Projet d'Algorithmique et de Développement Logiciel

Les Aventuriers du Rail

Incrément 7

Séances 21 à 24

Sommaire

1 Préambule

L'incrément 7 du Projet Informatique s'appuie sur la version électronique produite pendant l'incrément 6 du jeu des Aventuriers du Rail (ADR).

L'objectif principal du septième incrément est d'implémenter une nouvelle version de la pioche de cartes wagon : suite à la pioche d'une carte wagon (voir incrément 6), la main du joueur (i.e. la liste de cartes wagon en sa possession) est mise à jour et contient la carte précédemment piochée.

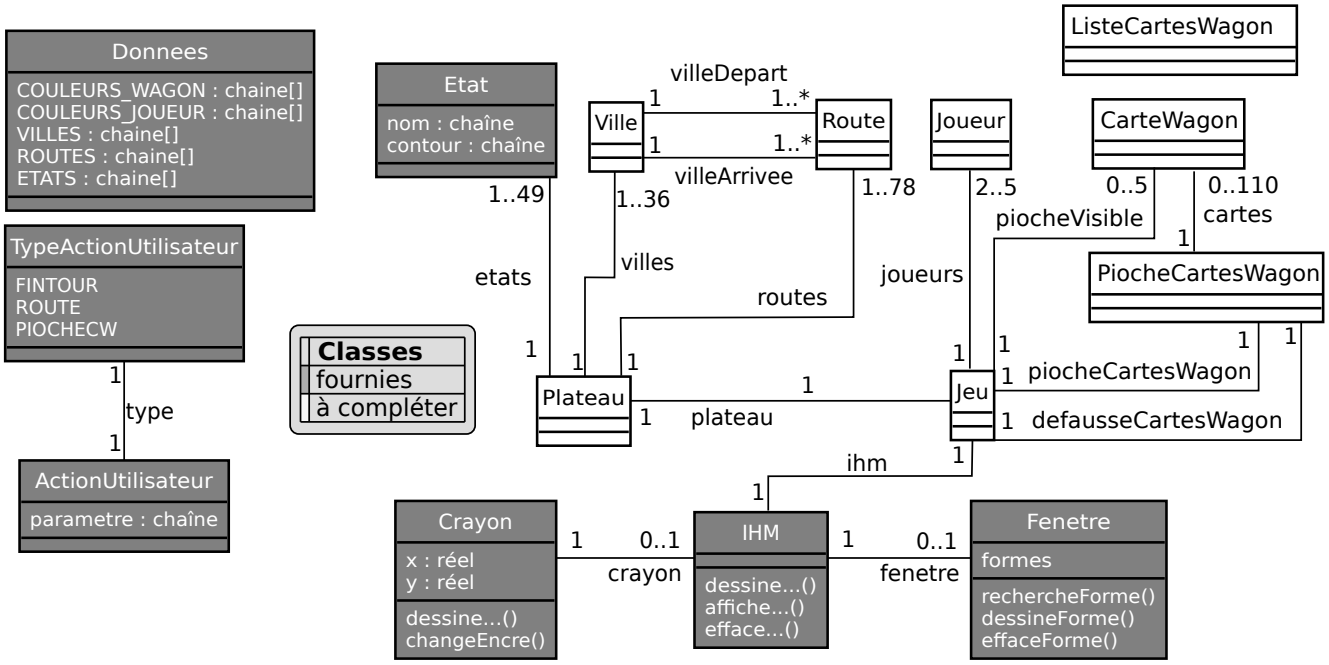

Figure 1: : Classes identifiées dans le jeu des ADR - Incrément 7

Pour implémenter cette fonctionnalité, il est nécessaire de définir une nouvelle classe **ListeCartesWagon** et de s'appuyer sur les classes **PiocheCartesWagon** et **Jeu** précédemment implémentées (voir le diagramme de classes UML simplifié et incomplet en figure 1).

2 Évaluation du travail

Le septième incrément consolide vos compétences en programmation Java en général et en particulier sur la notion de ***Liste***.

L'évaluation des compétences liées à ce septième incrément est réalisée par l'évaluation sur machine n°3 du 03/01/2022.

3 Travail demandé

Dans le jeu des ADR, les joueurs « piochent » des cartes wagon pour constituer leur main. La pioche de cartes wagon est proposée sous deux formes : une pioche « cachée » et une pioche « visible » de cinq cartes. La première version de « piocher une carte » (implémentée dans l'incrément 6) consiste à (1) afficher les pioches cachée et visible et la défausse, (2) détecter la sélection d'une carte wagon sur une des pioches et (3) retirer la carte wagon sélectionnée de la pioche correspondante.

Remarque La description précédente correspond à la pioche de cartes wagon de l'incrément 6.

Remarque Le transfert de la carte de la pioche (visible ou cachée) vers la main du joueur est l'objectif de l'incrément 7. Les cartes « piochées » sont intégrées dans la main du joueur et la main du joueur est affichée.

3.1 Préparation de l'environnement

Importez l'incrément 7 dans votre projet existant (incréments 1 à 6) en suivant la procédure décrite dans la documentation du logiciel DevCube.

3.2 La classe ListeCartesWagon

La classe **ListeCartesWagon** est l'implémentation de la liste de cartes qui constitue la main du joueur. Pour faciliter la manipulation des cartes de la main du joueur, on utilise le type abstrait Liste qui dispose d'attributs et de méthodes que l'on retrouve dans toute implémentation de liste.

Pour les besoins du jeu, nous définissons des méthodes supplémentaires qui sont décrites dans la figure 2.

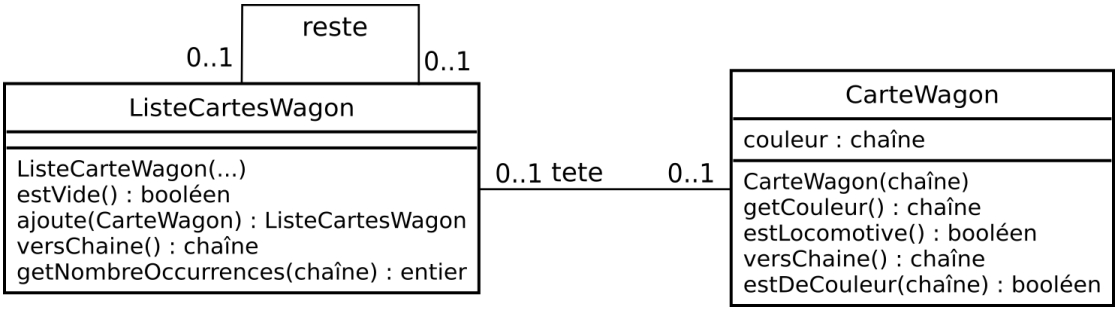

Figure 2: Diagramme de classe de la classe ListeCartesWagon

3.2.1 Définir la classe ListeCartesWagon

Travail à faire

Définir la classe **ListeCartesWagon** telle qu'elle est représentée dans la figure 2.

3.2.2 Création d'une liste et initialisation

Pour utiliser la liste de cartes de la main du joueur, il est nécessaire de créer un nouvel objet de type **ListeCartesWagon**.

Dans le cas de la classe **ListeCartesWagon**, on souhaite créer une liste à partir d'une tête et d'un reste passés en paramètre du constructeur.

Travail à faire

Écrire le constructeur `ListeCartesWagon()` de la classe **ListeCartesWagon**.
Exécuter la classe de test **TestListeCartesWagonInitialisation** pour vérifier l'implémentation du constructeur.

3.2.3 Méthodes

Travail à faire

Implémenter les accesseurs pour les attributs.
Écrire les quatre méthodes de la classe **ListeCartesWagon** (voir figure 2).
Exécuter les classes de test fournies pour valider l'implémentation de la classe **ListeCartesWagon**.

- TestListeCartesWagonVide
- TestListeCartesWagonAjout
- TestListeCartesWagonAffichage
- TestListeCartesWagonOccurrences

3.3 La main du joueur

La « main » d'un joueur est représentée par une liste de cartes wagon qui contient l'ensemble des cartes wagon du joueur. La définition du **Joueur** est donc modifiée pour intégrer la « main » et permettre d'y déposer des cartes wagon.

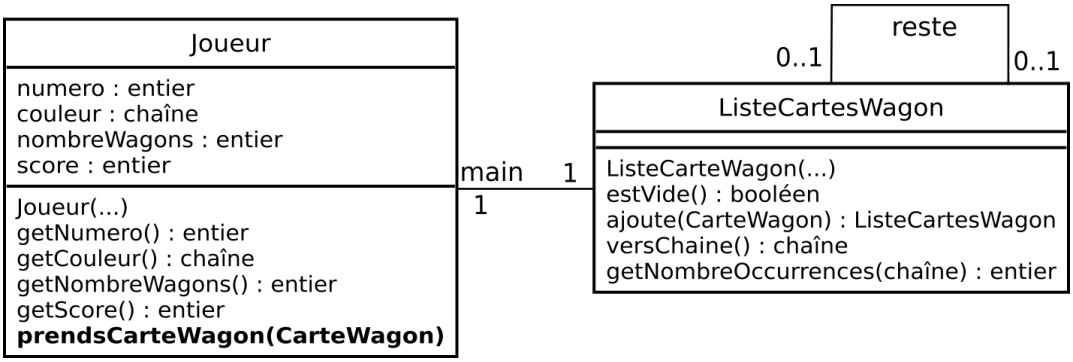

Figure 3: Diagramme de classe de la classe Joueur

3.3.1 Modifier la classe Joueur

Travail à faire

Modifier la définition de la classe **Joueur** telle qu'elle est représentée dans la figure 3.
Implémenter l'accesseur et le mutateur pour l'attribut `mainCartesWagon`.
Se référer à vos cours pour choisir la signature de vos accesseurs.
La méthode `prendsCarteWagon()` a pour objectif d'ajouter une carte à la main du joueur.
Écrire la méthode `prendsCarteWagon()`.
Penser à initialiser correctement la main du joueur lors de la création d'un joueur.

3.3.2 Vérifier l'implémentation de la classe Joueur

Travail à faire

Exécuter la classe **TestJoueurMain** pour valider l'implémentation de la classe **Joueur**.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

3.4 La classe Jeu

Dans cet incrément, la classe **Jeu** est modifiée pour (1) distribuer des cartes wagon aux joueurs lors de l'initialisation d'une partie, et (2) transférer les cartes wagons piochées par le joueur courant vers sa main.

Pour ce faire, une méthode supplémentaire est nécessaire (voir figure 4), ainsi que des modifications des méthodes existantes.

3.4.1 Distribuer des cartes wagon

Lors du démarrage d'une partie des ADR, chaque joueur reçoit quatre cartes wagons de la pioche de cartes wagon. La méthode `distribueCartesWagon()` pioche des cartes wagon dans la pioche du jeu et les transfère dans la main du joueur.

Travail à faire

Écrire la méthode `distribueCartesWagon()`.
Modifier la méthode principale de la classe **Jeu** pour distribuer les cartes wagons aux joueurs.

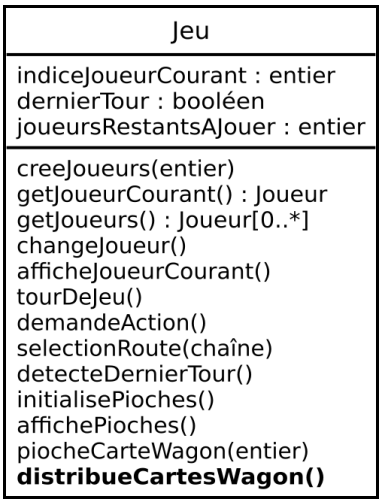

Figure 4: Diagramme de classe de la classe Jeu

Travail à faire

Valider la distribution de cartes wagon aux joueurs par l'exécution de la classe de test **TestJeuDistributionCW**.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

3.4.2 Afficher la main du joueur

Travail à faire

Modifier la méthode `afficheJoueurCourant()` pour afficher la main du joueur.
Compléter le cahier de test avec les tests de validations à exécuter et leurs résultats d'exécution.

3.4.3 Piocher des cartes wagon

Pour rappel, la pioche des cartes wagon est gérée par la méthode `piocheCarteWagon()` qui est appelée après la sélection d'une pioche par le joueur.

Dans cette version, la carte piochée par le joueur courant est transférée dans sa main.

Travail à faire

Modifier le diagramme d'activité lié au fonctionnement de la méthode `piocheCarteWagon()` pour intégrer le transfert de la carte piochée vers la main du joueur.
Modifier la méthode `piocheCarteWagon()` de la classe **Jeu** conformément au diagramme d'activité produit.

Travail à faire

Valider le comportement des pioches de cartes wagon.
Compléter le cahier de test avec les résultats de l'exécution des tests.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

4 Analyse du code ADR

Pour comprendre le fonctionnement de l'affichage du jeu et en particulier des pioches de cartes et de la main du joueur, il est demandé de fournir une analyse et une conception partielle de la classe **IHM**.

Travail à faire

En utilisant le formalisme UML vu en cours, analyser les méthodes `affichePioches()` et `afficheCartesJoueur()` de la classe **IHM** et proposer une modélisation correspondant à leur fonctionnement.

5 Fin de l'incrément 7

En arrivant à la fin de ce document, votre jeu permet de distribuer quatre cartes wagon aux joueurs lors de l'initialisation d'une partie, et le joueur courant peut se constituer une main, affichée sur l'interface graphique.

En fonction du temps qu'il vous reste avant l'incrément suivant, vous pouvez :

- Compléter et améliorer les documents de modélisation et de vérification/validation
- Travailler sur les piles et les listes (en auto-évaluation)
- Préparer l'examen sur machine n°3