveau clic sur le bouton si nécessaire.

ButtonReset.java : Cette classe définit un ButtonReset qui étend la classe Button. Lorsqu'un objet ButtonReset est créé, il prend un objet Plateau en argument. Ce bouton est associé à un événement de clic, qui déclenche l'exécution de la méthode Reset() de l'objet Plateau correspondant. Cette méthode Reset() peut être utilisée pour réinitialiser l'état du plateau. Lorsque le bouton est cliqué, la méthode Reset() est exécutée.

ButtonStart.java : La classe ButtonStart est une extension de la classe Button. Elle crée un bouton qui permet de démarrer ou de mettre en pause une tâche associée à un objet Plateau. Le bouton utilise une propriété booléenne qui indique si la tâche est en pause ou non, et permet de modifier l'état de cette propriété en fonction de l'état de la tâche. Si la tâche est en cours d'exécution, le bouton change l'état de la propriété booléenne et modifie le texte du bouton pour afficher "Pause" ou "Resume". Si la tâche n'est pas en cours d'exécution, le bouton lance une nouvelle tâche et modifie le texte du bouton pour afficher "Pause".

InfoFourmi.java : La classe InfoFourmi hérite de la classe VBox et contient trois labels : nbGraine, nbFourmi, et nbIter, qui sont des compteurs de graines, de fourmis et d'itérations, respectivement. Les valeurs de ces compteurs sont liées aux propriétés correspondantes : grainesProperty, fourmisProperty et iterProperty. Ces propriétés sont initialisées avec les valeurs actuelles de la grille et sont mises à jour via la méthode refresh_Info.

Interface.java : Cette classe est juste l'interface graphique de notre application.

ModifParam.java : Cette classe permet grâce à différents composant comme des sliders ou des boutons de modifier les paramètres de l'application tels que la taille, la vitesse de simulation ainsi que le nombre de graines.

Plateau.java : Il s'agit d'un code écrit en Java pour simuler une fourmilière. La classe "Plateau" hérite de la classe "GridPane" et définit plusieurs variables pour la taille du plateau, la taille de la cellule, le nombre d'itérations, etc. Elle contient également des variables pour la fourmilière, telles que la quantité de graines, les murs et les fourmis.

Le constructeur de la classe initialise la taille du plateau, crée un tableau de rectangles pour représenter la grille, initialise la fourmilière, ajoute une fourmi et des graines à des positions spécifiques et définit une tâche pour l'évolution de la fourmilière.

La classe a également des méthodes getter et setter pour les variables définies, telles que la largeur, la hauteur, le tableau de rectangles, le tableau de cercles, etc. Elle définit également une méthode pour initialiser la fourmilière et une méthode pour initialiser le plateau en fonction de la fourmilière. Enfin, elle a une méthode pour réinitialiser les tableaux de rectangles et de cercles et une méthode pour récupérer la fourmilière.

QuitBtn.java : Comme son nom l'indique c'est un bouton qui quand on le presse ferme l'application.

Simulation.java : Il s'agit d'une classe qui étend la classe "Task". Elle a une variable de membre appelée "p" de type "Plateau", une variable de membre "EtatPlateau" de type "Boolean", et elle prend en argument un objet "Plateau" lors de sa construction.

La classe "Simulation" a une méthode "call()" qui lance une boucle infinie dans laquelle elle met en pause le thread en cours d'exécution pendant p.speed millisecondes, puis appelle les méthodes "evolue()" et "Refresh()" sur l'objet "Fourmilier" contenu dans "p" si "EtatPlateau" est vrai. Enfin, la méthode "call()" retourne "null" lorsqu'elle est terminée.

SliderVitesse.java : Cette classe "SliderVitesse" étend la classe VBox et est utilisée pour créer un Slider et un Label affichant la valeur actuelle du Slider.

La valeur par défaut du Slider est de 40 et la plage de valeurs possibles va de 10 à 150. Le Slider a également des étiquettes de graduation, un curseur majeur tous les 25 points et un curseur mineur tous les 5 points. La taille du Slider est de 500 x 50 pixels et le Label est centré horizontalement.

Un Listener est ajouté pour mettre à jour la valeur du Label chaque fois que la valeur du Slider est modifiée.

Problèmes et difficultés :

Durant du projet nous avons rencontré certaines difficultés plus ou moins difficiles à surmonter. Ces dernières ont quasiment toutes été traitées, cependant il reste malgré tout encore quelque problème. Nous ne pouvons pas ajouter plus d'une seule graine par case ou en retirer, nous sommes obligés d'arrêter le Thread manuellement. Il faut également plusieurs clics sur le bouton start pour désactiver les composants modifiants les paramètres.

Conclusion :

En conclusion nous avons une application fonctionnelle qui comporte quelques bugs en liant toutes les fonctionnalités entre elles, cependant ces fonctionnalités prise une à une fonctionnent correctement.

Concernant la gestion et la réalisation du projet dans notre groupe, nous avons travaillé principalement ensemble et en parallèle en salle de TP avec bien sûr quelques heures en plus chez nous.