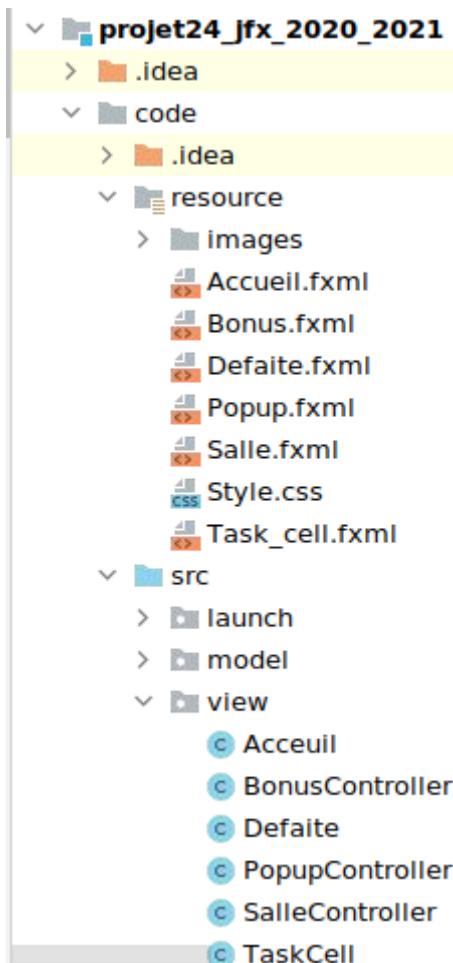


Preuves

Je maîtrise les règles de nommage Java.



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

```
vie.textProperty().bindBidirectional(joueur.pdvProperty(), new NumberStringConverter());  
vietot.textProperty().bindBidirectional(joueur.pdvMaxProperty(), new NumberStringConverter());
```

Je sais binder unidirectionnellement deux propriétés JavaFX.

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








Nous avons essayé d'appliquer au mieux possible les principes solides et utiliser certains patrons de conception tel que le singleton pour le Manager.

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.

```
TaskCell.java x
1 package view;
2
3 import ...
12
13 public class TaskCell extends ListCell<Carte> {
14
15     @FXML
16     private ImageView image;
17
18     @FXML
19     private Label titre;
20
21     @FXML
22     private Label effet;
23
24     public TaskCell() { loadFXML(); }
27
28     private void loadFXML() {...}
38
39     @Override
40     protected void updateItem(Carte carte, boolean empty) {...}
53 }
```

Je sais développer une application graphique en JavaFX en utilisant FXML & Je sais utiliser un fichier CSS pour styler mon application JavaFX.

-  Accueil.fxml
-  Bonus.fxml
-  Defaite.fxml
-  Popup.fxml
-  Salle.fxml
-  Style.css
-  Task_cell.fxml

Je sais éviter la duplication de code.

Je sais hiérarchiser mes classes pour spécialiser leur comportement.

```
public class Boss extends Monstre {
```

Nous avons une classe Boss qui découle de la classe Monstre.

Je sais intercepter des évènements en provenance de la fenêtre JavaFX & Je sais surveiller l'élément sélectionné dans un composant affichant un ensemble de données.

```
@FXML public void handleMouseClicked() throws IOException {  
    int selectedCarteIndex = deckListView.getSelectionModel().getSelectedIndex();  
    selectedItem = deckListView.getSelectionModel().getSelectedItem();  
    useCard(selectedItem);  
    this.joueur.remplacerDeckCarte(selectedCarteIndex);  
}
```

Lors d'un click sur une carte, on la joue puis on la change.

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

Je sais gérer la persistance de mon modèle.

```
public void serialiser(ObjectOutputStream oos){...}
```

```
public void deserialiser(ObjectInputStream ois){...}
```

Nous avons des fonctions de sérialisation et de désérialisation.

Je sais utiliser à mon avantage le polymorphisme.

```
Monstre b = new Boss( n: "Boss", pdv, i, degats);  
s.setMonstre(b);
```

A la création d'une nouvelle salle dans une partie on crée un Boss lors des salles qui sont des multiples de 5.

Je sais utiliser certains composants simples que me propose JavaFX & Je sais utiliser certains layout que me propose JavaFX.

```
<Button fx:id="defaite" alignment="TOP_RIGHT" contentDisplay="RIGHT" onAction="#defaite" text="Abandonner"
      textAlignment="RIGHT" GridPane.columnIndex="2" GridPane.halignment="RIGHT"/>
<Button fx:id="Sauvegarde" alignment="TOP_LEFT" contentDisplay="RIGHT" onAction="#sauvegarde"
      text="Sauvegarder" textAlignment="RIGHT" GridPane.columnIndex="0" GridPane.halignment="LEFT"/>
</GridPane>
</top>

<center>
  <GridPane fx:id="terrain">
    <columnConstraints...>
    <rowConstraints...>

    <VBox GridPane.columnIndex="0" GridPane.rowIndex="1" alignment="CENTER" fx:id="ptsdevie"...>
```

On utilise des boutons comme composants simples et des GridPane et VBox comme layouts.

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

The screenshot shows a Git web interface for a repository named 'projet24_jfx_2020_2021'. The top section displays repository statistics: 1,542 ko for the code and 768 octet for the documentation. Below this, the 'Dernières révisions' (Latest revisions) section shows a list of commits with their hashes, dates, authors, and comments.

#	Date	Auteur	Commentaire
0bbc0d3e	18/01/2021 14:34	Jérémy LIOT	[Dev] Changements sur la serialisation pour sérialiser une salle avec le monstre dedans
28d09af3	17/01/2021 22:42	Alban Niemaz	[MODIF] Nettoyage du code
f93d7458	17/01/2021 21:32	Jérémy LIOT	Merge branch 'master' of https://forge.clermont-universite.fr/git/projet24_jfx_2020_2021
295687e8	17/01/2021 16:39	Jérémy LIOT	Reglement des conflits
			[Dev] Changement de la serialisation pour qu'elle soit entièrement contenue dans la classe Manager.

Nous avons plusieurs branches et nous avons tout 2 utiliser les fonctionnalités push et merge afin d'actualiser le master et de partager nos avancées.

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

```
private final static int NBSALLE = 21;
```

Le nombre de salles de notre jeu est statique.

Je sais utiliser les collections.

```
private int numSalle;
private transient ObservableList<Carte> deck;
private Manager lemanager = Manager.getInstance();
private String image;
Random rand = new Random();
```

Notre joueur possède une ObservableList de cartes.

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

```
<ListView fx:id="deckListView" GridPane.columnIndex="0" GridPane.rowIndex="0" orientation="HORIZONTAL"
onMouseClicked="#handleMouseClicked" minWidth="950">
```

Nous utilisons une ListView pour afficher les différentes cartes.

Je sais utiliser les lambda-expression.

```
Timeline delai = new Timeline(
    new KeyFrame(Duration.seconds(2), event -> {
        try {
            attaquePlusTard(m, val);
        } catch (IOException e) {
            e.printStackTrace();
        }
    })
);
```

Je sais utiliser les listes observables de JavaFX.

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