Rapport du projet java : jeu de la vie

Le jeu de la vie est un jeu qui se base sur deux règles, la survie et la naissance, à partir de ses règles on pourra avoir des résultats incroyables.

Le projet était de réaliser ce jeu en utilisant le langage de programmation java.

Le jeu utilise des listes chaînés que nous avons implémenté pour représenter les cellules vivantes, chaque étapes de jeu (nouvelle génération) on génère une nouvelle liste à partir de l'ancienne, l'évolution du jeu peut nous mener a plusieurs résultats, mais le projet consiste a considéré que quatre types d'évolutions qui sont :

Mort : la liste est vide (aucune cellule vivante).

Stable : la liste ne change plus (pas d'évolution).

Oscillateur : le jeu a une période, on retrouve la même liste après cette période.

Vaisseau : le jeu a une liste qu'on retrouve après un intervalle régulier mais déplacé d'un nombre de colonnes et de lignes.

A part ses cas, on considère que le type est indéterminé, donc on stop l'évolution.

La répartition du projet était partagé ainsi, Allouche Mohamed Abdelkarim a fait les deux classes Cellule, ListeTest et le rapport PDF. Hamroun Mohamed Houssem-eddine a réalisé les trois classes Jeux_Circulaire, Jeux_Fini, LecteurLif. Chatti Les deux classes Comportement et Main était faite par Oussama Chatti et les deux classe Maillon et Liste par Chebbah Antar. La dernière classe qui reste est la classe main était partagé entre nous quatre parce qu'elle est la classe la plus importante pour le projet et ayant plusieurs méthodes.

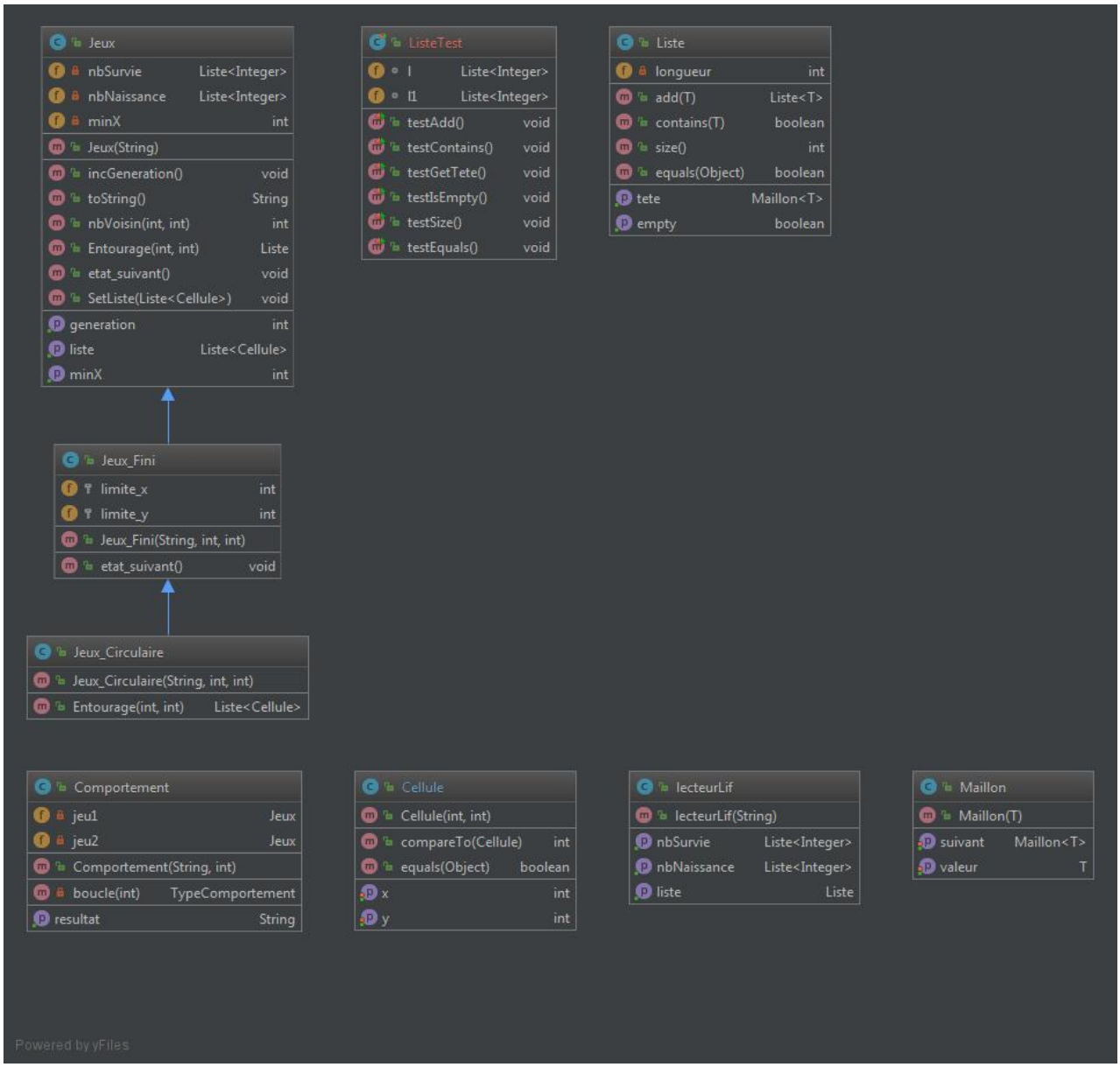
Pour les choix des algorithmes, on a suivi les consignes de notre prof de td sur quelques méthodes pour rester dans le contexte du projet,par exemple l'utilisation et l'implémentation des listes par nous-mêmes et d'autre méthodes. Expliquant les méthodes les plus importantes pour la réalisation du projet.

La classe Liste, Add(T e1) si l'élément e1 existe déjà elle ne l'ajoute pas sinon elle l'ajoute dans la bonne place en utilisant comparTo ,et la méthode equals(Object o) retourne true si l'élément o est égale à la liste sinon false.

La classe Jeux, getMinX() récupère le plus petit x (l'abscisse de la cellule) dans notre liste pour l'utiliser après dans la méthode toString() , cette dernière vas commencer par cet abscisse et parcourir les ligne (y l'ordonné) et afficher ses cellule, ensuite la méthode nbVoisin (int x,int y) , parcours toute la liste et compare l'abscisse et l'ordonné de la cellule avec les autres cellules si la distance est égale à 1 on incrémente le nombre de voisins, après la méthode Entourage(int x , int y), renvoi une liste des cellule fabriquer par la méthode add (T e1) et l'entourage de x et y, en utilisant ses deux méthodes, on a fait la méthode etat_suivant() parcours toute la liste de nos cellules et qui à partir de nbVoisin décide du sort de la cellule et prends la liste de l'entourage de cette cellule et fait la même chose pour les cellules de cette liste.

La classe Comportement, la méthode boucle (int nMaxRep) détermine le type du jeu qu'on a déjà mentionné précédemment et le constructeur renvoi le type et les caractéristiques du jeu.

La classe Jeux_Fini hérite de Jeux et la classe Jeux_Circulaire hérite de Jeux_Fini ,de petits changements sont rapportés a la méthode etat_suivant () et entourage respectivement dans les deux classes.


Jeux
nbSurvie Liste<Integer>
nbNaissance Liste<Integer>
minX int
Jeux(String)
incGeneration() void
toString() String
nbVoisin(int, int) int
Entourage(int, int) Liste
etat_suivant() void
SetListe(Liste<Cellule>) void
generation int
liste Liste<Cellule>
minX int
ListeTest
l Liste<Integer>
l1 Liste<Integer>
testAdd() void
testContains() void
testGetTete() void
testIsEmpty() void
testSize() void
testEquals() void
Liste
longueur int
add(T) Liste<T>
contains(T) boolean
size() int
equals(Object) boolean
tete Maillon<T>
empty boolean
Jeux_Fini
limite_x int
limite_y int
Jeux_Fini(String, int, int)
etat_suivant() void
Jeux_Circulaire
Jeux_Circulaire(String, int, int)
Entourage(int, int) Liste<Cellule>
Comportement
jeu1 Jeux
jeu2 Jeux
Comportement(String, int)
boucle(int) TypeComportement
resultat String
Cellule
Cellule(int, int)
compareTo(Cellule) int
equals(Object) boolean
x int
y int
lecteurLif
lecteurLif(String)
nbSurvie Liste<Integer>
nbNaissance Liste<Integer>
liste Liste
Maillon
Maillon(T)
suivant Maillon<T>
valeur T
Powered by yFiles