MATH EN INFO

TRAVAUX PRATIQUES N° 3

Voisins

1 Objectifs

Dans le sujet précédent nous avons appris à notre IA à demander et à interpréter la carte du jeu. Afin de pouvoir utiliser cette carte lors du déplacement de notre personnage, nous avons besoin que la carte adopte une structure de graphe (afin de pouvoir utiliser nos différents algorithmes de calcul de plus court chemin par exemple). Actuellement, cette structure de graphe n'est que partielle, nous ne disposons que des sommets : les cases, mais nous n'avons pas encore modélisé les arêtes.

2 Modification de la classe Coordonnée

Pour déterminer les voisins d'une case, il nous faut en déterminer les coordonnées. Pour cela nous allons apprendre à la classe coordonnée à calculer les coordonnées des 4 cases situées autour d'elle.

1. **A FAIRE : Dans le package "metier", créer une énumération "TypeMouvement" dont les éléments seront : TOP, BOTTOM, LEFT et RIGHT.**

De façon générale, une case de notre carte possède 4 voisins, 1 voisin pour chacun des 4 mouvements possibles. Nous parlerons donc par la suite du voisin haut, du voisin bas, du voisin droit... d'une case.

2. **A FAIRE : Dans la classe Coordonnée, implémenter une méthode :**

   public Coordonnee getVoisin(TypeMouvement mouvement)

   qui renvoie les coordonnées de la case obtenue en effectuant le mouvement donné en paramètre à partir de la case en cours.
3. **A FAIRE : Dans la classe coordonnée, implémenter une méthode :**

   public TypeMouvement getMouvementPourAller(Coordonnee destination)

   qui renvoie le mouvement à réaliser pour passer de la Coordonnee courante à la Coordonnee destination. La méthode renverra null si aucun mouvement ne permet cela.

Pour tester cette méthode, nous allons la faire passer un test unitaire.

4. **A FAIRE : Créez un nouveau test unitaire pour la classe Coordonnee. Pour cela, faites un clic droit sur la classe puis sélectionnez Outils>Créer/Mettre à jour tests (Tools>Create/Update Tests).**
5. **Remplacez le contenu de la classe de test par celui du fichier "CoordonneeTest" présent dans le répertoire "TP3/TestsUnitaires" sur le commun.**
6. **A FAIRE : Exécuter, dans Netbeans, le test unitaire "CoordonneeTest" et corriger le code de la méthode getVoisin si nécessaire.**

3 Modification de la classe Case

Maintenant que l'on sait trouver les coordonnées des voisins d'une case, nous allons pouvoir ajouter cette notion de voisins à nos cases (et donc les arêtes de notre graphe).

7. **A FAIRE : Ajouter un attribut "voisins" de type ArrayList<Case> à la classe "Case". (Bien penser à initialiser ce nouvel attribut dans le constructeur).**
8. **A FAIRE : Ajouter un getter (getVoisins) pour ce nouvel attribut ainsi qu'une méthode :**

   public void ajouterVoisin(Case voisin)

   qui ajoute la case donnée en paramètre à la liste des voisins.

4 Modification de la classe Carte et tests

Il faut maintenant que la carte s'occupe d'indiquer aux différentes cases quelles sont leurs voisins en utilisant les nouvelles méthodes de la classe "Coordonnée".

9. **A FAIRE : Après l'avoir compris, ajouter le code suivant au constructeur de la carte (impérativement après la double boucle générant les différentes cases de la carte) :**

```
//Gestions des voisins
for(int i=0;i<this.taille;i++) {
    for(int j=0;j<this.taille;j++) {
        Coordonnee cooCase = new Coordonnee(i,j);
        for(TypeMouvement mouvement : TypeMouvement.values()) {
            Coordonnee cooVoisin = cooCase.getVoisin(mouvement);
            if(this.cases.get(cooVoisin) != null) {
                this.cases.get(cooCase).ajouterVoisin(this.cases.get(
                    cooVoisin));
            }
        }
    }
}
```

Afin de pouvoir tester ce que nous venons de faire, nous souhaitons faire afficher les coordonnées des voisins des cases (3,3) et (6,4). Pour cela, nous allons ajouter une méthode pour afficher proprement une coordonnée dans la console.

10. **A FAIRE : Ajouter une méthode "toString" pertinente à la classe "Coordonnée".**

Maintenant que ceci est fait, nous pouvons faire notre test.

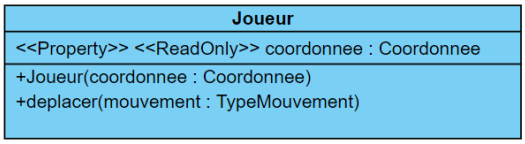
11. **A FAIRE : Ajouter à la fin du constructeur de la classe "Carte" le code nécessaire pour faire afficher dans la console les voisins de cases de coordonnées (0,0), (3,3) et (6,4). Vérifiez le résultat obtenu.**

Ce dernier code servant juste de test, **il doit être supprimé** dès que les résultats sont conformes à ceux obtenus à la main. De même, assurez-vous que la carte n'est plus affichée dans la console quand elle est créée.

5 Gestion du joueur

Maintenant que notre IA est capable d'obtenir entièrement la carte, il lui faut encore être capable de repérer la position du joueur dessus. Commençons par créer une classe servant à modéliser le joueur.

12. **A FAIRE : Dans le package "metier", créez et implémentez une classe "Joueur" respectant l'UML suivant :**



13. **A FAIRE : Créez un test unitaire pour la classe "Joueur" et remplacez son contenu par celui du fichier "JoueurTest.java" présent dans le répertoire "TP3/TestsUnitaires" sur TEAMS.**
14. **A FAIRE : Exécuter, dans Netbeans, le test unitaire "JoueurTest" et corriger le code de la classe "Joueur" si nécessaire.**

Maintenant que nous disposons d'une classe représentant le joueur, nous allons pouvoir permettre au module mémoire de le gérer.

15. **A FAIRE : Ajoutez un attribut "joueur" de type Joueur dans le module mémoire.**
16. **A FAIRE : Ajoutez et implémentez les trois méthodes suivantes dans le module mémoire :**
    - **public boolean hasJoueur() qui renvoie true si l'attribut joueur est null.**
    - **public Joueur getJoueur() qui renvoie l'attribut joueur.**
    - **public void genererJoueur(Coordonnee coordonnne) qui génère le joueur à partir des coordonnées données en paramètre.**

Il nous faut maintenant apprendre à notre IA à créer le joueur lors de la création de la carte. Pour cela, nous allons faire que la carte mémorise la position du point de départ (nous en aurons aussi besoin plus tard).

17. **A FAIRE : Ajoutez un attribut "CoordonneeDepart" de type "Coordonnee" à la classe "Carte" ainsi qu'un getter pour cet attribut.**
18. **A FAIRE : Modifiez la méthode "ajouterCase" de la classe "Carte" pour que si le caractère reçu en paramètre est 'D', l'attribut "CoordonneeDepart" prenne la valeur des coordonnées en cours.**

Il nous faut maintenant compléter notre module mémoire.

19. **A FAIRE : Modifiez la méthode "genererCarte" du Module de mémoire pour qu'après avoir généré la carte, elle appelle la méthode "genererJoueur".**
20. **A FAIRE : Faites afficher la coordonnée du joueur (à la fin de la méthode genererCarte par exemple) et testez votre IA pour vérifier que la communication se passe bien.**

Il nous reste un dernier petit détail à régler pour que la gestion de la carte soit complète, il nous faut placer les escaliers devant la maison.

21. **A FAIRE : Ajoutez le cas 'S' à la fabrique des Objets pour lequel la fabrique créera un nouvel escalier.**

22. **A FAIRE : Ajoutez à la fin du constructeur de carte le code nécessaire pour changer l'objet contenu dans les trois cases situées en dessous de la case de départ (on utilisera la méthode getVoisin de la classe coordonnée et la fabrique d'objet).**
23. **A FAIRE : Vérifiez que l'affichage de la carte prend bien en compte les escaliers.**