Projet d'Algorithmique et de Développement Logiciel

Les Aventuriers du Rail

Incrément 5

Séances 16 et 17

Synthèse des incréments réalisés et à venir

Le Projet Informatique comporte huit incréments dans l'objectif de produire une version simplifiée du jeu. Chaque incrément consiste à réaliser un objectif particulier qui se reflète dans l'implémentation, soit par l'ajout de nouvelles classes ou attributs, soit par l'ajout de méthodes, soit les deux.

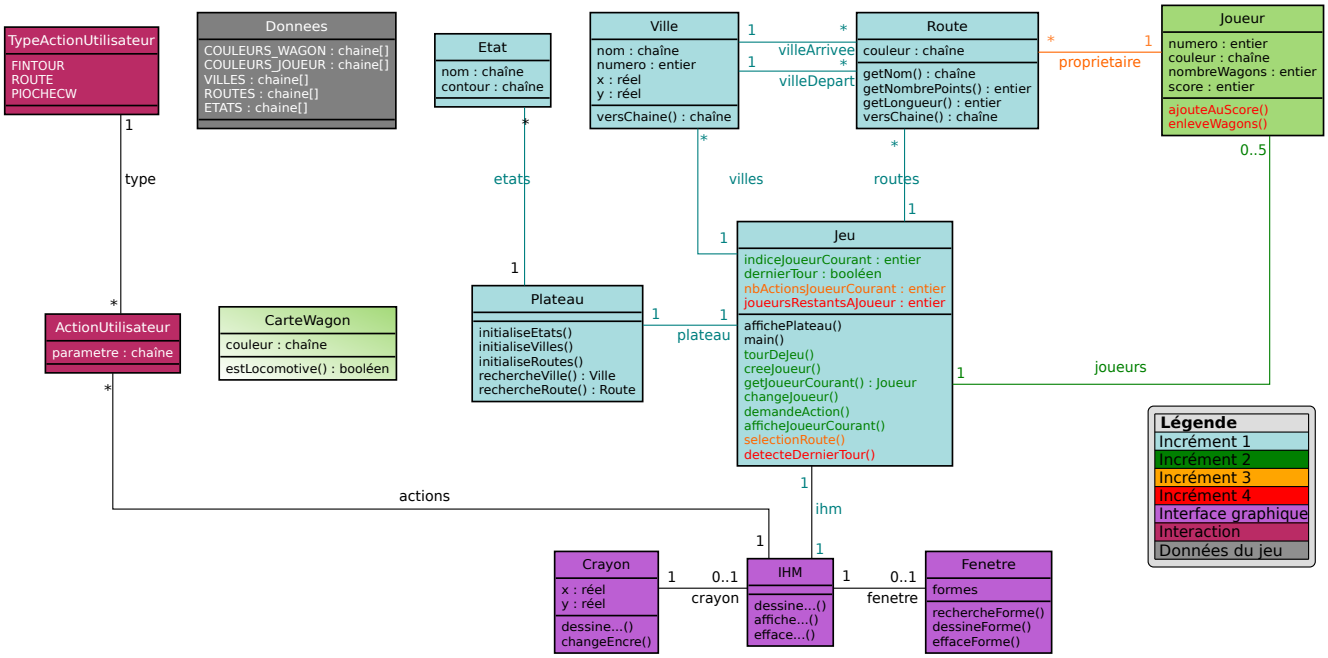

Figure 1: : Classes identifiées dans le jeu des ADR - Incréments 1 à 4

Incréments 1 à 4

Les quatre premiers incréments se concentrent sur les éléments de base du jeu (voir figure 1) :

- un **Plateau** qui représente la carte « géographique » et les informations du jeu
  - des **Etats** pour dessiner la carte des Etats-Unis
  - des **Villes**
  - des **Routes** pour relier les villes entre elles
- des **Joueur**s pour stocker les informations des joueurs dans le jeu (e.g. score, nombre de wagons)
- un **Jeu** qui réalise le déroulement d'une partie suivant les règles du jeu.

À ces éléments de base s'ajoutent des « utilitaires » (voir figure 1), qui permettent de stocker des données, de structurer le code ou de modéliser le comportement du jeu :

- l'interface graphique (**IHM**, **Crayon** et **Fenetre**)
- les actions des utilisateurs (**ActionUtilisateur** et **TypeActionUtilisateur**)
- les données du jeu original (e.g les noms des villes et leurs coordonnées) (**Donnees**)

Ces éléments permettent de produire, à la fin de l'incrément n°4, une première version du jeu qui est fonctionnelle mais incomplète :

- le jeu initialise et démarre une partie
  - le jeu crée des joueurs
  - le jeu affiche le plateau de jeu et les informations du joueur courant
- le joueur courant sélectionne des routes et termine son tour de jeu
- le jeu termine la partie

Le jeu implémente ces fonctionnalités dans les méthodes des classes (voir figure 1).

Incréments 5 à 8

Les incréments 5 à 8 permettent de compléter le jeu par l'introduction des cartes « wagon » et la détection du vainqueur de la partie :

- une **CarteWagon** qui représente une carte « wagon » colorée, ou une locomotive.
- des Pioches de cartes pour que les joueurs puissent piocher des cartes wagon et se constituer une « main »
- du calcul de scores pour intégrer les bonus de jeu en fin de partie et déterminer un vainqueur

Ces évolutions modifient également l'affichage du plateau de jeu et de la partie pour refléter les nouvelles fonctionnalités.

Sommaire

1 Préambule

L'incrément 5 du Projet Informatique s'appuie sur la version électronique produite pendant l'incrément 4 du jeu des Aventuriers du Rail (ADR).

L'objectif principal du cinquième incrément est de mettre en place les pioches de cartes wagon du jeu.

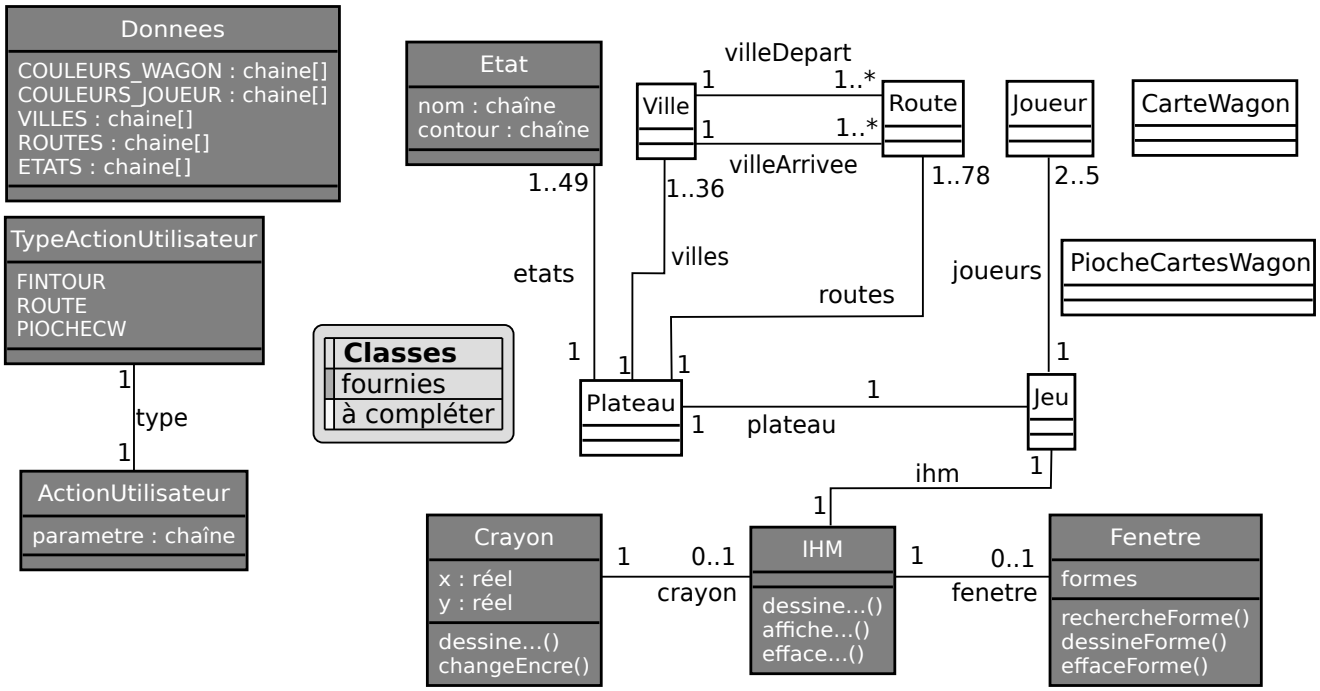

Figure 2: : Classes identifiées dans le jeu des ADR - Incrément 5

Pour implémenter cette fonctionnalité, il est nécessaire de définir une nouvelle classe **PiocheCartesWagon** et de s'appuyer sur la version de la classe **Jeu** déjà implémentée (voir le diagramme de classes UML simplifié et incomplet en figure 2).

2 Évaluation du travail

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

L'évaluation des compétences liées à ce cinquième incrément est réalisée par l'évaluation sur machine n°2.

3 Travail demandé

L'incrément 5 du projet consiste à implémenter la pioche de cartes wagon non visibles, ainsi que la défausse de cartes wagon associée. Cette pioche et cette défausse seront réalisées sous la forme d'une pile de cartes. Cet incrément consiste également à implémenter la pioche de cartes wagon visibles. Cette pioche est réalisée via un tableau de cartes (et non une pile).

3.1 Préparation de l'environnement

Importez l'incrément 5 dans votre projet existant (incréments 1 à 4) en suivant la procédure décrite dans la documentation du logiciel DevCube.

Vous devez avoir réalisé la création de la classe **CarteWagon**, demandée à la fin de l'incrément 2.

`NE PAS UTILISER LA FONCTION IMPORT` pour inclure votre classe **CarteWagon** : ouvrir le fichier DevCube contenant la classe **CarteWagon** et copier-coller le code de la classe **CarteWagon** dans la classe **CarteWagon** de l'incrément 5 du projet.

3.2 La classe PiocheCartesWagon

La classe **PiocheCartesWagon** est l'implémentation de la pioche « cachée » du jeu. Pour faciliter la manipulation des cartes de la pioche, on utilise le type abstrait Pile qui dispose d'attributs et de méthodes que l'on retrouve dans toute implémentation de pile.

Pour les besoins du jeu, nous définissons des méthodes supplémentaires qui sont décrites dans la figure 3.

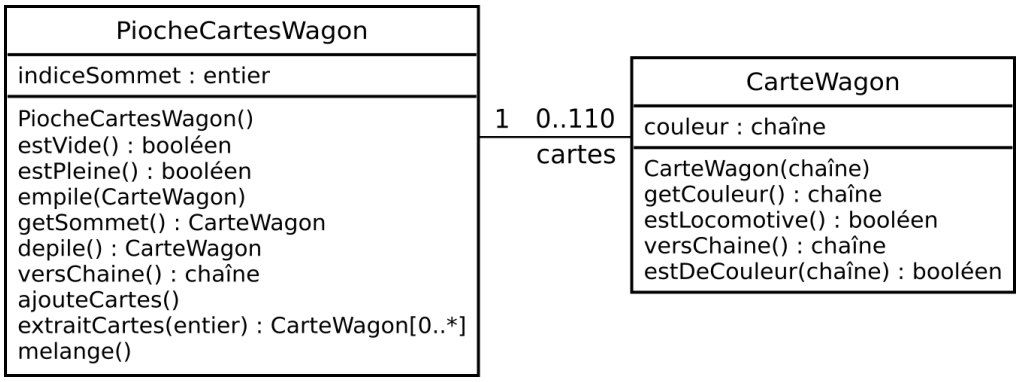

Figure 3: Diagramme de classe de la classe PiocheCartesWagon

3.2.1 Définir la classe PiocheCartesWagon

Travail à faire

Définir la classe **PiocheCartesWagon** telle qu'elle est représentée dans la figure 3.

3.2.2 Création d'une pile et initialisation

Pour utiliser la pioche de cartes « cachées », il est nécessaire de créer un nouvel objet de type **PiocheCartesWagon**.

Dans le cas de la classe **PiocheCartesWagon**, on procède aux initialisations du tableau de cartes et de l'indice du sommet de la pile.

Le tableau de cartes est initialisé de manière à accueillir les 110 cartes « wagon » du jeu. La pile est initialement vide.

Travail à faire

Écrire le constructeur `PiocheCartesWagon()` de la classe **PiocheCartesWagon**.

Travail à faire

Exécuter la classe **TestPiocheCWInitialisation**.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

3.3 Méthodes `estVide`, `estPleine`, `empile`, `getSommet` et `depile`

Travail à faire

Ecrire les méthodes `estVide()`, `estPleine()`, `getSommet()`, `empile()` et `depile()` de la classe **PiocheCartesWagon** (vues en cours sur le type abstrait *Pile*).
S'il est impossible d'empiler car la pile est pleine, « renvoyer » une erreur en utilisant la syntaxe suivante :

```
if(this.estPleine()){
  throw new Error("Impossible d'empiler une carte : pile pleine");
}
```

S'inspirer de la méthode `empile()` pour la gestion des cas d'erreur de la méthode `depile()`.

3.3.1 Vérifier l'implémentation de la classe PiocheCartesWagon

Travail à faire

Exécuter les classes **TestPiocheCWVidePleine**, **TestPiocheCWSommet**, **TestPiocheCWEmpile**, **TestPiocheCWDepile** pour valider l'implémentation de la classe **PiocheCartesWagon**.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.
REMARQUE Nous vous conseillons de tester chaque méthode au fur et à mesure.

3.3.2 Méthode `versChaine`

La méthode `versChaine()` a pour objectif de produire une représentation textuelle de la pile de cartes wagon. Cela permet notamment de vérifier que l'initialisation de la pioche se déroule correctement et de présenter une information lisible par un humain.

On désire produire une chaîne de caractères qui contient une ligne par carte, chaque ligne affiche le numéro de la carte et sa couleur.

Travail à faire

Ecrire la méthode `versChaine()` dans la classe **PiocheCartesWagon**.
Le résultat de l'appel à la méthode `versChaine()` pour une pioche initialisée et remplie (voir section 3.3.4) doit renvoyer :

```
1: Wagon orange
2: Wagon orange
3: Wagon orange
4: Locomotive
...
110: Wagon noir
```

3.3.3 Vérifier l'implémentation de la méthode `versChaine`

Travail à faire

Exécuter la classe **TestPiocheCWAffichage** pour valider l'implémentation de la méthode `versChaine()`.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.
REMARQUE Le succès de l'exécution de la méthode `versChaine()` dépend de l'implémentation de la méthode `versChaine()` de la classe **CarteWagon**.

3.3.4 Méthode `ajouteCartes`

La pioche de cartes wagon « cachée » contient, au début de la partie, l'ensemble des cartes wagon du jeu, c'est-à-dire 12 cartes wagon pour chacune des 8 couleurs du jeu, auxquelles on ajoute les 14 cartes « locomotive ».

La méthode `ajouteCartes()` crée les cartes wagon correspondantes et les empile dans la pile.

Travail à faire

Écrire la méthode `ajouteCartes()` dans la classe **PiocheCartesWagon**.
Pour la génération des cartes wagon, utiliser les couleurs fournies dans le projet (voir classe **Donnees**). Se reporter à la classe **CarteWagon** pour se remémorer son fonctionnement.

3.3.5 Vérifier l'implémentation de la méthode `ajouteCarte`

Travail à faire

Exécuter la classe **TestPiocheCWAjoutCartes** pour valider l'implémentation de la méthode `ajouteCarte()`.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

3.3.6 Méthode `extraitCartes`

La méthode `extraitCartes()` est utilisée pour distribuer des cartes aux joueurs. Son fonctionnement consiste à dépiler des cartes de la pile et à les stocker dans un tableau de cartes. Le tableau de cartes créé est renvoyé par la méthode.

Travail à faire

Écrire la méthode `extraitCartes()` dans la classe **PiocheCartesWagon**.

3.3.7 Vérifier l'implémentation de la méthode `extraitCartes`

Travail à faire

Exécuter la classe **TestPiocheCWExtraction** pour valider l'implémentation de la méthode `extraitCartes()`.
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 déclarer, créer et initialiser la défausse et les pioches de cartes « cachées » et « visibles ».

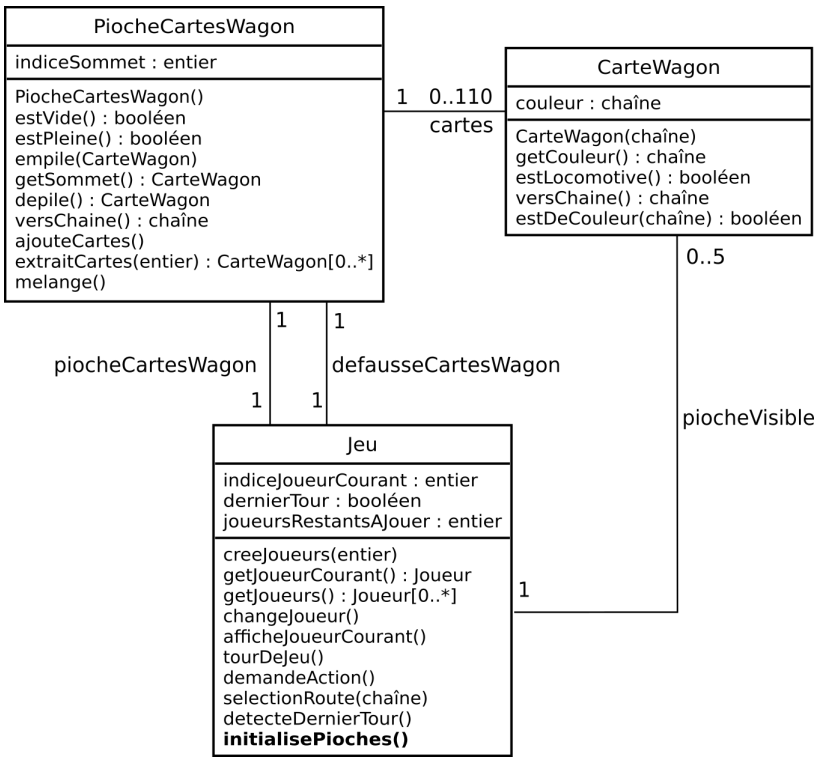

Figure 4: Diagramme de classe de la classe Jeu

3.4.1 Modifier la classe Jeu

Travail à faire

Modifier la définition de la classe **Jeu** telle qu'elle est représentée dans la figure 4.
Implémenter les accesseurs pour les nouveaux attributs.

3.4.2 Initialiser les pioches et la défausse

La méthode `initialisePioches()` initialise les pioches et la défausse de cartes wagon, conformément aux règles du jeu des ADR.

Travail à faire

Écrire la méthode `initialisePioches()`.
Utiliser les attributs de la classe **Jeu** et les méthodes de la classe **PiocheCartesWagon** pour initialiser la pioche cachée et la défausse de cartes wagon.
La pioche visible est représentée sous la forme d'un tableau de 5 cartes wagon. Initialiser la pioche « visible » du jeu.

3.4.3 Vérifier l'implémentation de la méthode `initialisePioches`

Travail à faire

Exécuter la classe **TestJeuInitialisationPioches** pour vérifier l'implémentation de la méthode `initialisePioches()`.
Si des erreurs surviennent lors de la compilation ou de l'exécution, apporter les corrections nécessaires.

3.4.4 Initialisation du jeu (version 4)

Travail à faire

Utiliser la méthode `initialiserPioches()` lors de l'initialisation du jeu des ADR pour initialiser les pioches et la défausse de cartes wagon.

4 Analyse du code ADR

Pour comprendre le fonctionnement de l'affichage du jeu et en particulier des routes sur le plateau de jeu, il est demandé de fournir une analyse et une conception préliminaire de la classe **IHM**. Le travail sera principalement orienté sur le fonctionnement qui peut être décrit avec les diagrammes UML vus en cours.

Travail à faire

Proposer un diagramme d'activités pour modéliser le fonctionnement de la méthode `dessineRoutes()` de la classe **IHM**.

5 Fin de l'incrément 5

En arrivant à la fin de ce document, votre jeu initialise les pioches de cartes wagon du jeu, en préparation à la gestion des cartes qui est réalisée dans les incréments suivants.

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
  - Mettre à jour le ou les diagrammes de classes avec les modifications apportées aux classes dans cet incrément.
  - Compléter le cahier de test à partir des données de test identifiées.
- Travailler sur les piles (en auto-évaluation) en préparation de l'examen sur machine n°2

- Réaliser les extensions de l'incrément 5 (optionnel - section 6)

6 Extensions

6.1 Mélange des pioches

Dans la version actuelle de l'implémentation des pioches, les cartes sont insérées « par famille/couleur ».

Travail à faire

Dans la méthode `melange()` de la classe **PiocheCartesWagon**, proposer une solution pour mélanger les cartes wagon de la pioche.