TD 16 - Projet Zeldiablo - Moteur Graphique

Dans cette itération, l'objectif est d'intégrer un moteur de jeu graphique dans votre projet. Des classes génériques vous sont fournies sur arche pour afficher le jeu et récupérer les commandes de l'utilisateur et il va falloir vous interfacer avec celles-ci.

1 Moteur de Jeu Générique

Un moteur de jeu est en charge d'organiser l'exécution du jeu. Globalement un moteur de jeu a pour objectif

- de lancer le jeu ;
- de gérer l'interaction avec l'utilisateur ;
- de gérer l'évolution du jeu ;
- de gérer l'affichage du jeu ;
- d'attendre un léger temps pour assurer un framerate constant.

1.1 Description des classes du moteur

Le moteur fourni est composé de plusieurs classes décrites dans le diagramme de classe présenté en figure 1. Les classes présentées se trouvent dans le package `moteurJeu` et permettent l'exécution et l'affichage d'un `Jeu`.

De manière générale,

- le point d'entrée pour exécuter un jeu est la classe `MoteurGraphique` qui possède trois attributs : un objet `Jeu`, un objet `DessinJeu` et un objet `InterfaceGraphique` :
- l'objet `Jeu` est en charge de stocker les données du jeu et de gérer leur évolution. C'est une interface qu'il faut implémenter lorsqu'on souhaite faire un nouveau jeu. Un jeu est caractérisé par la méthode `void evoluer(Commande c)` qui fait évoluer les données du jeu en fonction de la commande utilisateur.
- l'objet `DessinJeu` est un objet en charge de dessiner le jeu. C'est une interface dont il faut définir la méthode `dessinerJeu(BufferedImage image)` pour l'adapter à son jeu (cf ci-dessous). La méthode `dessinerJeu` a pour objectif de dessiner l'image du jeu sur l'image passée en paramètre `image`.
- l'objet `InterfaceGraphique` est géré en interne et fait le lien avec les packages java, en particulier pour la gestion des actions utilisateur `KeyListener` et le rendu dans un `JPanel`.

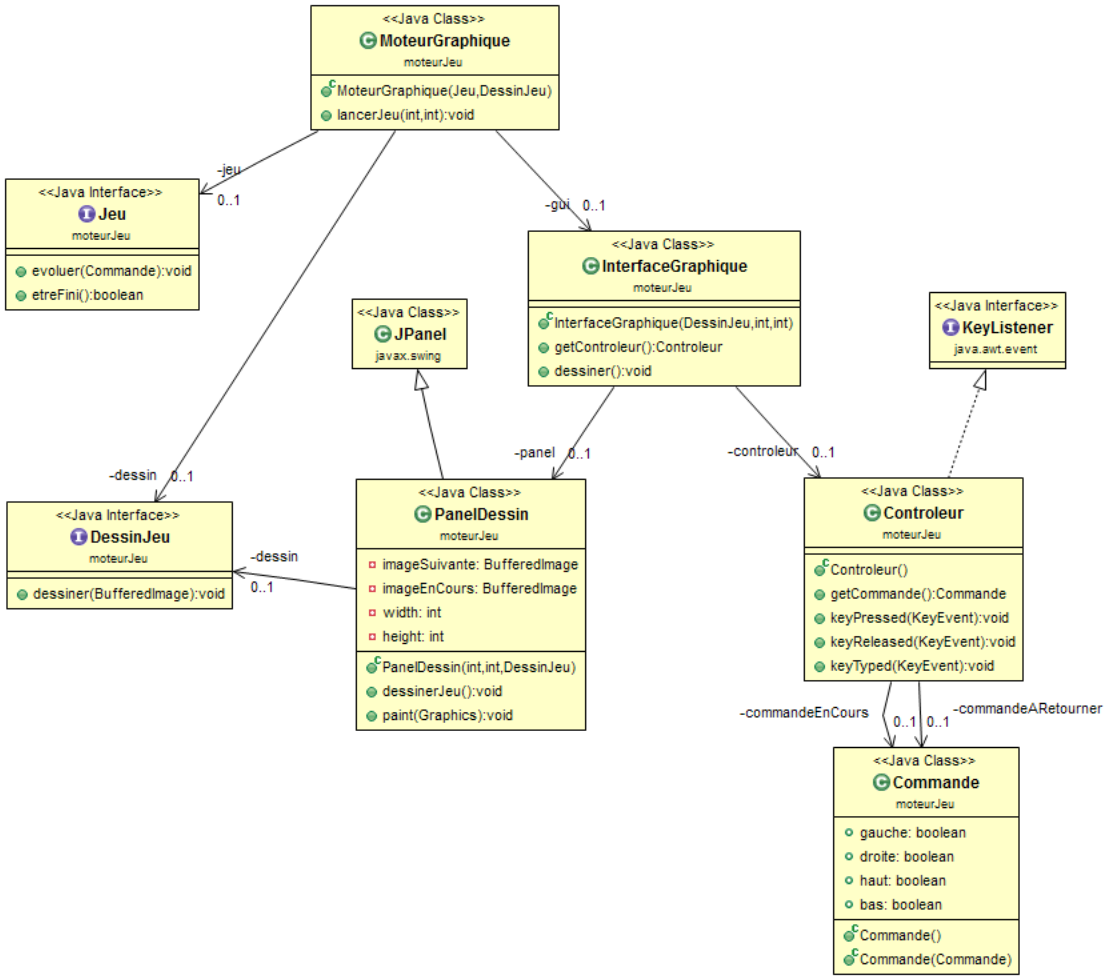

FIGURE 1 – Diagramme de classe décrivant les classes du package `moteurJeu`

1.2 Description du fonctionnement du moteur

A l'utilisation, on doit disposer d'un objet `jeu` implémentant `Jeu` et d'un objet `dessin` implémentant `DessinJeu`. Lorsqu'on dispose de ces deux objets, lancer un jeu fonctionne de la manière suivante (cf diagramme de séquence de la figure 2)

- on construit un `MoteurGraphique` avec les objets `jeu` et `dessin` en paramètre ;
- on appelle la méthode `lancerJeu(int x, int y)` en passant en paramètre la taille (x,y) de l'écran ;
- le moteur se charge alors de construire l'interface graphique et les composants ;
- la boucle de jeu se lance, elle consiste à chaque itération à
  - ☐ récupérer la commande utilisateur correspondant au touches appuyées du clavier via l'objet `Controleur` de l'interface graphique ;
  - ☐ faire évoluer le jeu en conséquence en appelant la méthode `evoluer(Commande c)` de la classe `Jeu` ;
  - ☐ demander l'affichage du jeu via le `PanelDessin` qui appelle la méthode `dessiner` de l'objet `dessin` passé à la construction.

Attention :
Les touches utilisées par défaut sont les touches `QSDZ`.

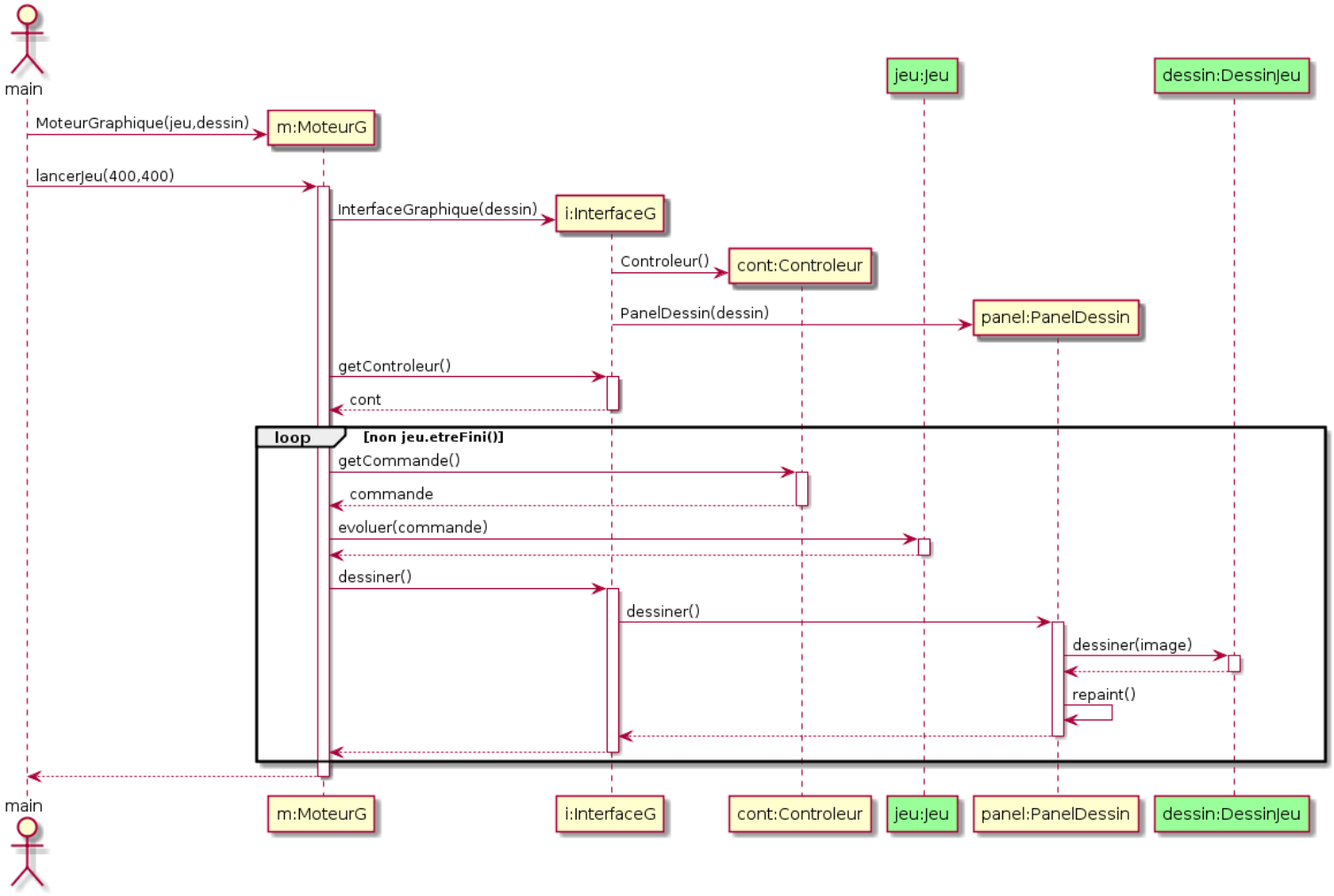

FIGURE 2 – Diagramme de séquence décrivant comment fonctionne le moteur de jeu. Les objets dont on dispose initialement sont en vert.

1.3 Utilisation du moteur

Lorsqu'on souhaite créer un nouveau jeu, il suffit simplement de définir une nouvelle classe implémentant `Jeu` et une nouvelle classe implémentant `DessinJeu` puis de passer les objets correspondant lors de la construction du `MoteurGraphique`.

Vous trouverez un exemple de jeu utilisant le moteur de jeu dans le package `monJeu` fourni sur arche (cf diagramme de classes figure 3) :

- la classe `MonJeu` est le jeu à lancer, il implémente l'interface `Jeu`, contient les données du jeu et définit la méthode `evoluer` qui encapsule les règles du jeu ;
- la classe `DessinMonJeu` correspond à l'afficheur permettant d'afficher le jeu `JeuTest` et hérite de `DessinJeu`. La classe `DessinMonJeu` connait le `Jeu` afin de l'afficher lorsque le moteur le demande ;
- d'autres classes liées au jeu sont nécessaires pour manipuler correctement les données (comme la classe `Personnage`).

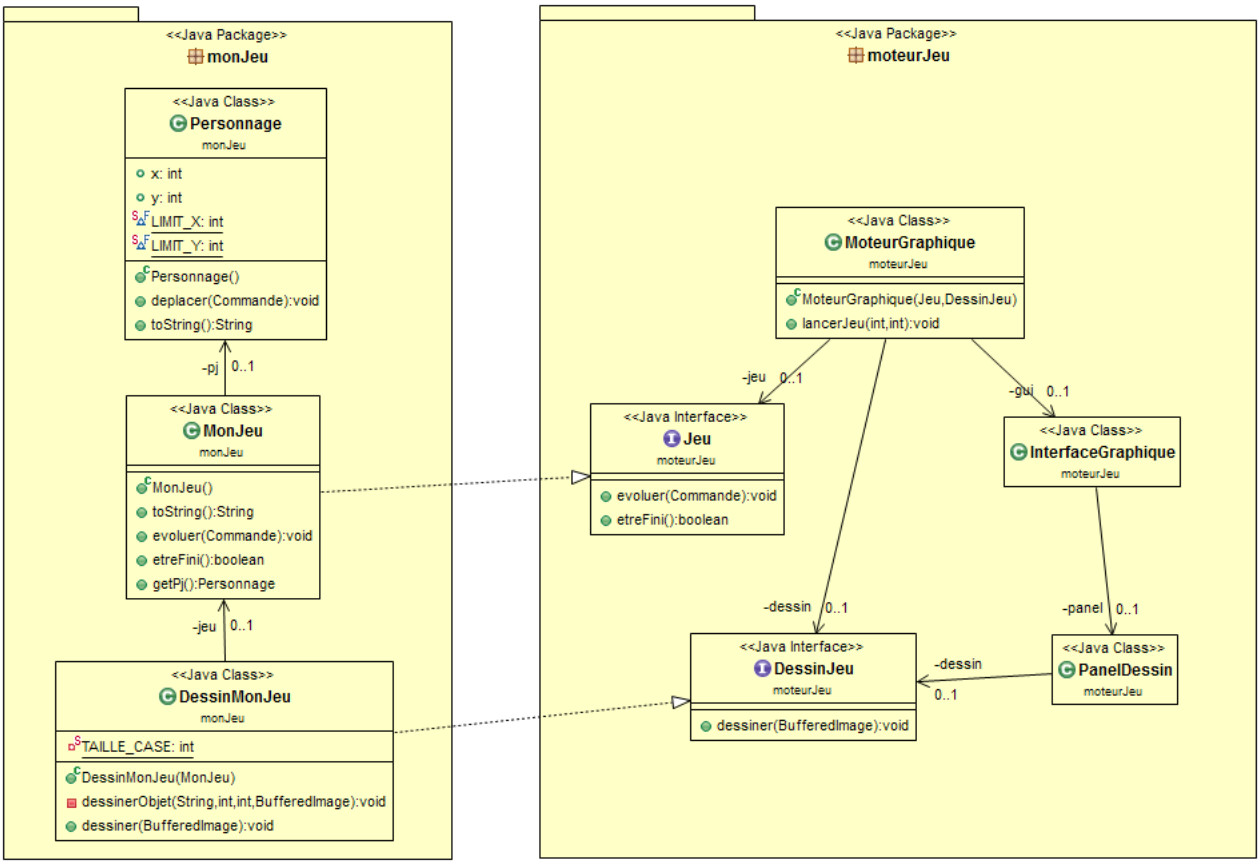

FIGURE 3 – Diagramme de classe décrivant la manière d'utiliser le moteurGraphique

2 Version v2.0

L'objectif de ce TP est d'adapter votre version du projet pour avoir un rendu graphique temps réel (à partir des classes présentées ci-dessus et fournies sous arche).

En cas de besoin, vous pouvez vous référer à l'annexe en fin de TP qui décrit la mécanique interne du moteur de jeu.

Question
Regarder comment ces classes fonctionnent puis proposer un diagramme de séquence modifié qui intègre vos classes.

Question
Modifier votre diagramme de classe pour intégrer les classes fournies.

Question
Modifier votre code java pour intégrer le moteur graphique à votre jeu.

Question
Ajouter le tag `v2` et stocker les différents documents dans votre dossier `documents\version2`.

3 Annexe : Fonctionnement interne du moteur de jeu

3.1 Gestion des actions utilisateur

La classe `ControleurGraphique` est chargée de gérer les interactions avec l'utilisateur :

- La classe `Controleur` implémente un `KeyListener` et définit donc les méthodes d'interaction clavier `keyPressed`, `keyReleased` et `keyTyped`. Désormais ces méthodes modifient un objet `Commande` en attribut privé ;
- La classe `Controleur` possède une méthode `Commande getCommande()` retournant l'attribut `Commande commandeARetourner` décrivant la commande actuelle du joueur ;
- Cet objet `Commande` possède en attribut un ensemble de booléens décrivant les touches appuyées par le joueur à l'instant où la méthode `getCommande()` est appelée.

Ainsi, il suffit d'appeler la méthode `getCommande()` pour connaitre les touches appuyées sur le clavier à chaque instant.

3.2 Gestion de l'affichage

L'interface `DessinJeu` a pour objectif de dessiner l'état du jeu

- La classe `PanelDessin` possède une méthode `dessinerJeu()` qui doit afficher à l'écran l'état du jeu. Pour cela la classe `PanelDessin` possède en attribut un objet de type `DessinJeu`. Elle passe en paramètre une image à l'objet `DessinJeu` via la méthode `dessiner(BufferedImage bf)`.
- La méthode `void dessiner(BufferedImage im)` de l'interface `DessinJeu` a pour objectif d'ajouter à l'image passée en paramètre les éléments à afficher. Cette méthode est à définir pour l'adapter à votre jeu ;
- Une fois l'image dessinée, la classe `PanelDessin` recopie l'image comme image à afficher et appelle la méthode repaint() ;
- Par exemple, la classe exemple `DessinMonJeu` redéfinit la méthode `dessiner` et utilise une méthode intermédiaire `void dessinerObjet(String s,int i, int j)` permettant de dessiner en mémoire l'objet décrit par la chaine `s` à la case (i,j). Pour cela, on récupère un objet `graphics` permettant de dessiner sur l'image (méthode `getGraphics` de `BufferedImage`).

3.3 Interface graphique

La classe `InterfaceGraphique` rassemble les deux composants associés au moteur et construit l'interface graphique :

- Lorsqu'un objet de la classe `InterfaceGraphique` est construit, une `JFrame` est créée avec un `PanelDessin` qui contient un objet de type `DessinJeu` et sur lequel est greffé un objet de type `Controleur`
- l'objet de type `Controleur` est accessible par la méthode `getControleur()` de la classe `InterfaceGraphique` et permet de faire des requêtes sur les touches appuyées par l'utilisateur ;
- l'objet de type `DessinJeu` est utilisé par la méthode `dessiner()` du `PanelDessin` pour mettre à jour l'image à afficher.