LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2021

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI, CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :* 21/09/2021

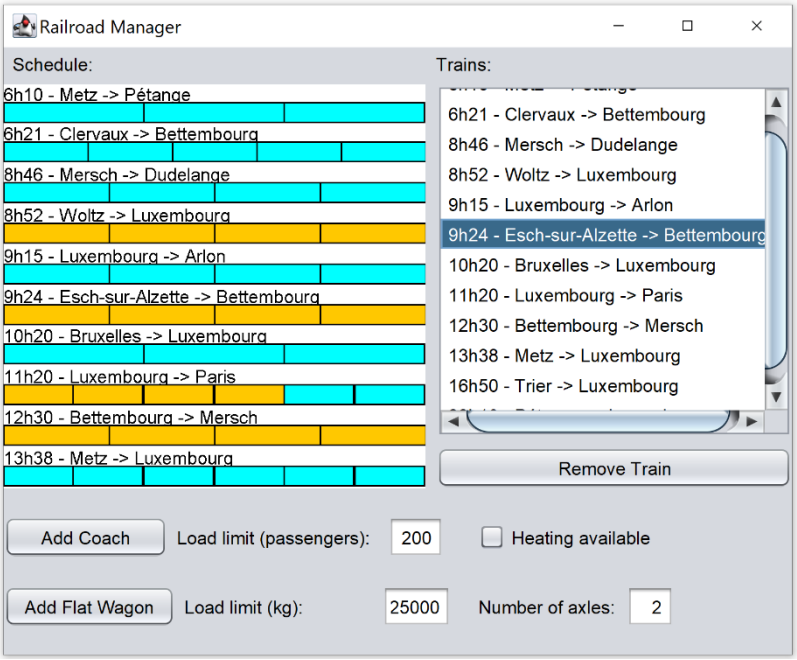
Dans votre répertoire de travail (à définir par chaque lycée), vous trouverez un dossier nommé **EXAMEN_CI**. Renommez ce dossier en remplaçant le nom actuel par votre code de l'examen (exemple de notation : **LXY_CI2_05**). Tous vos fichiers devront être sauvegardés à l'intérieur de ce dossier, qui sera appelé **votre dossier** par la suite.

Direction des chemins de fer

Partie modélisation — 10 points

- La partie modélisation doit être rendue avant que la partie implémentation puisse être débutée. Cette partie est à réaliser sur papier.
- La partie modélisation est à rendre après 35 minutes au plus tard, mais vous être libre de la rendre plus tôt.

La direction des chemins de fer, pour laquelle vous travaillez, vous demande d'implémenter un système pour visualiser les dix prochains trains qui vont arriver à leur destination. Chaque train dispose donc d'un horaire (heure d'arrivée) ainsi que d'une ville d'origine et d'une destination. En plus, à chaque train peuvent être rattachés des wagons.



Pour mieux pouvoir prendre des décisions, la direction vous demande d'implémenter un moyen de différencier entre les wagons qui transportent des passagers et ceux qui transportent de la cargaison (*EN : cargo*). Les wagons portant des passagers renseignent sur la présence d'un chauffage. Les

wagons transportant de la cargaison renseignent sur le nombre d'essieux (EN : axles). Chaque type de wagon dispose aussi d'une capacité maximale qui ne peut être excédée et indique sa charge actuelle.

Tous les trains circulant sur le réseau sont enregistrés dans un horaire (*EN : schedule*). L'horaire enregistre aussi le temps actuel.

La direction vous demande d'implémenter les fonctionnalités suivantes :

1. En lançant l'application, l'horaire charge les données des trains d'un fichier CSV fourni à l'application. L'horaire affiche toujours, au plus, les 10 prochains trains à arriver à leur destination. La liste doit toujours être triée selon l'horaire spécifique des trains.
2. La vue affiche les 10 prochains trains à arriver à leur destination. Si moins de 10 trains circulent sur le réseau, la vue affiche tous ces trains. Pour chaque train, l'application affiche son heure d'arrivée, son origine et sa destination (p. ex. : 6h10 – Metz -> Dudelange). Ensuite, un graphique montre le nombre de wagons dans le train ainsi que leur type ; cyan pour un wagon transportant des passagers, orange pour un wagon transportant de la cargaison.
3. Quand un train tombe en panne, il peut être nécessaire de le retirer de l'horaire. À cette fin, l'utilisateur peut sélectionner un train dans la liste et actionner le bouton « Remove Train ».
4. Quand un train se voit ajouter un wagon, les boutons « Add Coach » et « Add Flat Wagon » servent à ajouter soit un wagon pour des passagers soit un wagon pour transporter de la cargaison au train sélectionné dans la liste. A côté de ces boutons, des options supplémentaires sont disponible pour indiquer la charge maximale du wagon ainsi que certaines autres caractéristiques liées au wagon concret en question.

Travail à faire

Réaliser un diagramme de classe UML du **modèle** de l'application sur papier en respectant les conventions UML usuelles.

1. Limitez-vous uniquement aux attributs des classes.
2. Vous pouvez compléter votre modèle de classe par des explications supplémentaires.

LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2021

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI, CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :*

Partie implémentation — 50 points

Ouvrez dans votre dossier le projet **`RailroadManager`** mis à disposition. Complétez l'application en vous basant sur la version exécutable, le diagramme UML, le fichier CSV comportant les données à charger et les instructions données par la suite.

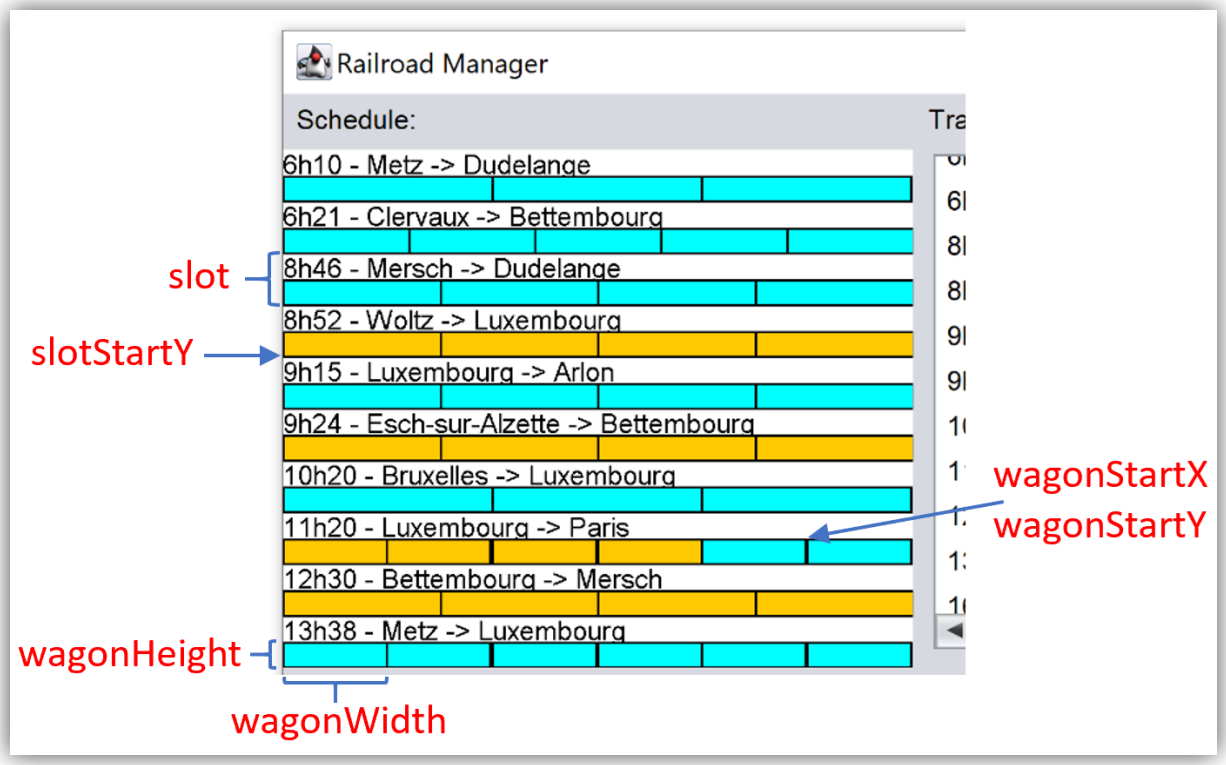


Figure 1: Figure montrant la relation entre paramètres et attributs et le graphique final.

La classe TrainWagon — *4 points*

Cette classe abstraite représente un wagon de train. Elle regroupe les caractéristiques communes de tous les types de wagons de train.

1. Créez la classe, ajoutez les attributs et générez le constructeur ainsi que les accesseurs et manipulateurs selon le diagramme de classe UML. **1,5p**
2. Implémentez la méthode `paint(…)`. La méthode est utilisée pour peindre le wagon comme rectangle dans la couleur appropriée avec un bord noir selon les paramètres fournis. Référez-vous à la Figure 1 pour une représentation du graphique final et l'utilisation de paramètres et attributs dans le graphique. **2,5p**

Les classes Coach et FlatWagon *4 points*

Ces classes représentent des wagons de train concrets. Elles représentent les wagons à passagers (EN : coach) respectivement les wagons pouvant charger de la cargaison (EN : flat wagon). La classe `Coach` enregistre le nombre de passagers actuels, si le wagon dispose d'un chauffage et est affichée sur l'horaire en cyan. La classe `FlatWagon` enregistre le poids de la cargaison chargée, le nombre d'essieux (EN : axle) et est affichée sur l'horaire en orange.

1. Créez les classes, ajoutez les attributs et générez les constructeurs de chaque classe selon le diagramme de classe UML. **2 * 2p**

La classe Train *20 points*

Cette classe représente un train avec une origine, une destination et un temps d'arrivée enregistrée comme heures et minutes dans deux attributs séparés. La classe maintient aussi une référence vers le premier wagon attaché et une indication de la taille tu train, comptant seulement les wagons. Par exemple, un train sans wagons à une taille de 0 et un train avec trois wagons a une taille de 3. La référence vers le premier wagon sert à maintenir la liste des wagons selon une implémentation d'une liste chaînée simple.
Une partie de cette classe vous est fournie. Utilisez donc cette classe et ne créez pas une nouvelle classe `Train`.

1. Complétez la liste des attributs comme donnée par le diagramme de classe UML.
2. Changez la classe pour qu'elle implémente l'interface `Comparable`. Un train est comparable à un autre train selon les règles suivantes :
    a. Un train *A* arrivant à sa destination est triée dans la liste avant un train *B* arrivant à sa destination plus tard.
    b. Si deux trains arrivent à leur destination au même temps, leur ordre est indéfini. Implémentez la solution qui vous semble le plus opportun.

    Pour cette implémentation, le temps d'arrivée se réfère aux attributs `arrivalHour` et `arrivalMinute`. **4,5p**
3. Implémentez la méthode `addToEnd` qui permet d'ajouter un wagon au dernier wagon actuellement rattaché au train. Attention de veiller sur l'intégrité de l'attribut `size`. **7p**
4. Implémentez la méthode `paint(…)`. Référez-vous à la Figure 1 pour une représentation du graphique final et l'utilisation de paramètres et attributs dans le graphique. Cette méthode va dessiner dans le créneau (EN : slot) fourni par l'horaire :
    a. Une chaîne de caractères informative renseignant sur la nature du train sous la forme : *[heure d'arrivée]*h*[minute d'arrivée] – [origine] -> [destination]*. Consultez le programme de démonstration ou la capture d'écran fournie avec cet énoncé pour des exemples. Pour optimiser le positionnement du texte, considérez que la taille de la police par défaut est de 12 pixels. **1,5p**
    b. Tous les wagons du train. Comme chaque train, indépendamment de sa taille, est dessiné sur toute la largeur fournie à cette méthode, cette méthode doit partitionner l'espace disponible et fournir à chaque wagon les bonnes coordonnées pour pouvoir se dessiner de manière autonome. La hauteur des wagons est donnée par `wagonHeight` qui correspond à la moitié de la hauteur d'un créneau comme déterminée par l'horaire. **7p**

La classe Schedule *22 points*

Cette classe représente l'horaire et comprend des attributs pour enregistrer le temps actuel et maintient une liste des trains sur le réseau. Cette classe permet de charger les données de l'horaire (`load(…)`), de trier l'horaire (`sort()`) et de déléguer des appels à la liste des trains, notamment l'appel pour dessiner l'horaire (`paint(…)`).

1. Créez la classe et ajoutez les attributs selon le diagramme de classe UML. Initialisez l'heure à 12h00. **1p**
2. Générez les méthodes délégant les appels vers l'instance de `ArrayList` maintenu par `Schedule`. **1p**
3. Implémentez la méthode `sort()` qui permet de trier la liste des trains. Utilisez le fait que la classe `Train` implémente l'interface `Comparable` pour simplifier cette méthode. **1,5p**
4. Implémentez la méthode `load`(…) qui permet de charger les données de tous les trains circulant sur le réseau. Pour pouvoir correctement charger les données, consultez le fichier « `schedule.csv` » qui vous est fourni. Ce fichier contient les données sur les trains, un train par ligne, sous la forme :

   [Origine],[Destination],[Heure d'arrivée],[Minute d'arrivée], Informations sur les wagons

   Les informations sur les wagons sont présentées en triples sous la forme :

   [Type de wagon],[Charge maximale],[Charge actuelle],[climatisation|\|nombre d'essieux]

   Le nombre de wagons par train peut varier et peut être 0. Votre implémentation doit pouvoir charger un train sans wagons.
   Traitez des exceptions dans la méthode et écrivez le message d'erreur dans la console. **11p**
5. Implémentez la méthode `paint`(…). Cette méthode permet de dessiner l'horaire. Référez-vous à la Figure 1 pour une représentation du graphique final et l'utilisation de paramètres et attributs dans le graphique. L'horaire ne fait dessiner que les dix premiers trains. L'horaire est toujours divisé en dix créneaux de hauteurs égales. Chaque train est dessiné dans un créneau. Si moins de dix trains circulent sur le réseau, les créneaux superflus restent vides. **7,5p**

Les classes MainFrame et DrawPanel *0 points*

Ces deux classes vous sont fournies. La classe `DrawPanel` implémente la vue de l'application. La classe `MainFrame` implémente le contrôleur de l'application. Ces deux classes sont complètes mais comportent du code commenté dans certaines méthodes pour éviter l'affichage d'erreurs dès le début. Après que vous avez implémenté les autres classes et leurs méthodes, enlevez les commentaires pour pouvoir lancer votre application.

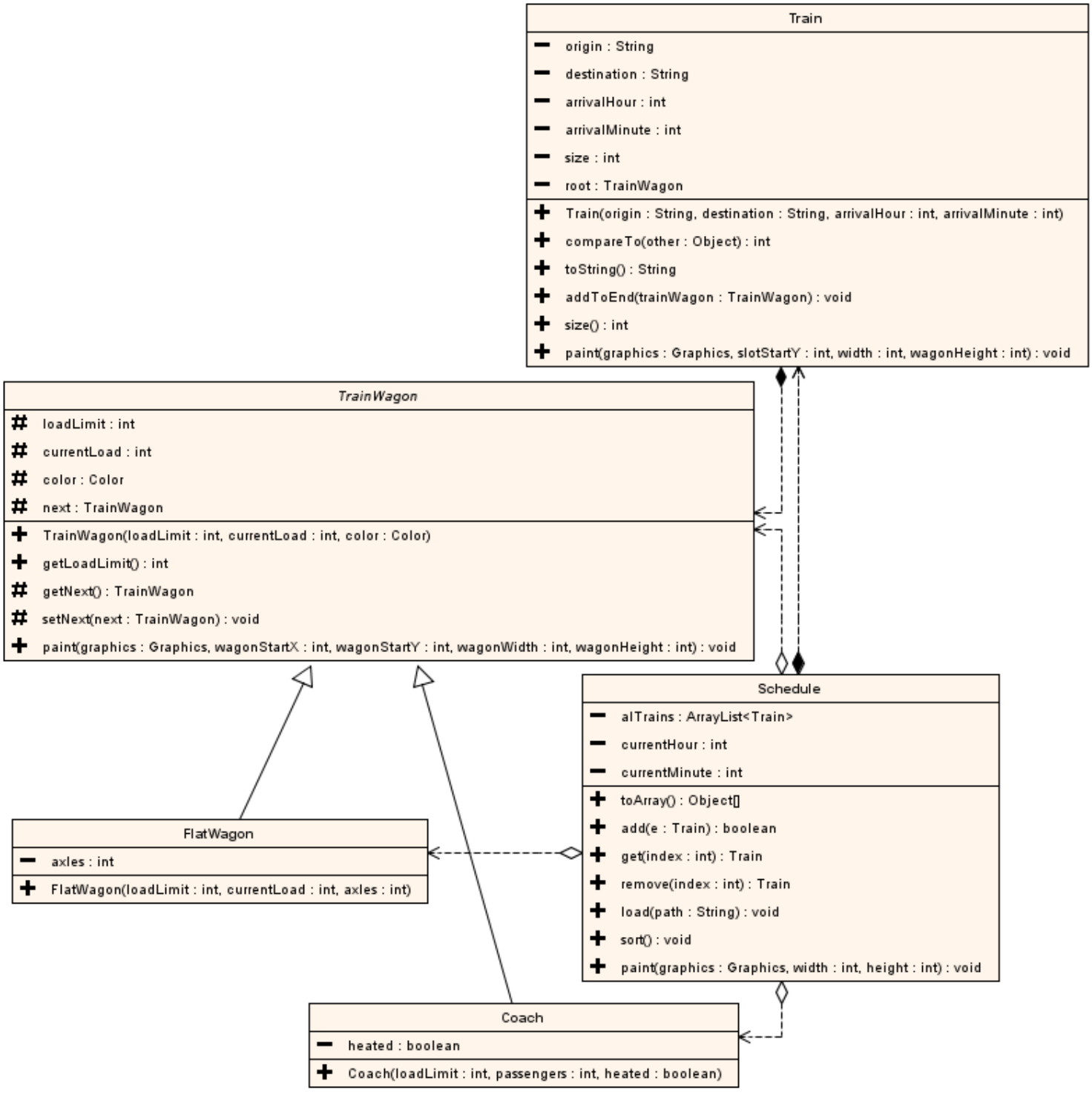


Diagramme 1: Diagramme de classe UML du modèle.