Romain Monteil & Clara Poncet G1-2

PREUVES

DOCUMENTATION

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

Le diagramme de classe est dans le dossier *Documentation*. Il possède des points d'extension (Open/Close Principle) et chaque classe ne répond qu'à une seule responsabilité (Single Responsibility Principle), notre diagramme intègre donc des notions de qualité.

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

Chaque diagramme est accompagné d'une description explicitant les points importants.

-Je sais documenter mon code et en générer la documentation.

Chaque méthode et chaque classe est commentée.

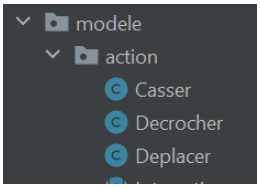
-Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert.

Le contexte est dans le dossier *Documentation.*

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

Le diagramme de cas d'utilisation est dans le dossier *Documentation*, chaque cas est détaillé dans la description du diagramme.

CODE



-Je maîtrise les règles de nommage Java.

Les classes sont nommées avec des majuscules et les packages des minuscules.

-Je sais binder bidirectionnellement deux propriétés JavaFX.

```
leFieldPseudo.textProperty().bindBidirectional(m.laPartie.pseudoProperty());
```

Un changement dans la vue déclenchera un changement dans le modèle et inversement. (NouvellePartie.java ligne 32)

-Je sais binder unidirectionnellement deux propriétés JavaFX.

```
img.layoutXProperty().bind(objet.positionXProperty());
img.layoutYProperty().bind(objet.positionYProperty());
```

Un changement dans le modèle changera la vue mais la vue ne peut pas modifier le modèle. (EcranJeu.java ligne 109)

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

Le code est commenté et indenté afin d'en faciliter la lecture, les classes sont facilement compréhensibles car elles ne répondent qu'à une seule intention (Single Responsibility Principle)

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

```
<Button fx:id="btnJouer" disable="${leFieldPseudo.text == 'pseudo'}"
```

Le bouton n'est pas cliquable si le pseudo n'a pas été changé de celui par défaut (qui est « pseudo »). (NouvellePartie.fxml ligne 31)

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

Nous n'avons pas utilisé de CellFactory dans notre projet.

-Je sais développer une application graphique en JavaFX en utilisant FXML.

Notre jeu s'affiche et est fonctionnel sous forme d'application graphique.

-Je sais éviter la duplication de code. Je sais hiérarchiser mes classes pour spécialiser leur comportement.

Nous avons utilisé des classes abstraites afin d'éviter la duplication du code et les comportements sont spécialisés par la création de classes filles qui héritent de ces classes abstraites. Par exemple les objets : la classe mère Objet est abstraite, les classes Armoire, Fauteuil etc. spécialisent le comportement de chaque objet.

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

```
img.setOnMouseClicked(new EventHandler<javafx.scene.input.MouseEvent>() {
    @Override
    public void handle(javafx.scene.input.MouseEvent mouseEvent) {
        if (objet.isCliquable()) {objet.getInteraction().interagir(objet);}
    }
});
```

Le clique sur une image dans une fenêtre graphique déclenche un comportement du modèle. (EcranJeu.java ligne 113)

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

Chaque classe a une seule responsabilité. Par exemple, un objet ne se déplace pas mais il appelle la méthode de son attribut interaction qui le fait se déplacer.

-Je sais gérer la persistance de mon modèle.

Une partie (composée d'un pseudo et d'un niveau) peut être sauvegardée (écrite dans un fichier binaire) puis rechargée pour pouvoir reprendre une partie en cours.

-Je sais surveiller l'élément sélectionné dans un composant affichant un ensemble de données.

Nous n'avons pas utilisé cela dans notre projet.

-Je sais utiliser à mon avantage le polymorphisme.

Chaque Objet peut avoir un comportement différent. En changeant son attribut interaction, on peut décider si un objet doit se déplacer, se casser ou autre après un clic sur l'objet.

-Je sais utiliser certains composants simples que me propose JavaFX.

Exemple : EcranJeu.fxml lignes 8 à 19.

```
<Pane fx:id="lePane" prefHeight="400.0" prefWidth="600.0" stylesheets="@../style.css" xmlns="http://ja
    <children>
        <Button fx:id="btnPorte" layoutX="400.0" layoutY="50.0" mnemonicParsing="false" onAction="#cli
        <Label fx:id="leLabel" alignment="CENTER" contentDisplay="CENTER" layoutX="136.0" layoutY="348
            <font>
                <Font size="24.0" />
            </font>
        </Label>
        <Label fx:id="numPiece" layoutX="41.0" layoutY="348.0" prefHeight="50.0" text="Pièce n° ?" />
      <Button layoutX="333.0" layoutY="360.0" mnemonicParsing="false" text="Sauvegarder et quitter" on
    </children>
</Pane>
```

-Je sais utiliser certains layout que me propose JavaFX.

Nous avons utilisé des Pane et des GridPane pour noes vues.

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

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

Nous n'avons pas utilisé d'attribut statique dans notre projet.

-Je sais utiliser les collections.

Piece.lesObjets et Jeu.lesPieces

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

Nous n'avons pas utilisé ces composants dans notre projet.

-Je sais utiliser les lambda-expression.

Nous n'avons pas utilisé de lambda-expressions dans notre projet.

-Je sais utiliser les listes observables de JavaFX.

Nous n'avons pas utilisé de listes observables dans notre projet.

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

Nous n'avons pas utilisé cela dans notre projet.

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

```
stylesheets="@../style.css"
```
EcranJeu.fxml ligne 8

```
#btnPorte {
    -fx-background-image: url('/img/porte.png');
}
```

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

Nous n'avons pas utilisé cela dans notre projet.