POINT Allan
BREUIL Yohann

Groupe 3-4

Bilan de compétences - Projet JavaFX

Documentation :

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

Cf. docs/uml/classes.mdj

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

Cf. docs/uml/classes.mdj

Je sais documenter mon code et en générer la documentation.

Cf. docs/javadoc/*

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

Cf. docs/contexte.pdf

Code :

Je maîtrise les règles de nommage Java.

Attributs : visibilité type nom

Constantes : visibilité *final* type NOM

Méthode : visibilité type-retour nomFonctionEnCamelCase (paramètres) {
traitements
}

Je sais binder unidirectionnellement deux propriétés JavaFX.

```
// Binding unidirectionel des informations à afficher (dimension de la grille et numéro de la génération
numTour.textProperty().bind(((ActualiseurTourUnParUn)manager.getActualiseurTour()).cptTourProperty().asString());
nbColGame.textProperty().bind(manager.getActualiseurCellule().getArbitre().getPlateau().coloneProperty().asString());
nbRowGame.textProperty().bind(manager.getActualiseurCellule().getArbitre().getPlateau().ligneProperty().asString());
```

code/jeu de la vie/src/views/VueJeu.java (lignes 114 à 117)

Je sais binder bidirectionnellement deux propriétés JavaFX.

```java
// Binding bidirectionel entre les regles de la vue et celles du model
regleChoiceBox.valueProperty().bindBidirectional(manager.getChangeurRegle().regleEnCoursProperty());

// Binding bidirectionel entre la ligne de la vue et celle du model
rowGame.getValueFactory().valueProperty().bindBidirectional((Property) manager.getActualiseurCellule().getArbitre().getPlateau().ligneProperty());

//Binding bidirectionel entre la colone du model et celle de la vue
colGame.getValueFactory().valueProperty().bindBidirectional((Property) manager.getActualiseurCellule().getArbitre().getPlateau().coloneProperty());
```

code/jeu de la vie/src/views/VueJeu.java (lignes 93 à 100)

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

Qualité de lecture :

- Attributs
- Constructeur
- Accesseurs
- API publique
- Méthodes internes

```java
public class Position {

    /** Position x */
    private int x;

    /** Position y */
    private int y;

    /**...*/
    public Position(int x, int y){
        setX(x);
        setY(y);
    }

    /**...*/
    public int getX() { return x; }

    /**...*/
    public void setX(int valeur) { x = valeur; }

    /**...*/
    public int getY() { return y; }

    /**...*/
    public void setY(int valeur) throws IllegalArgumentException{
        y = valeur;
    }

    /**...*/
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Position position = (Position) o;
        return x == position.x && y == position.y;
    }
```

code/jeu de la vie/src/model/cellule/Position.java

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

```
// Binding bidirectionel entre les regles de la vue et celles du model
regleChoiceBox.valueProperty().bindBidirectional(manager.getChangeurRegle().regleEnCoursProperty());

// Binding bidirectionel entre la ligne de la vue et celle du model
rowGame.getValueFactory().valueProperty().bindBidirectional((Property) manager.getActualiseurCellule().getArbitre().getPlateau().ligneProperty());

//Binding bidirectionel entre la colone du model et celle de la vue
colGame.getValueFactory().valueProperty().bindBidirectional((Property) manager.getActualiseurCellule().getArbitre().getPlateau().coloneProperty());
```

code/jeu de la vie/src/views/VueJeu.java (lignes 93 à 100)

Je sais éviter la duplication de code.

```
/**
 * Change l'actualiseur en fonction des règles
 * @param plateau Plateau actuel du jeu
 * @return Un ActualiseurCellule avec le bon arbitre
 * @see model.arbitre.Arbitre
 * @see ActualiseurCellule
 */
public ActualiseurCellule changerRegle(Plateau plateau) {
    switch (getRegleEnCours()){
        case REGLE_KILLER -> {
            return new ActualiseurEtatCellule(new ArbitreKiller(plateau));
        }
        default -> {
            return new ActualiseurEtatCellule(new ArbitreConwayStyle(plateau));
        }
    }
}
```

code/jeu de la vie/src/model/ChangeurRegle.java (lignes 26 à 42)

```
/**
 * Actualiser l'état des cellules
 */
private void deleguerChangementCellule() {
    CellulesVivantes reference = getActualiseurCellule().getArbitre().getPlateau().getCellulesVivantes().clone();
    for (int y = 0; y<actualiseurCellule.getArbitre().getPlateau().getLigne(); ++y){
        for(int x = 0; x<actualiseurCellule.getArbitre().getPlateau().getColone(); ++x){
            actualiseurCellule.changerCellule(x, y, reference);
        }
    }
}
```

code/jeu de la vie/src/model/Manager.java (lignes 62 à 72)

Je sais hiérarchiser mes classes pour spécialiser leur comportement.

```
public class ArbitreConwayStyle extends Arbitre{
```

code/jeu de la vie/src/model/arbitre/ArbitreConwayStyle.java (ligne 12)

```
/**
 *
 * @param x Coordonée x de la cellule à checker
 * @param y Coordonée y de la cellule à checker
 * @param reference Toutes les cellules qui était vivantes au début du tour et qui servent donc de references
 * @return L'état de la cellule au prochain tours
 * @see CellState
 */
public abstract CellState verifierChangementCellules(int x, int y, CellulesVivantes reference);
```

code/jeu de la vie/src/model/arbitre/Arbitre.java (lignes 46 à 54)

```
/**
 *
 * @param x Coordonée x de la cellule à checker
 * @param y Coordonée y de la cellule à checker
 * @param reference Toutes les cellules qui était vivantes au début du tour et qui servent donc de references
 * @return L'état de la cellule au prohain tour
 */
@Override
public CellState verifierChangementCellules(int x, int y, CellulesVivantes reference) {
    if(verifierNaissance(x, y, reference)) {
        return CellState.BIRTH;
    }
    if(verifierMort(x, y, reference)) {
        return CellState.DIE;
    }
    return reference.getAt(x, y) != null ? CellState.LIVE : CellState.DIE;
}
```

code/jeu de la vie/src/model/arbitre/ArbitreConwayStyle.java (lignes 23 à 39)

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

```
rect.setOnMouseClicked((src)->manager.inverserEtatCellule(c));
```

code/jeu de la vie/src/views/VueJeu.java (ligne 64)

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

Classe	Responsabilité
Launcher	Démarrer l'application
ActualiseurEtatCellule	Modifier l'état d'une cellule
ActualiseurTourUnParUn	Modifier le nombre de tour
ArbitreConwayStyle	Appliquer les règles Conway
ArbitreKiller	Appliquer les règles Killer
BoucleDeJeu5FPS	Boucler à un intervalle
BoucleDeJeu30FPS	Boucler à un intervalle
CreateurCellule	Créer une cellule
Cellule	Décrire une cellule
Position	Décrire une position
Plateau	Décrire un plateau
CellulesVivantes	Stocker les cellules vivantes
ChangeurRegle	Changer la règle
CompteurDeCellules	Compter les cellules vivantes
Manager	Faire façade entre le Modèle et les Vues

Je sais utiliser à mon avantage le polymorphisme.

```
private IBoucleDeJeu boucleDeJeu;
```

```
boucleDeJeu = new BoucleDeJeu5FPS();
```

code/jeu de la vie/src/model/Manager.java (lignes 21 et 38)

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

Cf. https://gitlab.iut-clermont.uca.fr/alpoint/jeu-de-la-vie

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

```
private ActualiseurTour actualiseurTour;
private ActualiseurCellule actualiseurCellule;
private IBoucleDeJeu boucleDeJeu;
private ChangeurRegle changeurRegle;
private boolean jeuLance;
```

code/jeu de la vie/src/model/Manager.java (lignes 19 à 23)

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

Cf. code/jeu de la vie/rsrc/fxml/VueJeu.fxml

Je sais utiliser les lambda-expression.

```
changeurRegle.regleEnCoursProperty().addListener((src)->actualiserActualiseurCellule());
```

code/jeu de la vie/src/model/Manager.java (ligne 36)

Je sais utiliser les listes observables de JavaFX.

```
// Remplissage des valeurs de la choice box des regles
regleChoiceBox.setItems(FXCollections.observableArrayList(Regle.values()));
```

code/jeu de la vie/src/views/VueJeu.java (lignes 87 à 88)

Je sais développer un jeu en JavaFX en utilisant FXML.

Cf. code/jeu de la vie/rsrc/fxml/VueJeu.fxml

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

Cf. code/jeu de la vie/src/model/boucleDeJeu/*
Cf. code/jeu de la vie/src/model/boucleDeJeu/observer/*
Cf. code/jeu de la vie/src/model/Manager.java