Perret Louis G3-G4
Wissocq Maxime

Projet JavaFx : Bilan de compétences

Partie Documentation :

→Cf. documentation/Document_PokemonUltimate.pdf

- Je sais documenter mon code et en générer la documentation.
  → Cf. documentation/javadoc puis ouvrir dans un navigateur index.html

Partie Code :

- Je maîtrise les règles de nommage Java.
  → Méthode : visiblité typeRetour nomMéthode(){
  }

- Je sais binder unidirectionnellement deux propriétés JavaFX.

```
changeur.bind(manager.changeurProperty()); //binding unidirectionnel
compteur.bind(manager.compteurProperty()); //binding unidirectionnel
```

Cf. code/src/vues/controller/Fenetre.java

- Je sais binder bidirectionnellement deux propriétés JavaFX.
  → Nous n'avons pas eu besoin d'en utiliser, mais si nous avions dû en utiliser nous aurions utilisé la méthode bindBidirectional au lieu de bind.

- Je sais coder une classe Java en respectant des contraintes de qualité de lecture de code.
  → Nous avons respecté à chaque fois l'ordre suivant :
  - Déclaration des attributs
  - Constructeur
  - Méthodes
  - Getter & setter

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

```
nomJoueur.textProperty().bind(manager.getPokemonCourant().nomProperty());
pvJoueur.textProperty().bind(manager.getPokemonCourant().pvProperty().asString());
nomEnnemi.textProperty().bind(manager.getPokemonEnnemiCourant().nomProperty());
pvEnnemi.textProperty().bind(manager.getPokemonEnnemiCourant().pvProperty().asString());
```

Cf. code/src/vues/controller/FenetreCombat.java

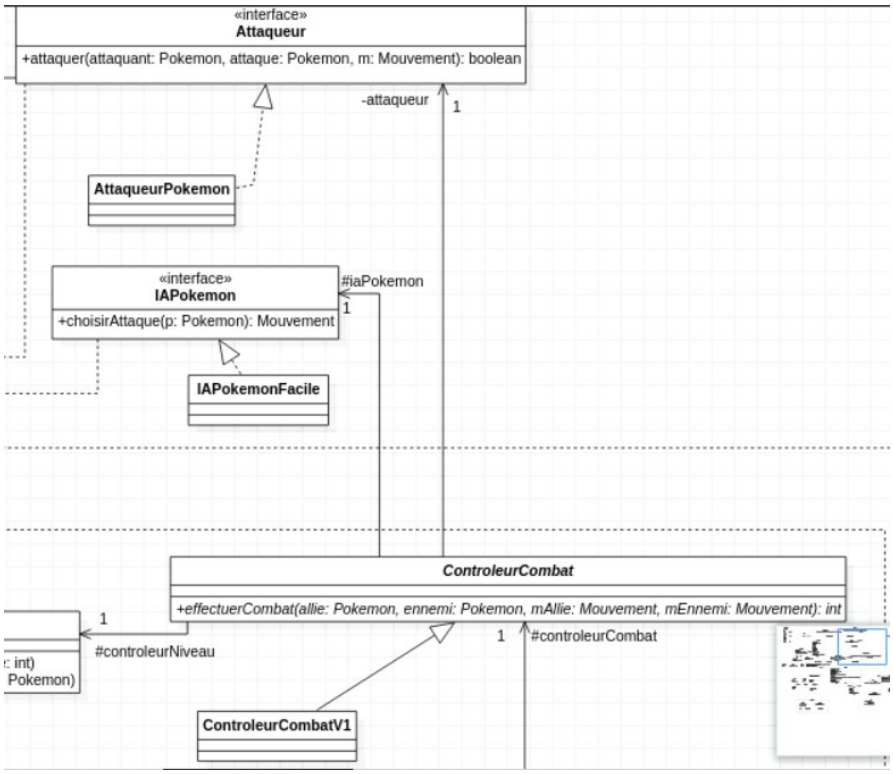
- Je sais définir une CellFactory fabriquant des cellules qui se mettent à jour au changement du modèle
  → Nous n'en avons pas utilisé au sein de notre projet car nous n'en avons pas ressenti le besoin

- Je sais éviter la duplication de code.
  → Voici un bon exemple :

```java
public int effectuerCombat(Pokemon allie, Pokemon ennemi, Mouvement mAllie){
    Mouvement mEnnemi = iaPokemon.choisirAttaque(ennemi);
    if(allie.getVitesse() >= ennemi.getVitesse()){ //si le pokemon du joueur est plus rapide
        if(attaqueur.attaquer(allie,ennemi,mAllie)) {
            controleurNiveau.gagnerExperience(allie, ennemi); //On lui fait gagner de l'expérience
            return 1; //le pokemon ennemi est ko
        }
        else{
            if(attaqueur.attaquer(ennemi,allie,mEnnemi)) {
                return 2; //le pokemon du joueur est ko
            }
        }
    }
    else{ //si le pokemon ennemi est plus rapide
        if(attaqueur.attaquer(ennemi,allie,mEnnemi)) {
            return 2; //le pokemon du joueur est ko
        }
        else{
            if(attaqueur.attaquer(allie,ennemi,mAllie)) {
                controleurNiveau.gagnerExperience(allie, ennemi); //On lui fait gagner de l'expérience
                return 1; //le pokemon ennemi est ko
            }
```

Cf. code/src/modele/combat/ControleurCombat.java

- Je sais hiérarchiser mes classes pour spécialiser leur comportement.
  → Voici un exemple :



Cf. documentation/diagrammesUML/diag-classes.mdj

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

```
public void choixPokemon(ActionEvent actionEvent) {
    if (actionEvent.getSource() == boutonbulb){
        manager.setPokemonCourant(listeStarter.get(0));
    }
    if (actionEvent.getSource() == boutonsalam){
        manager.setPokemonCourant(listeStarter.get(1));
    }
    if (actionEvent.getSource() == boutoncarap){
        manager.setPokemonCourant(listeStarter.get(2));
    }
    navigateur.lancerFenetreJeu();
}
```

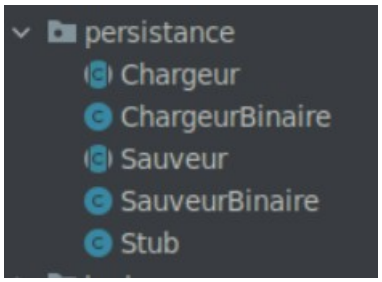
Cf. code/src/vues/controller/FenetreSelection.java

- Je sais maintenir, dans un projet, une responsabilité unique pour chacune de mes classes
  → Exemple avec le paquetage combat :

Attaqueur	Gère l'attaque d'un pokémon vers un autre
CalculCoefficient	Calcul un coefficient en fonction des types
ControleurNiveau	Gère le gain d'expérience et le niveau d'un pokémon
IAPokemon	Simule le choix d'attaque du pokémon ennemi
ControleurCombat	Contrôle les enchaînements d'un combat

Cf. documentation/Document_PokemonUltimate.pdf pour voir la responsabilité des autres classes

- Je sais gérer la persistance de mon modèle.



Cf. code/src/persistance

- Je sais utiliser à mon avantage le polymorphisme
  → Exemple concret pour la persistance de notre application

```
Chargeur chargeur = new ChargeurBinaire(cheminFichier);
manager = chargeur.charger();
```

Cf. code/src/launch/launcher.java

```
public void quitterJeu() throws Exception {
    Sauveur sauveur = new SauveurBinaire(cheminFichier);
    if (!sauveur.sauver(manager)) {
        throw new Exception("Problème dans la sauvegarde des données");
    }
    primaryStage.close();
}
```

Cf.code/src/vues/controller/Navigateur.java

- Je sais utiliser GIT pour travailler avec mon binôme sur le projet.
  → Voir notre repository Gitlab :
  https://gitlab.iut-clermont.uca.fr/mawissocq/projet_perret_wissocq

- Je sais utiliser le type statique adéquat pour mes attributs ou variables.
  → Exemple dans notre launcher : Cela nous permet d'avoir un manager et un stage global à l'application et réutilisable par nos fenêtres.

```
private static Manager manager;
private static Stage primaryStage;
```

Cf. code/launch/launcher.java

- Je sais utiliser les différents composants complexes (listes, combo…) que me propose JavaFX.
  → Nous n'avons pas utilisé de composants complexes car nous n'en avons pas ressenti le besoin

- Je sais utiliser les lambda-expression.
  → Voici un exemple où cela nous permet d'ajouter facilement un listener.

```
compteur.addListener((observableValue, number, newValue) -> {
    a = afficheurPokemon.affiche(manager.getPokemonCourant(),manager.getPokemonCourant().getPosition());
    imageView.setImage(a.getImage());
    imageView.setTranslateX(a.getX());
    imageView.setTranslateY(a.getY());

}
);
```

Cf. code/src/vues/controller/Fenetre.java

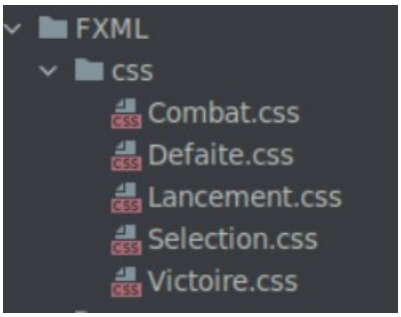
- Je sais utiliser les listes observables de JavaFX.
  → Nous n'avons pas utilisé de listes observables car nous n'en avons pas ressenti le besoin

- Je sais utiliser un convertisseur lors d'un bind entre deux propriétés JavaFX.
  → Pour passer d'un IntegerProperty à un StringProperty

```
pvJoueur.textProperty().bind(manager.getPokemonCourant().pvProperty().asString());
nomEnnemi.textProperty().bind(manager.getPokemonEnnemiCourant().nomProperty());
pvEnnemi.textProperty().bind(manager.getPokemonEnnemiCourant().pvProperty().asString());
```

Cf. code/src/vues/controller/FenetreCombat.java

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

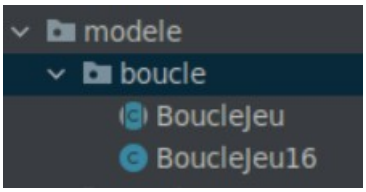


Cf. code/Ressources/FXML/css

- Je sais utiliser un formateur lors d'un bind entre deux propriétés JavaFX
  → Nous n'avons pas utilisé de formateur car nous n'en avons pas ressenti le besoin

- Je sais développer un jeu en JavaFX en utilisant FXML.
  → Voir la vidéo : documentation/video_presentation_Pokemon_Ultimate.mp4 (vous pouvez aussi lancer le jeu. Plaisir garanti !)

- Je sais intégrer, à bon escient, dans mon jeu, une boucle temporelle observable.
  → Elle nous permet d'actualiser la position (et l'image) du pokemon sur la fenêtres



Cf. code/src/modele/boucle