Une image contenant insecte

Description générée automatiquement**Document Preuves**

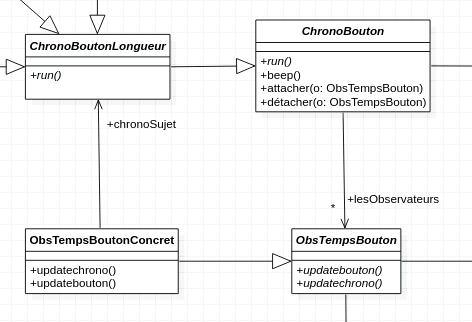
Bastien Malvezin (G3)

Matthias Bizet (G4)

**Documentation**

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

Cf. « Diagrammes.mdj »

Notion de qualité : réalisation d’un

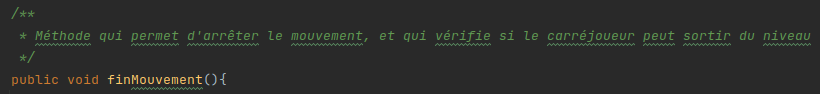
patron de conception Observateur

de type comportemental

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

cf. « DescriptionDiagrammeUML.docx »

**Je sais documenter mon code et en générer la documentation**

****

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

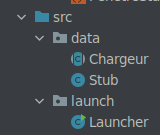
Cf. « contexte.docx »

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

Cf.  « Diagrammes.mdj »

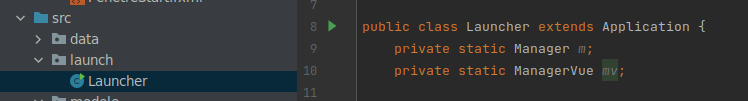
**Code**

**Je maîtrise les règles de nommage Java**

****

Les noms des classes sont en majuscule et les packages en minuscule :

Les fichiers des classes ont le même nom que ces dernières :



**Je sais binder bidirectionnellement deux propriétés JavaFX**

non

Une image contenant texte

Description générée automatiquement**Je sais binder unidirectionnellement deux propriétés JavaFX**

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

Une image contenant texte

Description générée automatiquementL’attribut Positions est accessible via un getter et on peut le modifier avec une méthode setP()

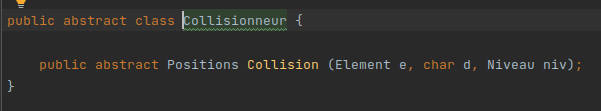
**Je sais contraindre les éléments de ma vue, avec du binding FXML**

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

non

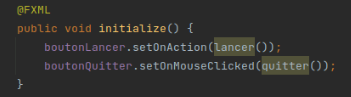
**Je sais éviter la duplication de code**

**Je sais hiérarchiser mes classes pour spécialiser leur comportement**

Une image contenant texte

Description générée automatiquement

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

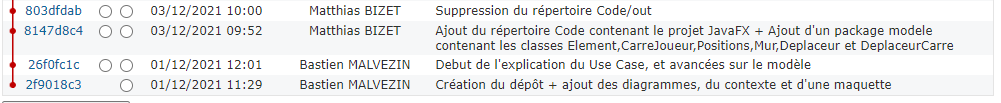
La méthode lancer() se lance quand le boutonLancer est appuyé.

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

**Je sais gérer la persistance de mon modèle**

**Je sais utiliser à mon avantage le polymorphisme**

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

****

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

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

non

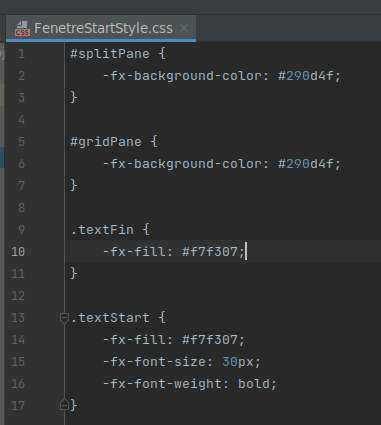
**Je sais utiliser les lambda-expression. Je sais utiliser les listes observables de JavaFX**

non

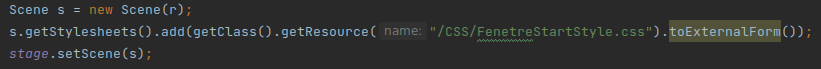
**Je sais utiliser un convertisseur lors d’un bind entre deux propriétés JavaFX**

non

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



****

****

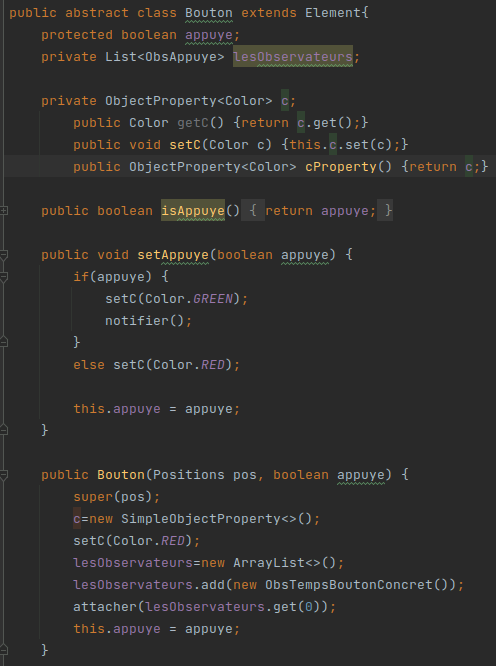
**Je sais utiliser un formateur lors d’un bind entre deux propriétés JavaFX**

non

**Je sais développer un jeu en JavaFX en utilisant FXML**

cf. dossier FXML de rsrc

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

La boucle temporelle de ce jeu réside dans le fait qu’un bouton se désactive au bout d’un certains temps.

A chaque fois qu’on va appeler la méthode setAppuye(), on appelle la méthode notifier() qui va appeler la méthode updatebouton de tous ses observateurs.

