**Déroulement du template donné**

**Légende :**

**Classe** *méthode*

Le jeu commence grâce à la fonction **Main** qui se trouve dans le dossier Start*.* On y initialise le jeu (**PacmanGame** qui utilise un fichier d’aide/règles), le tableau graphique (**PacmanPainter**) et la classe qui va gérer les contrôles de jeu (Faire avancer avec ZQSD).

Ces trois classes spécifiques (chaque jeu est différent) sont utilisés par le moteur générique **GameEngineGraphical**.

**On peut découper les classes en trois familles : les classes Graphiques, les classes Controles, les classes Jeu.**

**1. Classes Graphiques**

**GameEngineGraphical** met en place une méthode *run* qui crée un gui(Graphical User Interface/ Interface Graphique) et récupère les commandes utilisateurs tant que le jeu n’est pas terminé pour le faire évoluer.

GUI : Est un **GraphicalInterface** qui crée un nouveau **Jframe**, un **DrawingPannel** qui est un contenu du **Jframe**. Les quatre commandes f. permettent d’ajuster la fenêtre à la taille des éléments concernés, rend visible le graphique et recentre l’image. On met ensuite à jour le dessin avec *paint()* qui appelle **DrawGame** dans **DrawingPannel**

**DrawingPanel** est un **JPanel** (panneau sur une fenêtre) où l'on peut dessiner une image en utilisant un **GamePainter**.

Son constructeur fait appel à un **GamePainter** pour prendre ses dimensions pour le **JPanel** et les buffers d'images.

La méthode *drawGame()* met à jour le rendu de l'image sur le **JPanel** avec le bufferedImage (du **GamePainter**).

**GamePainter** est une classe abstraite d’une interface. Elle représente la manière de dessiner un **Jpanel**.

**PacmanPainter** : Implémente la classe **GamePainter**. Initialise width et height à 100 (cadre 100x100). Avec *draw()* renvoie une image du jeu en 2D grâce aux images fournies.

**2. Classes Jeu**

**Pacman Game** implémente la classe **Game**. On utilise un fichier Source « help.txt » qui est affiché. La classe vérifie si le jeu est terminé et exécute la commande en cours.

**Game** est une interface qui permet l’évaluation en entrant une commande, vérifie si le jeu est fini ou non.

**3. Classes Controles**

**GameController** est une interface héritée de **KeyListener** (permet le déplacement sans devoir taper une valeur et puis appuyer sur Enter : appuyer sur Z fait Z) Cette classe récupère la commande en cours.

**PacmanController** implémente la classe **GameController**, prend en argument la commande en cours. On initialise la commande comme étant IDLE (équivalent de « en attente »). *Keypressed* récupère la dernière touche pressée par l’utilisateur et défini en fonction la direction que va prendre le héros. Si on n’appuie sur rien, la commande est définie sur IDLE.