Preuves

Pac-Man

Florent MARQUES – Faustin LEPERON

Documentation

Je sais concevoir un diagramme UML intégrant des notions de qualité et correspondant exactement à l'application que j'ai à développer.

Notre diagramme de classes se situe dans le répertoire « documentation/Diagramme_de_classes.pdf ».

Je sais décrire un diagramme UML en mettant en valeur et en justifier les éléments essentiels.

Notre description du diagramme de classes se situe dans le répertoire « documentation/Description_Diagramme_de_classes.pdf ».

Je sais documenter mon code et en générer la documentation.

La documentation du code se situe dans le répertoire « documentation/javadoc ».

Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert.

La description du contexte de notre application se trouve dans le répertoire « documentation/Contexte.pdf ».

Je sais faire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application

Notre diagramme de cas d'utilisation se situe dans le répertoire « documentation/Cas_Utilisations.pdf ».

Code

Je maîtrise les règles de nommage Java.

Tous les packages sont écrits en minuscules. Le nom des variables suit la règle du CamelCase. Le nom des classes commence par une majuscule.

Je sais binder bidirectionnellement deux propriétés JavaFX.

Nous n'avons pas utilisé de binding bidirectionnel.

Je sais binder unidirectionnellement deux propriétés JavaFX.

Nous avons bindé unidirectionnelement les positions des entités avec leurs images sur la vue.

Je sais coder une classe Java en respectant des contraintes de qualité de lecture de code.

Nous avons commenté et documenté notre code. Dans nos classes, nous mettons d'abord les variables puis les constructeurs puis les méthodes.

Je sais contraindre les éléments de ma vue, avec du binding FXML.

Dans le code behind de nos vues, nous avons des éléments avec l'annotations @FXML pour avoir une référence vers l'objet du fichier FXML associé. Par exemple dans le fichier GameOverView.java avec les champs :

```
@FXML
public Label scoreLabel;
@FXML
public TextField textField;
```

Je sais définir une CellFactory fabriquant des cellules qui se mettent à jour au changement du modèle.

Nous avons implémenté une CellFactory pour l'affichage des scores dans la vue de score « view/ScoreView.java ».

Je sais éviter la duplication de code.

Le code est découpé en méthode pour pouvoir réutiliser les méthodes plusieurs fois et donc éviter la duplication de code.

Je sais hiérarchiser mes classes pour spécialiser leur comportement.

Nous avons hiérarchisé nos classes : entités, déplaceurs, collisionneurs, boucleurs, managers.

Je sais intercepter des évènements en provenance de la fenêtre JavaFX.

Dans la classe Game, nous avons créé une méthode « onKeyPressed » pour récupérer les touches appuyées par l'utilisateur et qui est positionné lorsque la partie commence.

Je sais maintenir, dans un projet, une responsabilité unique pour chacune de mes classes.

Chaque classe que nous avons créée possède une seule responsabilité. La classe « PacManDisplacer » ne sert qu'à déplacer Pac-Man ou bien la classe « MovementLooper » ne sert qu'à boucler.

Je sais gérer la persistance de mon modèle.

Nous utilisons la sérialisation pour sauvegarder les scores et les pseudonymes des joueurs.

Je sais utiliser à mon avantage le polymorphisme.

Dans notre classe abstraite « BaseEater », il y a un attribut de type « BaseCollider » qui est une interface et qui permet donc d'utiliser la méthode « isCollide » ou « getColliding » sur les classes qui implémentent cette interface. La méthode appelé dépendra du type dynamique de la classe instanciée.

Je sais utiliser GIT pour travailler avec mon binôme sur le projet.

Nous avons utilisé GIT pour travailler de manière asynchrone puis effectuer le merge du travail. (voir historique sur gitlab).

Je sais utiliser le type statique adéquat pour mes attributs ou variables.

Nous avons une liste de « BaseEntity » qui permet de regrouper les différentes entités disponibles sur la carte sans avoir plusieurs listes de chaque type d'entités.

Je sais utiliser les différents composants complexes (listes, combo…) que me propose JavaFX.

Nous avons utilisé une ListView pour l'affichage de nos scores.

Je sais utiliser les lambda-expression.

Nous utilisons les lambda-expression à plusieurs endroits comme dans le Launcher avec :

```
stage.setOnCloseRequest(event -> game.close());
```

Je sais utiliser les listes observables de JavaFX.

Nous utilisons une liste observable dans notre classe Game pour les scores des joueurs.

Je sais utiliser un convertisseur lors d'un bind entre deux propriétés JavaFX.

Nous avons utilisé un convertisseur pour convertir le score et le pseudonyme d'un joueur en texte grâce à la classe « PlayerScoreConverter ».

Je sais utiliser un fichier CSS pour styler mon application JavaFX.

Nous utilisons un fichier CSS pour styliser les éléments de notre application JavaFX. Il se trouve dans le répertoire « css/stylesheet.css ».

Je sais utiliser un formateur lors d'un bind entre deux propriétés JavaFX.

Nous n'avons pas utilisé de formateur lors d'un bind.

Je sais développer un jeu en JavaFX en utilisant FXML.

Nous avons créé les vues de notre jeu grâce au FXML.

Je sais intégrer, à bon escient, dans mon jeu, une boucle temporelle observable.

Nous avons intégré 3 boucles temporelle observable, le déplacement du Pac-Man, le déplacement des fantômes et les animations des entités.