

Preuves des compétences attendus :

DOCUMENTATION

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

Voir diagramme de classe.

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

Voir diagramme de classe.

Je sais documenter mon code et en générer la documentation.

Voir code.

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

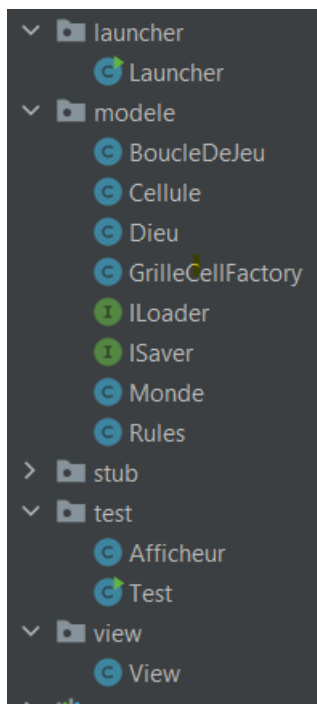
Voir contexte.

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

Voir le fichiers diagramme et sélectionner cas d'utilisation.

CODE

Je maîtrise les règles de nommage Java.



```
public class Cellule {
    private int nextTimeStatus; // si c'est -1 la cellule doit mourir

    private BooleanProperty alive = new SimpleBooleanProperty();
    public boolean getAlive(){ return alive.get(); }
    public void setAlive(boolean valeur){ alive.set(valeur);}
    public BooleanProperty aliveProperty(){ return alive; }

    private int x;
    private int y;
}
```

```
public void update(){
    if(nextTimeStatus == 1) setAlive(true);
    else if(nextTimeStatus == -1) setAlive(false);
}
```

Voir le code

Je sais binder bidirectionnellement deux propriétés JavaFX.

```
@Override
public void initialize(URL location, ResourceBundle resources){
    //Liste des enfants des Hbox des règles
    ObservableList<Node> bornChildren = bornHbox.getChildren();
    ObservableList<Node> surviveChildren = surviveHbox.getChildren();

    CheckBox box;
    for(int i=0;i<9;i++){ //boucle de création des checkbox et du binding
        box = new CheckBox(String.valueOf(i)); //initialisation d'une nouvelle checkbox avec le texte i.
        box.selectedProperty().bindBidirectional(Dieu.rules.bornRulesProperty(i)); //binding de la checkbox avec le booléen de règle
        bornChildren.add(box); //ajout de la checkbox dans le hbox
    }
}
```

Je sais binder unidirectionnellement deux propriétés JavaFX.

```
hauteurSpinner.valueProperty().addListener((caller,oldValue,newValue) ->{
    Dieu.monde.generer();
    createGrid(plateau);
});
Dieu.monde.tailleYProperty().bind(hauteurSpinner.valueProperty());
```

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

voir code.

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

```
<HBox>
    <Text text="Préréglages des règles: " />
    <ComboBox fx:id="presetsRules" onAction="#switchRules" />
</HBox>

<HBox>
    <Text text="Vitesse: " />
    <Slider fx:id="sliderTime" blockIncrement="10" max="1000" min="50" rotate="180" />
</HBox>

<HBox>
    <VBox>
        <Text text="Hauteur" />
        <Spinner fx:id="hauteurSpinner" initialValue="30" max="1000" min="3" editable="true" />
    </VBox>
    <VBox>
        <Text text="Largeur" />
        <Spinner fx:id="largeurSpinner" initialValue="30" max="1000" min="3" editable="true" />
    </VBox>
</HBox>
```

voir code.

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

```
public class GrilleCellFactory {
    public static Cellule[][] createCellGrid(int sizeX, int sizeY) {
        Cellule[][] grille = new Cellule[sizeX][sizeY];
        for(int x=0; x<sizeX; x++){
            for(int y=0; y<sizeY; y++){
                grille[x][y] = new Cellule(x, y);
            }
        }

        return grille;
    }
}
```

Je sais éviter la duplication de code.

voir code ou diagramme de classe

Je sais hiérarchiser mes classes pour spécialiser leur comportement.

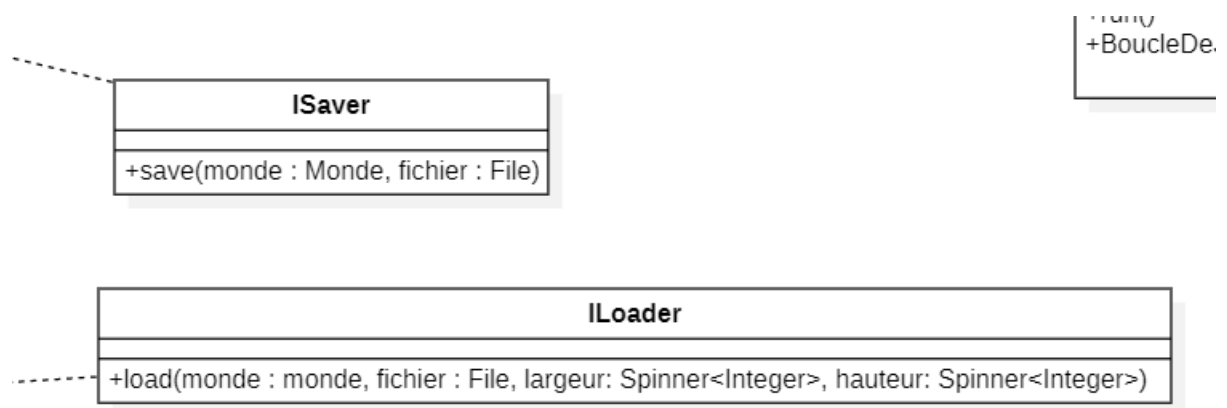
Je sais intercepter des évènements en provenance de la fenêtre JavaFX.

```
primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {  
    @Override  
    public void handle(WindowEvent event) { System.exit( status: 0); }  
});
```

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



Voir le diagramme de classe.




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.

 **Conway Game of Life** 
Project ID: 2028 [Leave project](#)

  Star 0  Fork 0

 48 Commits  1 Branch  0 Tags  922 KB Files  922 KB Storage

Arthur Lashermes Mathis Ribémont

-  **modification du changement de motif**
maribemont authored 17 minutes ago
-  **Merge branch 'main' of https://gitlab.iut-clermont.uca.fr/maribemont/conway-game-of-life into main**
maribemont authored 1 hour ago
-  **+modification taille grille**
maribemont authored 1 hour ago
-  **[code] CSS transformation visuelle de checkbox vers case**
arlasherme authored 1 hour ago
-  **+préréglages des règles, +bouton pour vide la grille**
maribemont authored 2 hours ago
-  **fonction createGrid dans vue, inversion slider de vitesse**
maribemont authored 3 hours ago
-  **[code] ajout du css**
arlasherme authored 4 hours ago

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

```
package modele;

import ...

public class Dieu {
    public static Monde monde;
    public static Rules rules;
```

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

```
<HBox>
    <Text text="Préréglages des règles: " />
    <ComboBox fx:id="presetsRules" onAction="#switchRules"/>
</HBox>
```

```
<ListView fx:id="TypeDeMonde" onMouseClicked="#switchMonde"/>
```

```
<HBox>
    <Text text="Vitesse: " />
    <Slider fx:id="sliderTime" blockIncrement="10" max="1000" min="50" rotate="180"/>
</HBox>
```

```
<VBox>
    <Text text="Largeur" />
    <Spinner fx:id="LargeurSpinner" initialValue="30" max="1000" min="3" editable="true"/>
</VBox>
```

Je sais utiliser les lambda-expression.

```
hauteurSpinner.valueProperty().addListener((caller, oldValue, newValue) ->{
    Dieu.monde.generer();
    createGrid(plateau);
});
Dieu.monde.tailleYProperty().bind(hauteurSpinner.valueProperty());
```

Je sais utiliser les listes observables de JavaFX.

Non mais des tableaux de propriété :

```
public class Rules {
    private SimpleBooleanProperty[] BornRules; //tableau de propriété booléen
    public boolean getBornRules(int i){ return BornRules[i].get(); }
    public void setBornRules(int i,boolean valeur){ BornRules[i].set(valeur);}
    public final BooleanProperty bornRulesProperty(int i) { return BornRules[i]; };
```

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.

```
Scene scene = new Scene(root, 1500, 1500);
scene.getStylesheets().add(getClass().getResource("/css/application.css").toExternalForm());
```

```
.check-box .box {

    -fx-background-color: white;
    -fx-border-color: grey;
    -fx-border-radius: 3px;

}

.check-box:selected .mark {

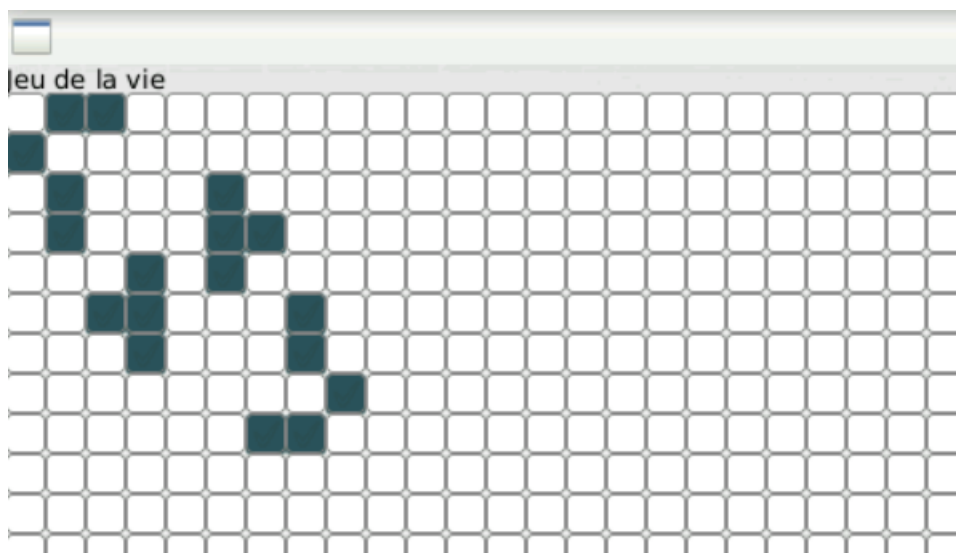
    -fx-background-color: #2A5058;

}

.check-box:selected .box {

    -fx-background-color: #2A5058;

}
```



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

Non.

Je sais développer un jeu en JavaFX en utilisant FXML.

Voir vidéo.

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

```
public void ajouterObservateur(Object a) { observateur.add(a); }

@Override
public void run() {
    int cpt = 0;
    while(true){
        if(BoucleDeJeu.getPlayed() == true){
            dieu.evolution();
            dieu.updateCells();
        }
        try { //ne pas mettre dans la boucle, sinon ça ne tourne pas
            Thread.sleep(getTime());
        } catch (Exception e){

        }
        //System.out.println(cpt = cpt+1);
    }
}
```