isep
École d'ingénieurs du numérique

ISEP 2016 -2017

# JAVA – Pacman

Nicolas Benmennad, Antoine Poussot

# I. Rappel du Sujet

## 1. Origine du Pacman

Le sujet de java de cette année est un Pacman, un jeu d'arcade très célèbre créé en 1980 pour la société Namco. La notoriété de ce jeu est telle que l'on le retrouve encore aujourd'hui et tout isépien connait bien évidement le personnage et son image de camembert jaune.

Le but du jeu est simple : collecter l'ensemble des pac-gommes répartis dans l'ensemble du labyrinthe. Seulement, celui-ci est aussi hanté par 4 fantômes qui cherchent, eux, à manger Pacman. Ce dernier n'est pas sans défense pour autant, il a la possibilité de manger des super pac-gommes qui lui permettent ensuite de dévorer nos adorables fantômes… Ces power-ups sont au nombre de 4 et se situent dans les quatre coins du labyrinthe

Le joueur commence avec 3 vies, à chaque contact non protégé avec les fantômes, il en perd une et gagne le bonus MST. Le déplacement du Pacman s'effectue avec les touches fléchées uniquement, la direction choisie est non bloquante, ce qui veut dire que le Pacman s'arrêtera uniquement s'il rencontre un mur.

Le comptage du score repose sur un système simple, +10 pour chaque pac-gomme dévorée, +100 pour une super pac-gomme. D'autres objets peuvent apparaitre aléatoirement dans le labyrinthe qui apportent chacun +500 au score et éventuellement une vie supplémentaire au joueur.

## 2. Cahier des charges

Pour ce projet, nous avons eu un cahier des charges très large : Réaliser un Pacman. Néanmoins, nous avons créé un certain nombre d'objectifs plus précis pour nous aider dans la réalisation. Certains de ces objectifs proviennent du sujet et nous avons complété avec les nôtres.

- Un joueur humain doit pouvoir y jouer en mode graphique,
- Le jeu doit pouvoir être joué en version 2 joueurs,

Concernant le déroulement d'une partie :

- Le jeu se déroule dans un labyrinthe prédéfini,
- Le Pacman doit se déplacer à l'aide des touches du clavier,
    - Les entrées doivent être non-bloquantes pour le reste du programme
    - 
- Les fantômes sont au nombre de 4 et doivent se déplacer de manière aléatoire.
    - Une détection de collision est impérative pour ne pas rentrer dans les murs.
    - Si possible, les fantômes doivent traquer le (les) Pacman(s).
- Le Pacman récupère les PacGommes et son score s'incrémente en accord.
- Une détection des collisions Pacman-fantôme doit être fonctionnelle
    - Si le Pacman est invincible, il mange le fantôme et celui-ci revient à sa position de départ. Le score est incrémenté en conséquence.

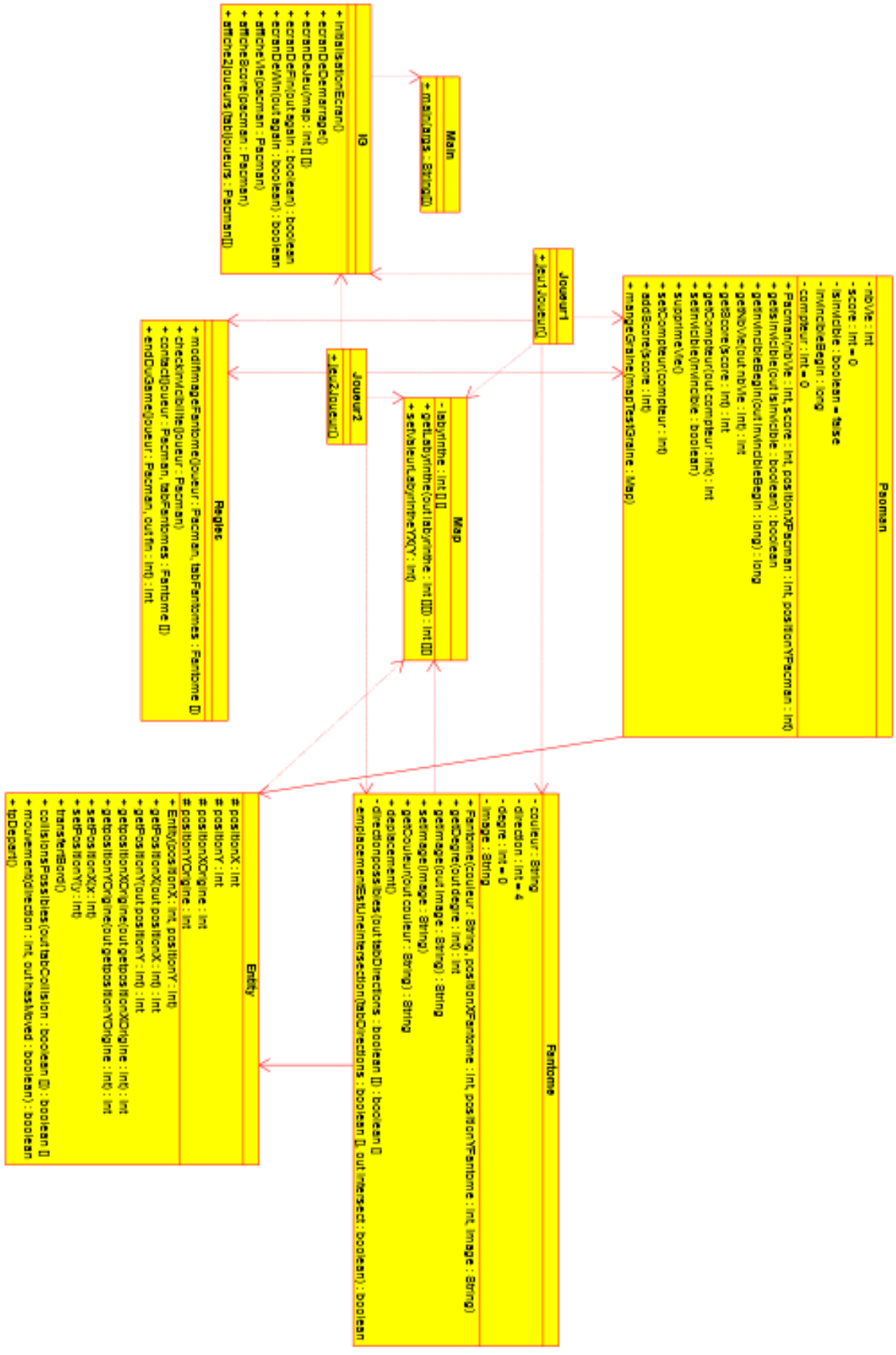
    - Sinon, le Pacman revient à sa position initiale, les fantômes également et le joueur perd une vie.
- Un suivi du score et des vies des Pacman doit être affiché sur l'écran
- Des zones de téléportations doivent être implémentés sur les deux extrêmes du labyrinthe

Concernant l'interface graphique :

- Un écran de menu principal
    - Comporte un bouton cliquable « 1 joueur »
    - Comporte un bouton cliquable « 2 joueurs »
    - Comporte un bouton cliquable « contrôles »
    - Affiche une image titre
- Un écran de jeu, modifié en fonction du mode de jeu choisi.
    - L'écran de jeu affichera le score et le nombre de vie du (des) joueur(s)
    - L'écran de jeu comporte le labyrinthe sur lequel se déplacent les entités affichées
- Deux écrans de fin
    - Affichent le score du joueur
    - Comportent un bouton cliquable « replay », si cliqué ce bouton renvoie au menu
    - Un écran en cas de victoire
    - Un écran en cas de défaite

# II. Modélisation du jeu

## 1. Diagramme UML



## 2. Classes du programme

Le programme Pacman que nous avons codé repose sur une structure de code orientée objet par opposition aux méthodes procédurales. Ainsi, le code repose sur une répartition des méthodes par classes. Ces classes sont ensuite instanciées et nous utilisons les méthodes des instances en fonction des besoins dans le déroulement du jeu. Cette philosophie permet une économie non négligeable de code pour les structures qui se répètent, dans notre cas, les fantômes et un deuxième Pacman dans le mode à deux joueurs.

### 0x1 – Main

La classe Main est le point d'entrée de la machine virtuelle Java dans le lancement du programme, c'est de cette classe que nous allons lancer le reste du programme. A cet effet, nous devons importer les librairies StdDraw et l'ensemble du package Java.awt.

Le programme tournera ensuite dans une boucle infinie qui ne sera cassée que si l'utilisateur décide de quitter le jeu. Cette boucle nous permet d'assurer le redémarrage du jeu lorsque les boucles de jeux s'arrêtent en cas de défaite ou de victoire.

La main instancie un objet de la classe IG et appelle les méthodes *initialisationEcran()* et *ecranDeDemarrage().* Ces méthodes seront développées plus loin.

### 0x2 – Joueur1

Cette classe comporte l'ensemble de l'algorithme nécessaire au bon déroulement du jeu dans sa version 1 joueur. Elle comporte une unique fonction : *jeu().* C'est cette fonction que nous devons appeler pour lancer la partie

La fonction instancie ensuite tous les objets nécessaires au déroulement de la boucle qui s'ensuit, soit 4 fantômes Fantôme, 1 Pacman Pacman. Elle instancie également un objet Map qui contient le labyrinthe et un objet reglesDuJeu qui contient les méthodes implémentant les règles de Pacman.

Une fois les instanciations effectuées, Java rentre dans la boucle qui fera fonctionner le jeu. La boucle commence par vérifier si le jeu est fini. Ensuite elle vérifie si une touche a été pressée, si oui la variable direction change en conséquence (8,6,2,4 comme sur un pavé numérique), le degré de représentation de l'image également.
Le labyrinthe est ensuite dessiné en mémoire par la méthode ecranDeJeu de IG
Une vérification de l'invincibilité du Pacman est fait et les images des fantômes sont changés en conséquence.
Le mouvement du Pacman est ensuite effectué suivant la direction qui a été donnée après qu'un touche ait été pressée (si aucune n'est pressée, la direction actuelle continue). On vérifie s'il est nécessaire de téléporter le Pacman et enfin s'il y a une graine. Les méthodes utilisées proviennent de la classe Pacman et sont : mouvement(dir), transfertBord() et mangeGraine(Map).
Vient enfin la vérification de collision avec les fantômes par Regle.contact(Pacman, Fantome). A partir de là, le jeu effectue le déplacement et la vérification de TP pour chaque fantôme par la méthode Fantome.deplacement(). Enfin on revérifie les collisions avec Pacman.

Ensuite il faut dessiner ces entités sur le labyrinthe, ce qui est effectué avec la méthode StdDraw.picture() enfin, on affiche le résultat à l'écran et la boucle peut recommencer

Il est possible de sortir de la boucle lorsque la partie est perdue ou gagnée et en choisissant le bouton replay sur l'écran de fin.

### 0x3 – Joueur2

La classse joueur2 est globalement identique à la classe joueur1 dans le fonctionnement, cependant l'ajout d'un second Pacman contrôlé par un humain modifie certaines méthodes qui prennent en entrée un objet de type Pacman
Il s'agit principalement des méthodes de la classe Règle, qui sont donc doublées pour prendre en compte le second Pacman
D'autres méthodes sont aussi adaptées, comme par exemple l'affichage des vies et du score qui sont inédites à ce mode de jeu.

### 0x4 – Entity

Entity est notre classe mère pour les classes Pacman et Fantome, en effet on retrouve beaucoup d'attributs et de méthodes communs à ces deux classes, il était donc logique de les rassembler en une seule et d'ensuite faire jouer le système d'héritage de Java.

Entity comporte les attributs suivants ainsi que leurs *Setters* et *Getters* :

- PositionX
- PositionY
- PositionYOrigine
- PositionXOrigine

Dans les méthodes à retenir, nous avons :

- transfertBord() qui permet à chaque entité de jeu d'être téléportée si elle atteint l'un des deux bout du tunnel
- collisionPossibles() qui est une méthode indiquant dans un tableau de booléens les collisions possibles aux positions directement accessibles au prochain tour de boucle par l'entité.
- Mouvement(direction) effectue le mouvement en vérifiant les collisions : le mouvement sera effectué si possible, sinon rien ne se passera. On retourne un booléen indiquant si il y a eu mouvement.
- TpDepart() renvoie l'entité à sa position d'origine

### 0x5 – Pacman

Pacman hérite des méthodes mais aussi des attributs de Entitty grâce a son constructeur auquel nous avons ajouté un super() qui fait, lui, appel au constructeur de la classe mère. En sus des attributs d'Entity, Pacman nécessite les attributs suivants :

- NbVie
- Score, initié à zéro par défaut
- isInvicible
- InvicibleBegin, long qui marque le temps depuis lequel le Pacman est invincible
- Compteur, initié à zéro par défaut

Le constructeur initie les attributs vie et positions du Pacman instancié.
La méthode mangeGraine de Pacman permet de manger les graines, mettre à jour le labyrinthe et d'incrémenter le score. Elle prend en entrée le labyrinthe.

### 0x6 – Fantôme

Dans cette classe, il est nécessaire d'importer Java.Math.Random().

Fantome hérite des méthodes mais aussi des attributs de Entitty grâce a son constructeur auquel nous avons ajouté un super() qui fait, lui, appel au constructeur de la classe mère. En sus des attributs d'Entity, Fantome intègre les attributs suivants :

- Couleur, indiquant la couleur du fantome
- Sa direction, gérée en interne dans la classe
- Degre, aussi en interne dans la classe
- Image, utilisée pour representer le fantome dans la GUI

Le constructeur initie tous ces attribues en plus d'un super pour la positionX et positionY.
Le mouvement d'un fantome est régi par deplacement() cette méthode appelle à son tour d'autres méthodes pour déterminer si la position actuelle est une intersection, si des directions sont possibles et enfin génère un nombre aléatoire qui indiquera une direction et déclenchera si possible le mouvement.

### 0x7 – Règles

La classe Règle est maitresse dans le bon fonctionnement du jeu : c'est elle qui assure la détection de collision entre fantômes et Pacman(s).

Elle comporte les méthodes suivantes

- modifImageFantome(Pacman, Fantôme) qui modifie l'image des fantômes en fonction de l'invincibilité de Pacman
- checkInvincibilite(Pacman) permet de timer l'invincibilité de Pacman, actuellement fixée à 5 secondes
- Contact(Pacman, Fantôme) indique une collision entre un fantôme et Pacman
- endDuGame( Pacman)

### 0x8 – IG (Interface Graphique)

La classe interface graphique comporte toutes les méthodes ayant rapport aux écrans de jeux :

- initialisationEcran
- ecranDeDemarrage
- ecranDeJeu
- ecranDeFin
- ecranDeWin
- afficheVie
- AfficheScore
- AfficheScore2joueurs

Toutes ces méthodes sont appelées dans leurs écrans respectifs au fur et à mesure du jeu.

# III. Annexes

## 1. GitHub

https://github.com/tonito659/Pacman

## 2. Diagramme UML

https://drive.google.com/file/d/0BziUB-mapwzeMWFxcGJGTmRvbDQ/view?usp=sharing

## 3. Chaton