BUT INFORMATIQUE 2

SAÉ 3.01 – Chasse au monstre

RAPPORT D'ANALYSE

RÉALISÉ PAR

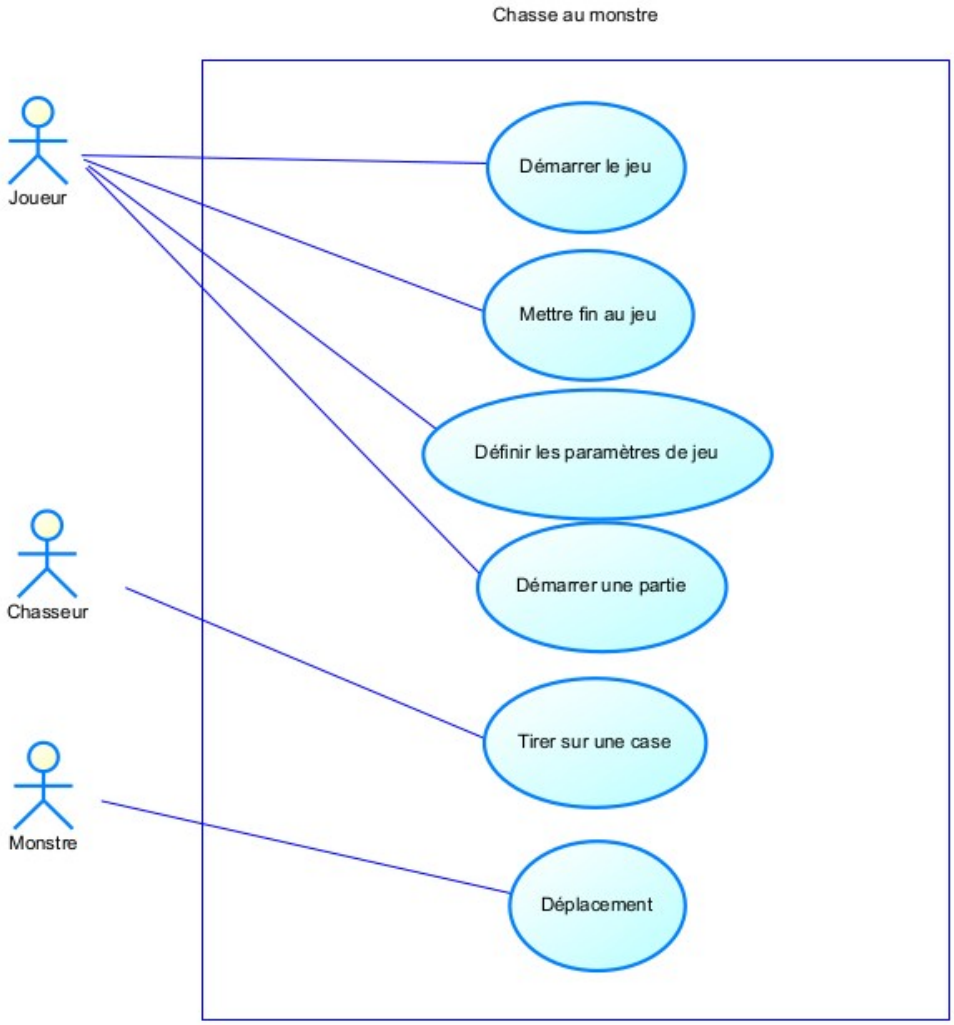
Groupe I4

HAMITOUCHE QUENTIN
DEWISME GUILLAUME
CANCEL PAUL
FRANCKELEMON CLÉMENT

13 octobre 2023

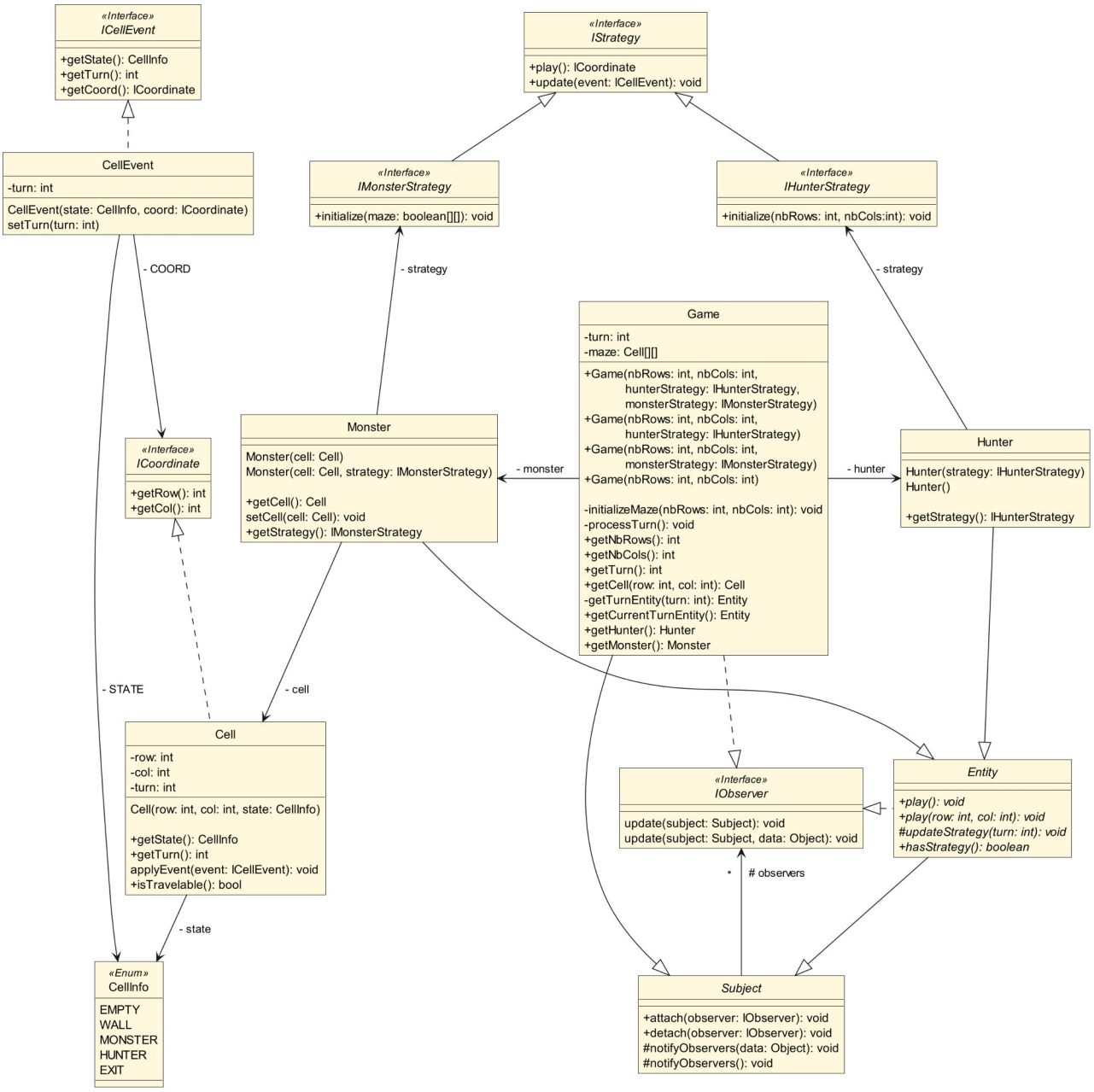
TABLE DES MATIÈRES

I/ DIAGRAMME DE CAS D'UTILISATION



II/ DIAGRAMME DE CLASSES

1/ Diagramme de classes



2/ Nos choix d'implémentation

Au début du jeu, nous créons une instance de la classe `Game` en spécifiant le nombre de lignes et de colonnes, ainsi que les éventuelles stratégies du monstre et du chasseur.

1.Nous initialisons un labyrinthe composé de cellules.

2.Puis, nous initialisons les stratégies si elles sont données.

3.Ensuite, nous créons et associons le monstre et le chasseur à notre objet `Game`.

4.Enfin, nous initialisons le compteur de tours à zéro.

À ce stade, le jeu est prêt. Notre objet `Game` attend maintenant d'être informé de l'entité à laquelle revient le tour.

En pratique, les **entités** agissent comme des **contrôleurs**, transmettant les actions de l'utilisateur au **modèle** (`Game`) via un objet `CellEvent`.

Le modèle vérifie alors si c'est bien le tour de l'entité en question, exécute l'événement `CellEvent`, puis informe les **observateurs** associés.

En ce qui concerne la visibilité et l'organisation des paquetages, tout a été agencé de ce que nous croyons être la manière la plus optimale possible :

Paquetage `model.maze` :

- Classe `Monster` :
  - Le constructeur (*défaut*) est appelé lorsqu'une instance de la classe est créée dans la classe `Game`.
  - La méthode `setCell` (*défaut*) est appelée dans la classe Game lorsque le tour est validé.
- Classe `Hunter` :
  - Le constructeur (*défaut*) est appelé lorsqu'une instance de la classe est créée dans la classe `Game`.

•Classe `Cell` :

- •Le constructeur (*défaut*) est appelé lorsqu'une instance de la classe est créée dans la classe `Game`.
- •La méthode `applyEvent` (*défaut*) est appelée dans la classe `Game` lorsque le tour est validé.

•Classe `Game` :

- •La méthode `initializeMaze` (*privée*) est appelée dans le constructeur pour séparer l'algorithme de remplissage du reste du code.
- •La méthode `getTurnEntity` (*privée*) est utilisée en interne par la méthode `getCurrentTurnEntity`.
- •La méthode `processTurn` (*privée*) est utilisée en interne dans la méthode `update` par `Entity.notifyObservers` avec un paramètre de type `ICellEvent`. Elle appelle `notifyObservers` si le tour est valide avec ce même `ICellEvent`. Enfin, elle appelle `Cell.applyEvent` sur la cellule concernée.

•Classe `Entity` :

- •La méthode `updateStrategy` (*protégée*) est appelée dans les méthodes `update` de ses classes filles.
- •La méthode `play` (*publique*), sans paramètre, est appelée par la vue. Les coordonnées de la case à jouer sont récupérée à partir de la méthode `play` de la stratégie encapsulée.
- •La méthode `play` (*publique*), avec paramètres, est appelée par la vue, où les coordonnées sont fournies par le joueur.
- •Les méthodes `play` invoquent `notifyObservers` en passant un paramètre de type `ICellEvent` qu'elles auront préalablement forgé.

•Classe `Subject` :

- •La méthode `notifyObservers` (*privée*) est appelée par ses classes filles lorsque cela est nécessaire.

•Classe `CellEvent` :

- •Le constructeur (*défaut*) est utilisé par `Entity.play` pour créer un objet `CellEvent`, mais il n'a pas accès au numéro de tour, car il n'existe aucun lien direct entre `Entity` et `Game`.
- •La méthode `setTurn` (*défaut*) est appelée dans la méthode `Game.processTurn`.

•Interface `IObserver` :

- •Les méthodes `update` sont par défaut pour empêcher tout appel depuis les vues.

En résumé, les classes situées en dehors du paquetage `model.maze` sont limitées à l'instanciation de la classe `Game`, qui fonctionne en tant que modèle.

Pour interagir avec le modèle, elles doivent nécessairement passer par les classes `Monster` ou `Hunter`, qui remplissent le rôle de contrôleurs, en utilisant exclusivement leurs méthodes `play(int, int)`.

Toutefois, il reste possible de casser le jeu en appelant la méthode `detach` depuis une classe fille de `Subject`, par exemple avec `game.detach(game.getHunter())`.

Nous condamnons **fermement** tout appel à la méthode `detatch` de cet ordre.

III/ FICHES DESCRIPTIVES

1/ Déplacement du monstre

Système : Le jeu de la chasse au monstre
Cas d'utilisation : Déplacement du monstre
Acteur principal : Le monstre
Acteur secondaire : /
Préconditions : Être dans une partie et être sur le tour du monstre
Garantie en cas de succès : Le monstre se déplace
Garantie minimale : Rien ne se passe

Scénario nominal :

1. Lancement de la fonctionnalité "Déplacement du monstre".
2. Le système affiche les possibilités de déplacement et demande au monstre d'en choisir une ($\leftarrow$, $\uparrow$, $\downarrow$, $\rightarrow$)
3. Le monstre choisit une direction
4. Le système enregistre le déplacement et enregistre le numéro de tour dans la case

Scénarios alternatifs :

A. Si le monstre entre une direction invalide lors de l'étape 3 (mur/bordure), alors :

- 4(A) : Retour à l'étape 3 du scénario nominal

B. Si le monstre se déplace vers la sortie lors de l'étape 3, alors :

- 4(B) : Le système affiche l'écran de victoire du monstre. Le système met fin à la partie courante

2/ Le chasseur tire

Système : Le jeu de la chasse au monstre
Cas d'utilisation: Le Chasseur tire
Acteur principal: Le Chasseur
Acteur secondaire: /
Préconditions: Être dans une partie et être au tour du chasseur
Garantie en cas de succès: le chasseur tire
Garantie minimale: Rien ne se passe

Scénario nominal:

1. Lancement de la fonctionnalité "Le chasseur tire"
2. Le système affiche le plateau
3. Le chasseur sélectionne la case sur laquelle il souhaite tirer
4. Le système vérifie s'il y a un gagnant et affiche la case sélectionnée

Scénarios alternatifs:

A. Si le chasseur sélectionne la case où se trouve le monstre, alors à l'étape 4:

- 4(A): Le système affiche un écran de victoire pour le chasseur puis le système met fin à la partie courante

3/ Lancer une partie

Système : Le jeu de la chasse au monstre
Cas d'utilisation: Lancer une partie
Acteur principal: Le Joueur
Acteur secondaire: /
Préconditions: Le jeu est ouvert
Garantie en cas de succès: La partie est paramétrée puis la partie se lance
Garantie minimale: Rien ne se passe

Scénario nominal:

1. Le joueur clique sur la fonctionnalité "Lancer une partie"
2. Le système affiche une liste de choix à sélectionner (Joueur vs Joueur, Joueur vs IA, IA vs IA)
3. Le joueur choisit Joueur vs Joueur ou Joueur vs IA
4. Le système demande au joueur de choisir un rôle
5. Le joueur choisit
6. Le système enregistre le choix et lance la partie

Scénarios alternatifs:

A. Si le joueur sélectionne IA vs IA à l'étape 3, alors :

- 4(A): Le système lance la partie