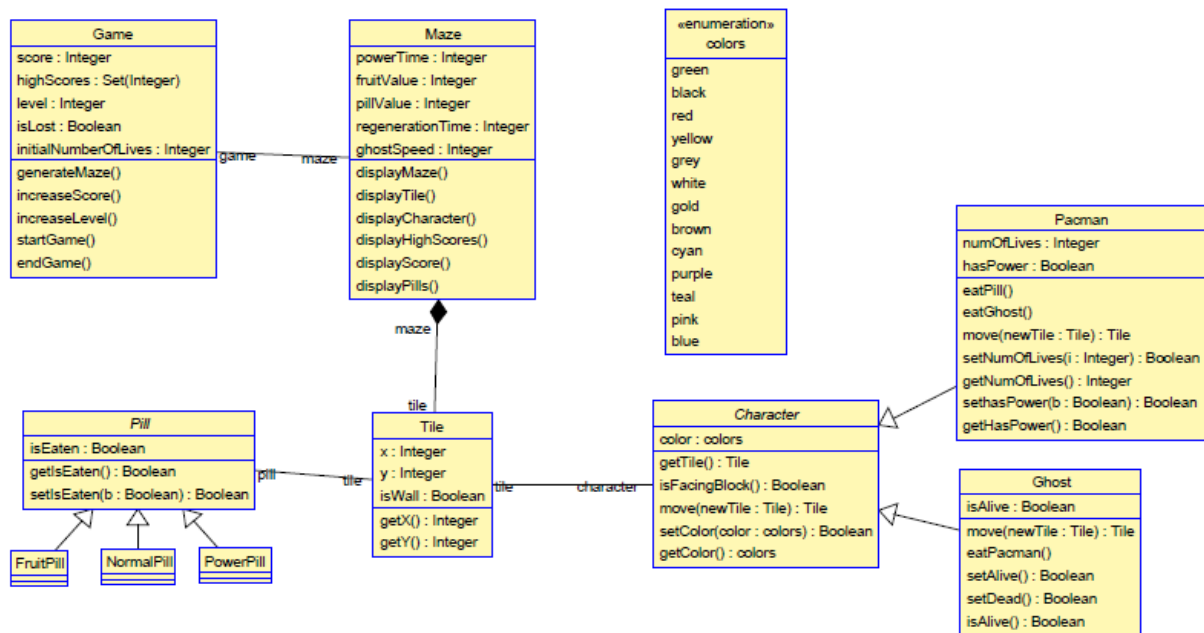
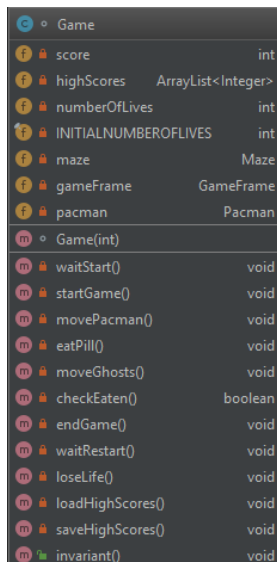


Modélisation initialement prévue



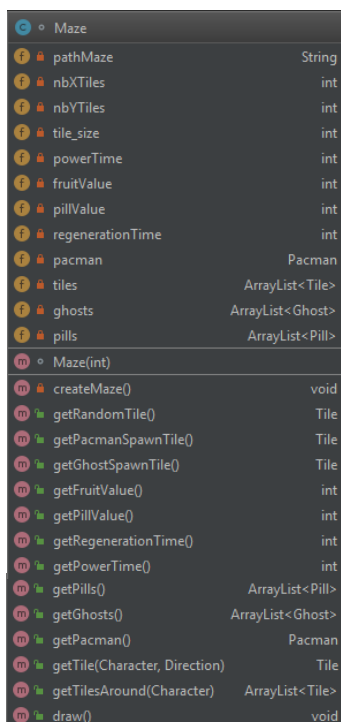
Différences par rapport à la prévision



Game	
score	int
highScores	ArrayList<Integer>
numberOfLives	int
INITIALNUMBEROFLIVES	int
maze	Maze
gameFrame	GameFrame
pacman	Pacman
Game(int)	
waitStart()	void
startGame()	void
movePacman()	void
eatPill()	void
moveGhosts()	void
checkEaten()	boolean
endGame()	void
waitRestart()	void
loseLife()	void
loadHighScores()	void
saveHighScores()	void
invariant()	void

Les méthodes `eatPacman()`, `eatGhost()` ou encore `eatPills()` qui étaient censées être dans les classes `Pacman` et `Ghost` sont finalement dans la classe `Game`, puisque ces méthodes nécessitent des données de tout le plateau, or les classes `Pacman` et `Ghost` n'ont pas accès aux autres cases du plateau. C'est donc dans la classe `Game` que tout est géré.

Des méthodes initialement prévues n'ont finalement pas été implémentées, comme `increaseLevel()`, puisqu'il suffit d'incrémenter l'attribut `level`. Enfin, nous avons rajouté quelques méthodes comme `waitStart()` et `waitRestart()`, qui communiquent avec la fenêtre `GameFrame`.



Maze	
pathMaze	String
nbXTiles	int
nbYTiles	int
tile_size	int
powerTime	int
fruitValue	int
pillValue	int
regenerationTime	int
pacman	Pacman
tiles	ArrayList<Tile>
ghosts	ArrayList<Ghost>
pills	ArrayList<Pill>
Maze(int)	
createMaze()	void
getRandomTile()	Tile
getPacmanSpawnTile()	Tile
getGhostSpawnTile()	Tile
getFruitValue()	int
getPillValue()	int
getRegenerationTime()	int
getPowerTime()	int
getPills()	ArrayList<Pill>
getGhosts()	ArrayList<Ghost>
getPacman()	Pacman
getTile(Character, Direction)	Tile
getTilesAround(Character)	ArrayList<Tile>
draw()	void

Nous avons rajouté, en attribut de la classe `Maze`, `Pacman` et les Fantômes, puisque différentes méthodes des classes `Maze` ou `Game` nécessitent ces objets. Par rapport à la modélisation prévue, nous avons rajouté `getTilesAround()`, permettant d'obtenir les cases disponibles autour d'un personnage. Cette méthode est notamment utilisée pour le déplacement des fantômes. Nous avons également rajouté `getTile()`, qui renvoie la `Tile` située à la direction indiquée en paramètre par rapport au personnage également donné en paramètre.

Toutes les méthodes `display()` sont regroupées dans la méthode `draw()`, qui dessine tous les éléments du plateau.

Figure	
width	int
height	int
x	int
y	int
color	Color
Figure(int, int, int, int, Color)	
Figure(int, int, int, int)	
centerImage(int, int)	void
changePosition(int, int)	void
getX()	int
getY()	int
getWidth()	int
getHeight()	int
getColor()	Color
draw()	void
invariant()	void

Afin de dessiner les éléments du jeu dans une fenêtre, nous avons créé une classe Figure, dont héritent les classes Character, Pill et Tile. Une figure est un élément qui peut être affiché à l'écran. Chaque figure a des coordonnées x et y indiquant leur position sur l'écran, et des dimensions.

Tile	
size	int
color	Color
pacmanSpawn	boolean
ghostSpawn	boolean
isWall	boolean
Tile(int, int, int, boolean)	
isGhostSpawn()	boolean
isPacmanSpawn()	boolean
setGhostSpawn()	void
setPacmanSpawn()	void
isWall()	boolean
draw()	void
getColor()	Color
getSize()	int

Nous avons rajouté des attributs pacmanSpawn et ghostSpawn, indiquant si la Tile est le point d'apparition de pacman ou des fantômes. Si ceux-ci ne sont pas définis, alors les personnages apparaissent aléatoirement sur le plateau.

La taille des cases est également un attribut de Tile.

Pill	
tile	Tile
Pill(Tile, Color, int)	
draw()	void
removeTile()	void
getTile()	Tile

L'attribut isEaten initialement prévu n'a pas été implémenté. Pour savoir si la Pill a été mangée, on vérifie si la Tile associée à la Pill est existante ou null. Lorsque Pacman mange une Pill, alors la Tile associée à cette Pill devient null.

Pacman	
hasPower	boolean
isOpenMouth	boolean
wantedDirection	Direction
direction	Direction
image_op	BufferedImage
image_cl	BufferedImage
image_p_op	BufferedImage
image_p_cl	BufferedImage
imageActualOp	BufferedImage
imageActualCl	BufferedImage
Pacman(Tile)	
move(Tile)	void
setDirection(Direction)	void
getDirection()	Direction
setWantedDirection(Direction)	void
getWantedDirection()	Direction
getHasPower()	boolean
setHasPower(Boolean)	void
getImage()	BufferedImage
rotateImage(BufferedImage, double)	BufferedImage

Nous avons rajouté beaucoup d'attributs et méthodes dans la classe Pacman. L'attribut isOpenMouth indique si Pacman a la bouche ouverte ou fermée, puisqu'il ouvre/ferme celle-ci de manière continue pendant la partie. En fonction de cette ouverture, l'image à afficher est différente. Les images de Pacman dans ses différents états sont aussi des attributs rajoutés, comme image_cl ou encore image_op.

Le mouvement de Pacman est différent par rapport à ce qui avait été prévu. Pacman se déplace sur la case se situant dans la direction actuelle. Quand l'utilisateur appuie sur une des flèches, la « wantedDirection » de Pacman change, et dès qu'il a l'occasion d'aller dans cette direction, il s'y rend et sa « direction » est changée. Si aucun changement de direction n'est demandé, ou si la direction voulue n'est pas accessible, il continue dans sa direction actuelle.