Dossier Projet

SASSADY Vanpasone

Architecture :

Le jeu est composé de plusieurs packages, model, view, controller, une mainApp, et d'un document ressources pour les images.

Le jeu possède une classe ***Carte*** qui créée l'ensemble des pays (une liste) infectés et sains en lisant dans un fichier où la plupart des attributs des pays y sont. Cette classe permet aussi d'avoir les informations sur la liste de pays qui va détecter les couleurs de la carte et les lier avec les pays correspondant.

Il y aussi une classe ***Epidemie*** avec des caractéristique pouvant varier pour la vitesse de propagation, la létalité.

La classe ***Jeu***, quant à elle, va créer la carte et l'épidémie, c'est aussi elle qui va être responsable de lancer le jeu grâce à la GameLoop, les fonctions pour infecter les Pays et se propager se fait directement dans la classe, les fonctions sont ensuite ajouter dans la gameLoop afin de propager le virus en temps réel.

Il y a la classe ***LecteurPays*** qui va permettre, grâce aux fichiers de liaisons de pays, et de liste de pays, de créer les listes à partir de fichiers lus, on utilisera, un FileReader et un Bufferreader pour cela.

La classe ***Monde*** va s'occuper de mettre à jour les données du jeu grâce à des property, elle représente tous les habitants du monde, les gens sains, infectés et morts. C'est dans cette classe que la fonction mettre à jour les données sur le monde a été faite, afin de mettre à jour les valeurs dans le jeu.

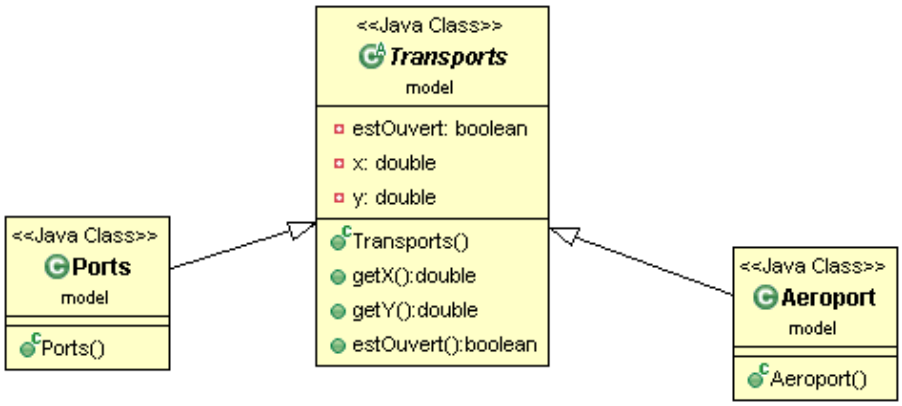
La classe ***Pays*** a beaucoup d'attributs, la richesse, le milieu, le climat, la température, une liste d'aéroports, de pays voisins, et une population.

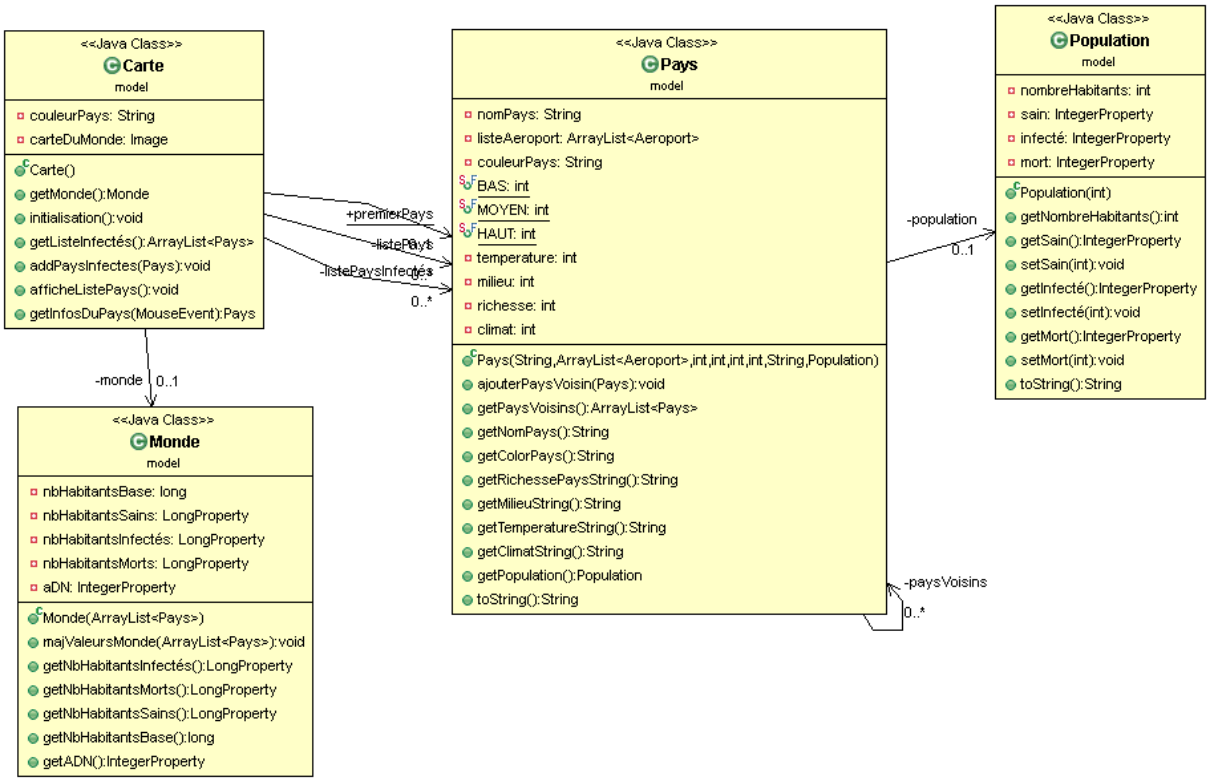
La classe ***Transports*** est une classe abstraite qui contient les coordonnées des ports et des aéroports. Elle permet aux classes Ports et Aéroports d'hériter des fonctions pour récupérer les coordonnées. Il y a en effet deux types de Transports, par voie maritime ou aérienne, c'est pour cela qu'on ne peut pas instancier Transports.

Le controller ***MainController*** permet de gérer les évènements que ce soit pour lancer le jeu, cliquer pour avoir les informations du pays, ou encore créer des objets que l'on mettra dans Scenebuilder avec le fichier fxml.

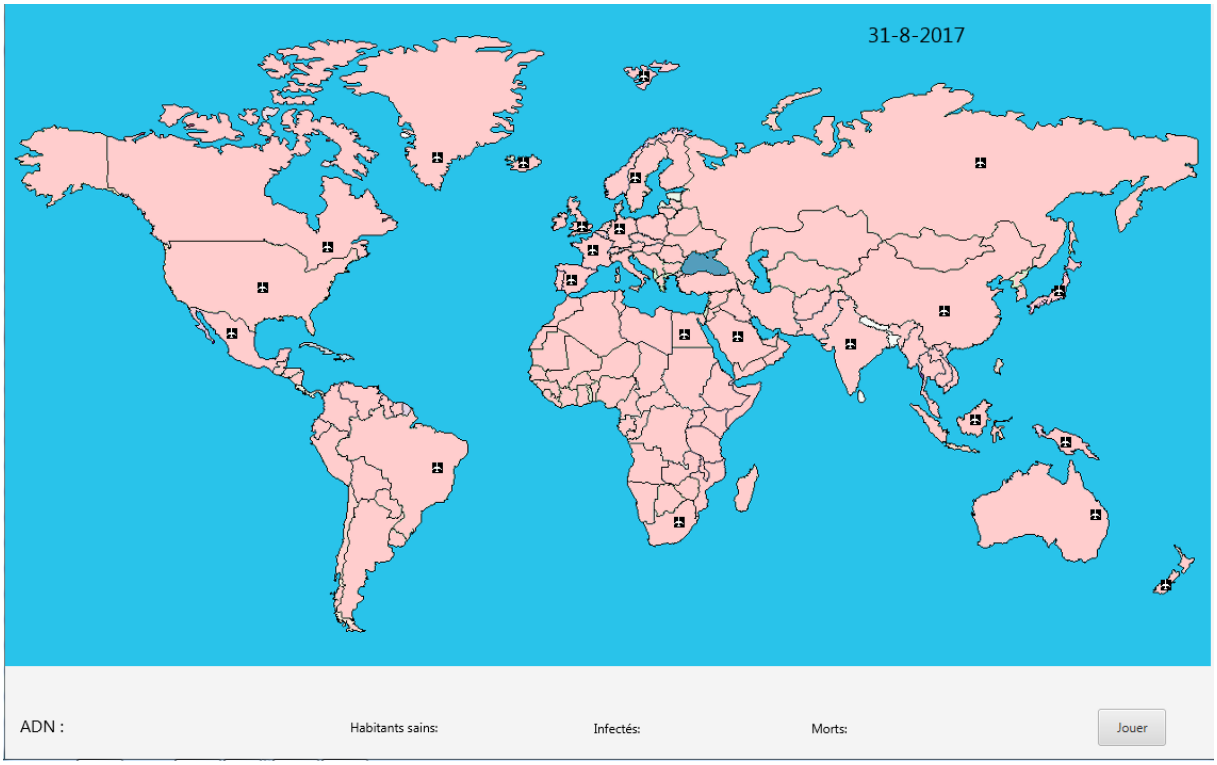
La view est ce que l'on va voir afficher à l'écran, c'est-à-dire la carte du monde avec ses pays, ses aéroports, et les informations des pays et du Monde en dessous de la carte, cette carte sera lancer dans la classe PlagueApp lorsque l'on va lancer l'application et la fenêtre avec le fichier fxml.

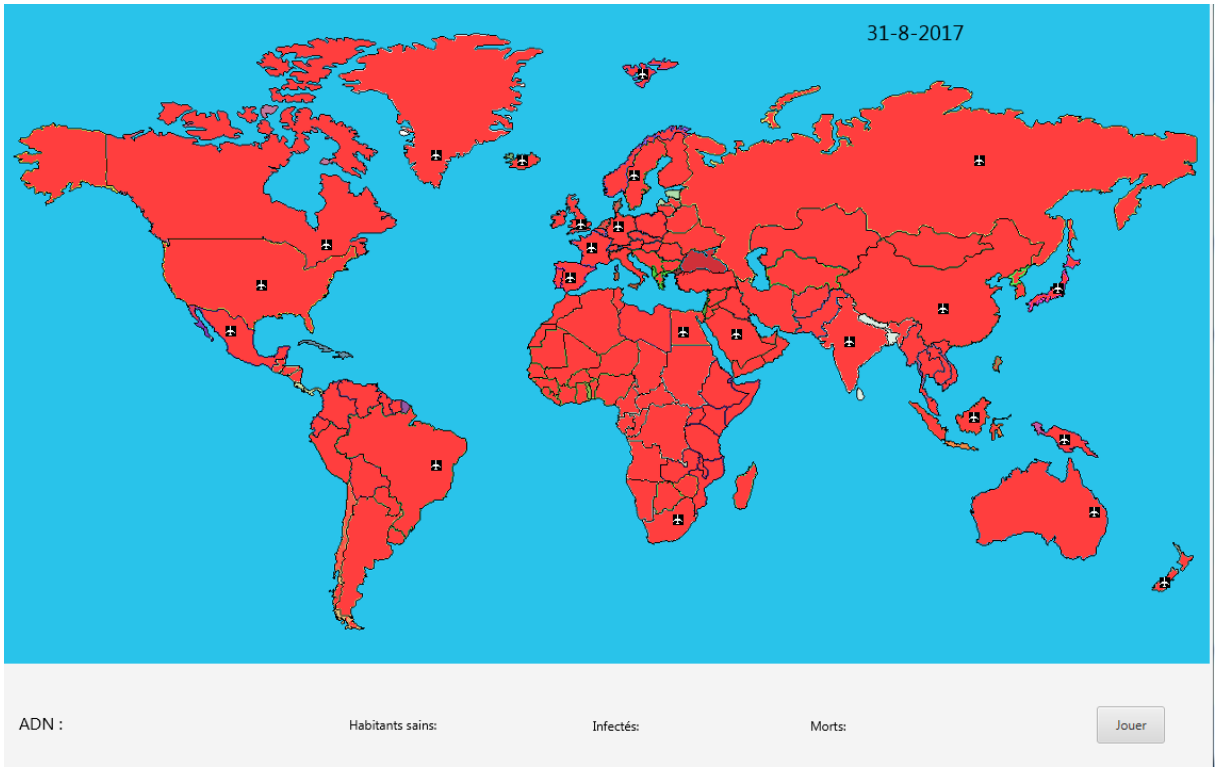
Voici plusieurs diagrammes de classe qui illustrent l'héritage :


<<Java Class>>
Transports
model
estOuvert: boolean
x: double
y: double
Transports()
getX():double
getY():double
estOuvert():boolean
<<Java Class>>
Ports
model
Ports()
<<Java Class>>
Aeroport
model
Aeroport()


<<Java Class>>
Carte
model
couleurPays: String
carteDuMonde: Image
Carte()
getMonde():Monde
initialisation():void
getListeInfectés():ArrayList<Pays>
addPaysInfectes(Pays):void
afficheListePays():void
getInfosDuPays(MouseEvent):Pays
+premierPays
-listePays
-listePaysInfectés
0..*
-monde
0..1
<<Java Class>>
Monde
model
nbHabitantsBase: long
nbHabitantsSains: LongProperty
nbHabitantsInfectés: LongProperty
nbHabitantsMorts: LongProperty
aDN: IntegerProperty
Monde(ArrayList<Pays>)
majValeursMonde(ArrayList<Pays>):void
getNbHabitantsInfectés():LongProperty
getNbHabitantsMorts():LongProperty
getNbHabitantsSains():LongProperty
getNbHabitantsBase():long
getADN():IntegerProperty
<<Java Class>>
Pays
model
nomPays: String
listeAeroport: ArrayList<Aeroport>
couleurPays: String
BAS: int
MOYEN: int
HAUT: int
temperature: int
milieu: int
richesse: int
climat: int
Pays(String,ArrayList<Aeroport>,int,int,int,int,String,Population)
ajouterPaysVoisin(Pays):void
getPaysVoisins():ArrayList<Pays>
getNomPays():String
getColorPays():String
getRichessePaysString():String
getMilieuString():String
getTemperatureString():String
getClimatString():String
getPopulation():Population
toString():String
-population
0..1
-paysVoisins
0..*
<<Java Class>>
Population
model
nombreHabitants: int
sain: IntegerProperty
infecté: IntegerProperty
mort: IntegerProperty
Population(int)
getNombreHabitants():int
getSain():IntegerProperty
setSain(int):void
getInfecté():IntegerProperty
setInfecté(int):void
getMort():IntegerProperty
setMort(int):void
toString():String

Voici la maquette utilisée à l'aide de SceneBuilder. La carte est dans un AnchorPane, et est composée de deux cartes, une colorée en premier plan qui est opaque et ne sert juste qu'à reconnaître les couleurs de la carte pour lier les pays à l'image et l'autre, blanche en second plan pour gérer les évènements qui vont faire apparaître des animations sur cette carte, colorée ou non. Il y a un Pane en dessous de la carte qui contient un bouton pour lancer le jeu lorsque le pays a été sélectionné, et des informations sur les populations des pays et du monde, il y a également une zone de texte pour l'affichage des caractéristiques des pays qui n'ont pas besoin d'être mis à jour, comme le nom, la richesse, le milieu, la température, et le climat.





Les cartes ci-dessus montrent l'évolution de l'infection en fonction du nombre de personnes infectées dans le monde. En effet plus il y a de personnes infectées dans le monde, plus la couleur rouge devient foncée.

Spécifications fonctionnelles :

Dans la classe ***Carte*** la méthode Initialisation () où nous créons les pays et les liaisons des pays, nous utilisons une méthode de la classe ***LecturePays*** qui permet de lire les fichiers. On fait une boucle for et on utilise une liste de Pays afin de lire chaque ligne du fichier et ajouter à liste chaque ligne du fichier, contenant les attributs du pays. Il y aussi la méthode getInfosDuPays(MouseEvent e) qui permet de d'utiliser pixelReader pour reconnaître les couleurs de la carte du monde en couleur et d'associer chaque couleur de pays à un pays (grâce au pixel lu pendant le clic de la souris).

Dans la classe ***Jeu*** la méthode lancerJeu(Pays pays) instancie la gameLoop et permet de faire tourner le jeu indéfiniment. Plusieurs méthodes sont appelées à chaque tour de boucle pour mettre à jour le jeu concernant le nombre d'infectés, la couleur de carte et les points d'ADN gagnés.
La méthode infecterPays(Pays pays) initialise le nombre d'infectés à 2, et est appelé chaque fois qu'un nouveau pays est appelée à chaque fois dans propagation() pour infecter un pays voisin.
La méthode propagation() permet de propager le virus à travers un pays, selon un algorithme qui rajoute aléatoirement entre 0 et 30% de la population infectée à chaque tour de boucle. Elle a 60% de chance d'infecter la population d'un pays voisin si le pays déjà infecté a plus de 50% de sa population qui est infectée.
Et la méthode changement() permet d'augmenter l'opacité de la carte rouge en fonction du nombre d'infectés dans le monde.

Dans la classe ***Monde***, il y a des integerProperty pour permettre l'affichage des changements de valeurs des infectés, des personnes saines dans le monde. Il y aussi la méthode majValeursMonde(ArrayList<Pays> listePaysInfectés) qui permet de mettre à jour ces valeurs dans la vue, cette méthode est appelée dans la fonction lancerJeu() à chaque tour de boucle pour mettre à jour les valeurs.

Dans la classe ***MainController***, les méthodes sont utiliser pour gérer les évènements, comme par exemple getInfosPaysController, qui gère le clic de la souris en utilisant la fonction dans carte pour afficher (grâce à un bind) les informations d'un pays lors du clic.

Et la fonction lancerJeu() dans le controller qui gère le clic de l'infection du premier pays, en effet, lorsqu'un pays est sélectionné, et qu'ensuite on appuie sur le bouton jouer, la première infection commence.

Diagramme de séquence du lancement du jeu jusqu'à la fin de la propagation :

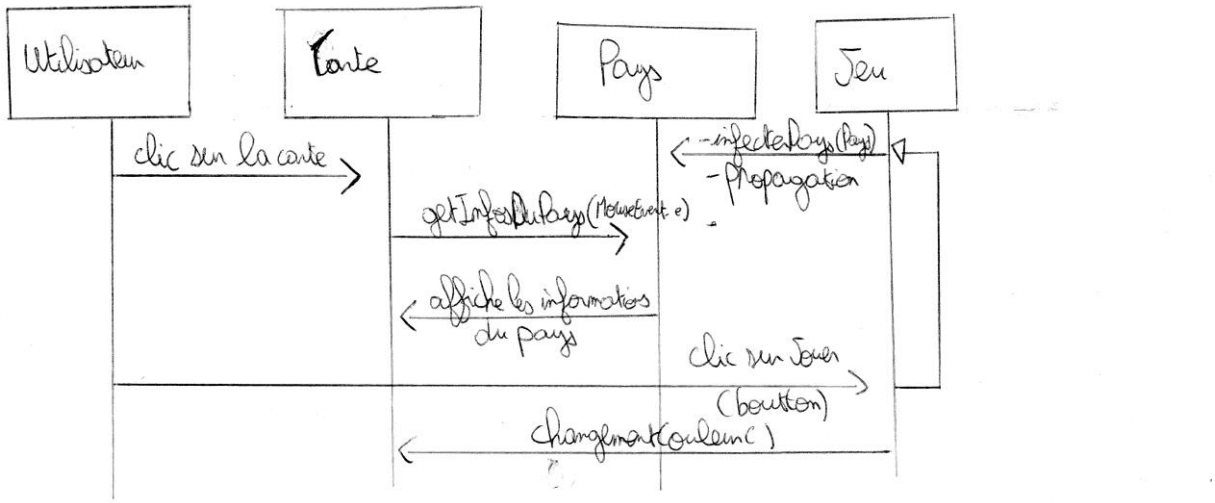


Diagramme de séquence pour l'utilisateur jusqu'à la propagation du virus

UML :

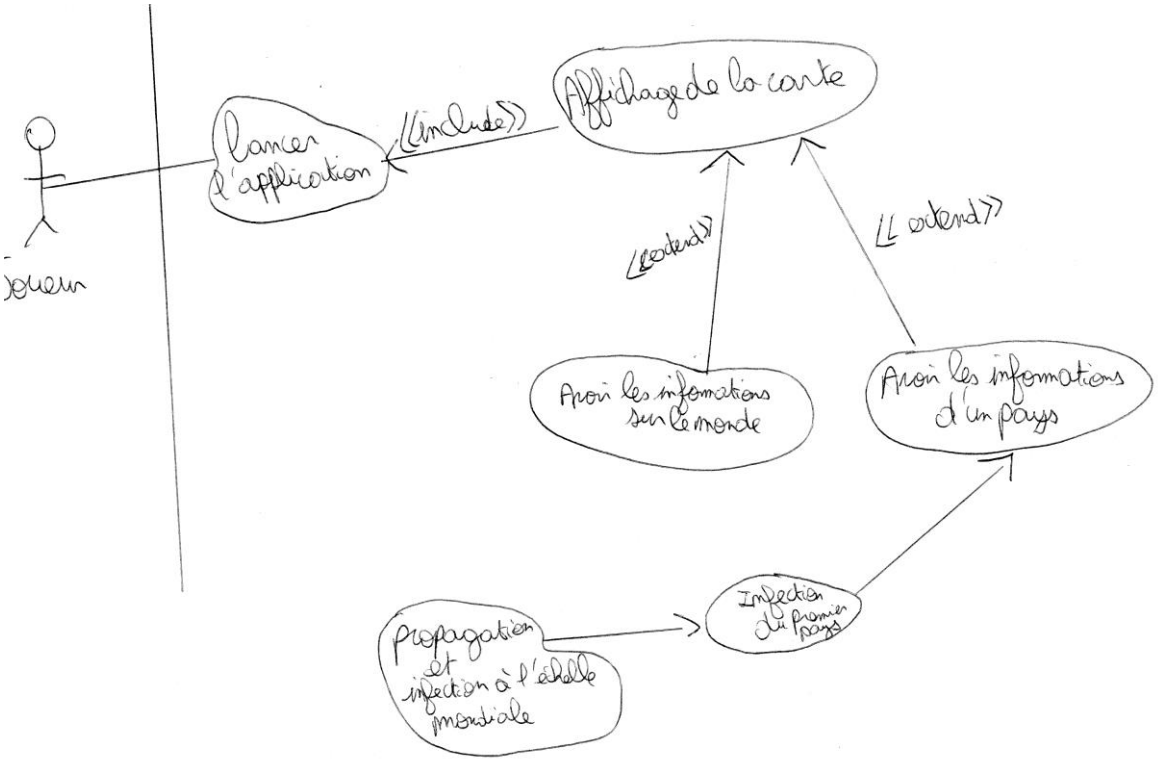


Diagramme cas d'utilisation UML