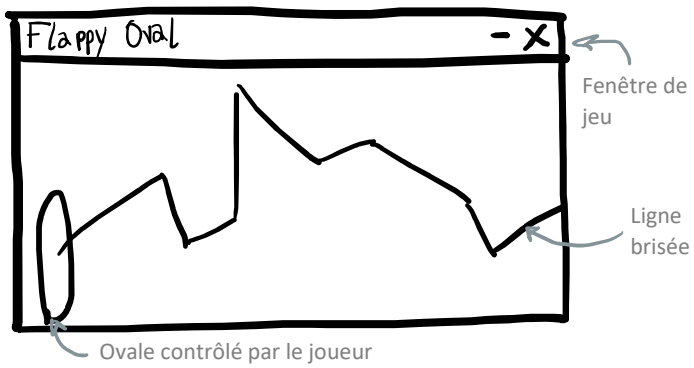
L'objectif du projet que nous entreprenons est la production d'un mini-jeu inspiré du jeu Flappy Bird. Le jeu comportera deux éléments : une ligne brisée et un ovale qui doit se déplacer le long de la ligne. Pour cela, le joueur pourra agir sur la position de l'ovale en cliquant sur la fenêtre, pour le faire sauter. Sinon, il redescendra de lui-même le reste du temps. La partie est perdue si l'ovale sort de la ligne brisée.
L'interface graphique sera donc très simplement composée du dessin d'un ovale et d'une ligne brisée.
Voici un brouillon de ce qui est imaginé pour cette interface graphique :



A travers ce projet, nous nous intéresseront au motif de conception MVC (Modèle, Vue, Contrôleur) pour garantir une synergie entre le programme et l'utilisateur. Ce modèle permettra de gérer les relations entre l'action du joueur et son environnement de jeu. (Ici cet environnement simplement composé de la ligne brisée.)

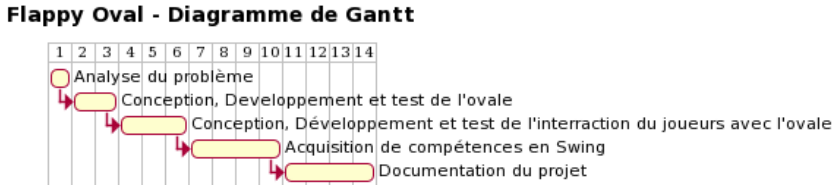
Lors de notre première séance, nous ne réaliseront pas toutes les fonctionnalités du projet décrit ci-dessus. On se concentrera donc sur un nombre réduit de fonctionnalités, minutieusement choisies pour leur rôle dans la suite du processus de développement de ce projet.

Les principales fonctionnalités à développer sont l'interface graphique, le défilement automatique de la ligne brisée (pour donner l'impression que l'ovale « avance » sur celle-ci) ainsi que le contrôle de l'ovale par le joueur : le programme doit réagir aux clics de l'utilisateur pour faire sauter l'ovale.
Nous avons lors la première session de travail amorcé ce projet en créant la base de l'interface graphique : une fenêtre. On y dessine un ovale, et on pourra par la suite y faire figurer la ligne brisée.
Nous avons également implémenté la réaction du programme lorsque l'utilisateur clique sur la fenêtre : l'ovale est propulsé vers le haut (il saute/ sa hauteur est modifiée).
Ce sont les fonctionnalités les plus simples à implémenter, et sont, notamment pour l'interface graphique, primordiale pour faciliter le développement de la suite. En effet, l'interface graphique permettra de faciliter le travail de test du jeu et de débug. L'interaction réalisant le saut de l'ovale représentant la mécanique principale du jeu, il était également nécessaire d'implémenter cette fonctionnalité dès le début du projet.

A partir de ces fonctionnalités, on peut dresser le plan de développement du projet. Il se décompose en cinq tâches essentielles à réaliser :

- Une analyse du problème. C'est une étape qui nous permet d'analyser l'objectif du projet et de décomposer celui-ci en sous-problèmes, pour ensuite organiser les tâches. Nous y consacrons 15 minutes.
- Concevoir, développer et tester l'affichage d'une fenêtre avec le dessin d'un ovale. (30 minutes)
- Concevoir, développer et tester la mécanique de saut de l'ovale en réponse au clic du joueur (45 minutes)
- Acquérir les compétences nécessaires à l'utilisation de la bibliothèque Swing (60 minutes)
- Documenter le projet (60 minutes)

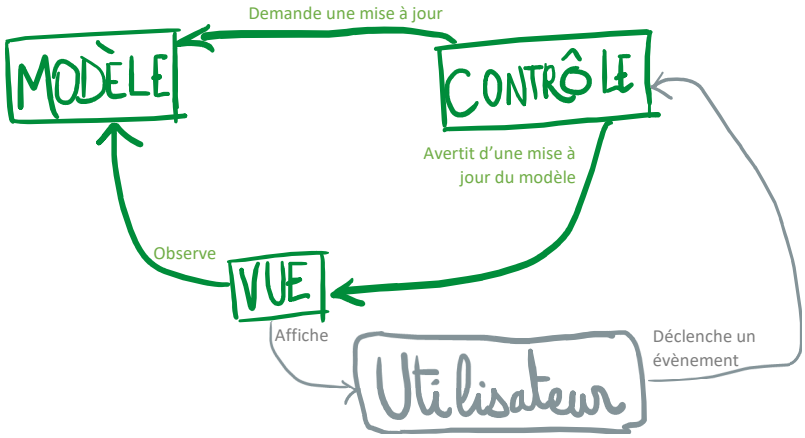
On aboutit au diagramme de Gantt suivant :



(Diagramme produit sur la plateforme https://www.planttext.com/ , une unité de temps représente 15 minutes.)

L'ensemble de la conception de ce projet s'articulera autour du motif de conception MVC (Modèle, Vue, Contrôle), selon lequel on organise les méthodes en trois parties. Les relations entre les trois parties seront les

suivantes : Le Modèle contient les données et les méthodes sur l'état actuel du programme. La Vue se charge de récupérer les données du modèle et de les utiliser pour créer l'interface graphique. Le contrôle sert à faire le lien entre ces deux parties et l'utilisateur. Lorsqu'un évènement de contrôle (ici un clic de souris) est réalisé par l'utilisateur, on actualise le modèle et on informe la vue de cette mise à jour du modèle, afin de modifier l'affichage en conséquence.



Pour la première fonctionnalité, l'interface graphique, nous utilisons la bibliothèque Swing ainsi que la classe JPanel. Cela nous permet de créer une fenêtre dont la dimension est définie par les constantes `hauteur` et `largeur`. Cela nous permet également de dessiner l'ovale sur la fenêtre, avec des constantes de dimension ainsi qu'une constante qui définit sa position sur l'axe des abscisses. (Respectivement, les constantes `ovalWidth`, `ovalHeight` et `bordX`. (Cf : diagramme de classe figure 1.)

Ensuite, la fonctionnalité qui permet l'interaction du joueur avec le programme nécessite une classe qui est attentive aux évènements extérieurs au programme. (Ici, c'est un clic droit de l'utilisateur sur la fenêtre.) On utilise donc une classe Contrôle qui implémente une instance de la classe MousseListener. Cette fonctionnalité nécessite donc la présence d'un attribut de `hauteur` de l'ovale et de valeur du `saut` dans la classe Etat, pour pouvoir effectuer les modifications nécessaires lorsque la classe Contrôle indique un évènement la classe Etat. (Cf. figure 2)

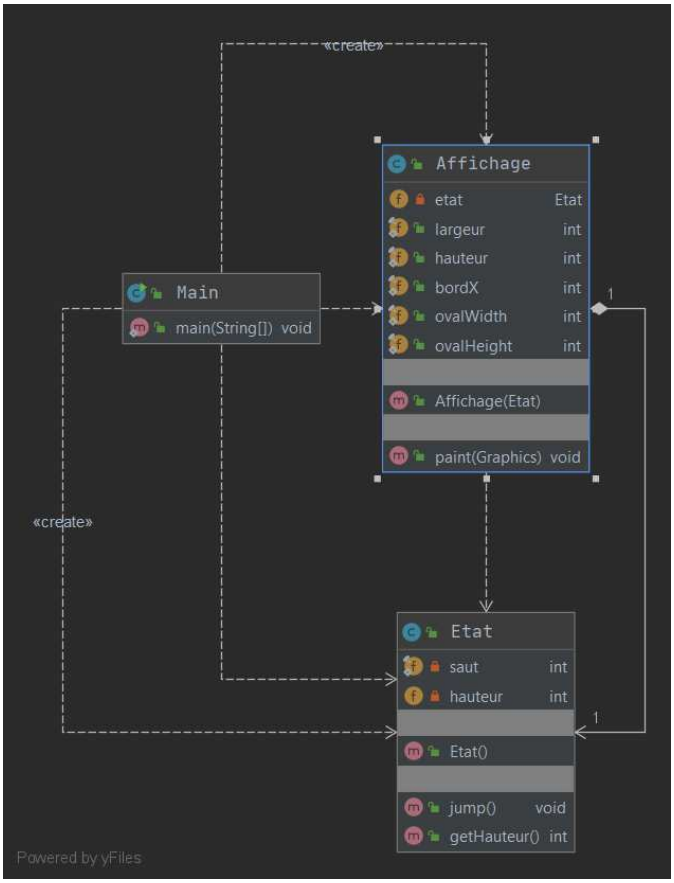


Figure 1 : Fonctionnalité d'affichage de l'interface graphique

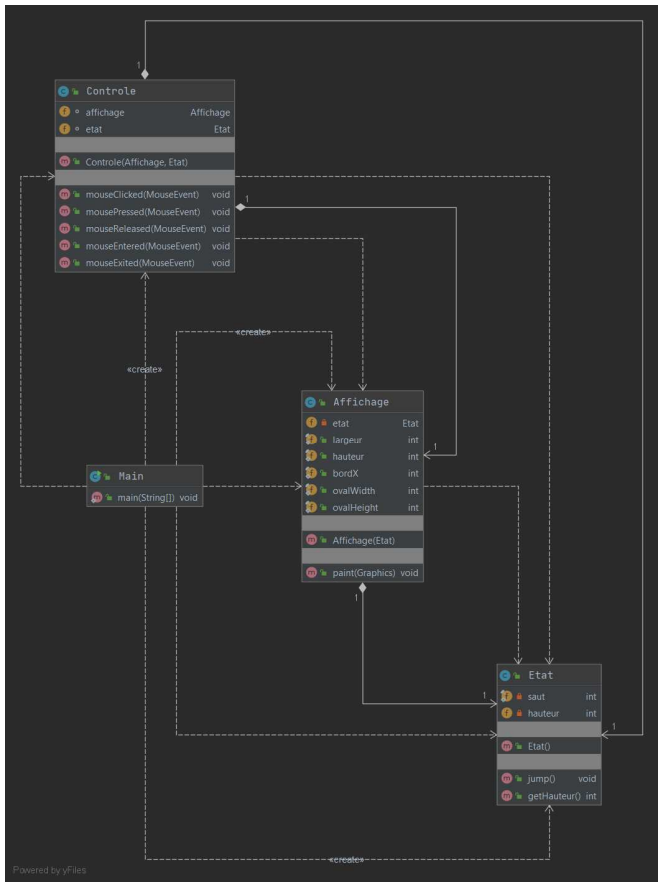


Figure 2 : Gestion des interactions avec le programme selon le motif MVC

Le résultat obtenu lors de l'exécution du programme est donc le suivant :

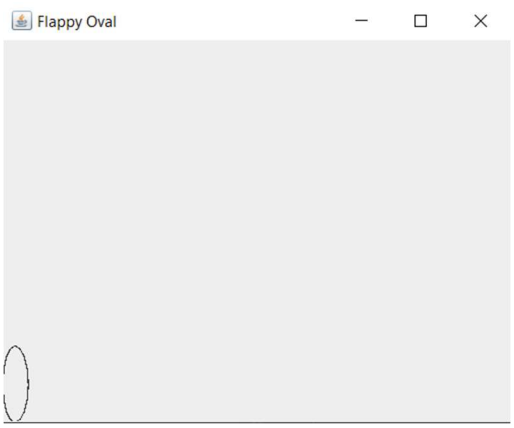


Figure 3 : Etat initial de l'interface graphique

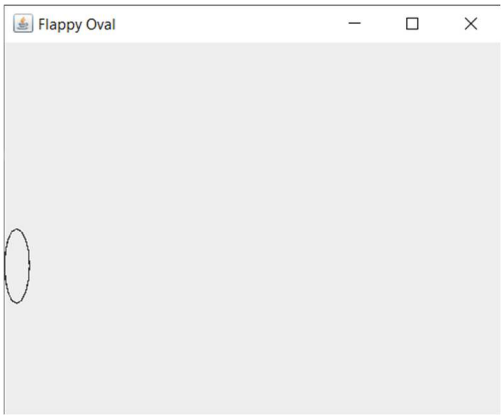


Figure 4 : ...Après quelques sauts

Pour exécuter ce programme, il faudra disposer de Java, avec un IDE. Une fois le projet importé dans votre IDE, sélectionnez la classe Main dans le package principal (src) puis « Run as Java Application ». Pour faire monter l'ovale, il suffit de cliquer n'importe où sur la fenêtre qui vient de s'ouvrir.

Les prochaines fonctionnalités à implémenter pour ce projet sont la création et l'affichage d'une ligne brisée, ainsi que le défilement automatique de celle-ci vers la gauche, pour donner l'impression que l'ovale se déplace sur celle-ci, vers la droite.

Avec les deux premières fonctionnalités que nous avons implanté lors de cette séance, les base du projet sont solidement posées. Le squelette du motif MVC ainsi formé permettra, lors de la suite de l'écriture du code, de rester organisé. L'affichage graphique du programme permettra un retour rapide et concret sur certaines fonctionnalités, faciliterons l'implémentation de test, ce qui nous permettra d'être plus efficace. A terme, et après avoir implémenté les fonctionnalités citées dans le paragraphe précédent, il faudra implémenter les « règles du jeu » : détecter s'il y a bien collision entre l'ovale et la ligne brisée, sinon, déclencher la fin de la partie.