Projet Trains et Circuits – Nicolas Sempéré

Table des matières

[Exercice 1 - Le comportement d’un train 2](#_Toc125303871)

[⊳ Question 1.1 : **Dans le diagramme de classes précédent, quel sera le rôle de chaque classe dans la réalisation du déplacement d’un train ?** 2](#_Toc125303872)

[⊳ Question 1.2 : **Modifiez le diagramme de classes initial en ajoutant les méthodes et/ou attributs nécessaires à la réalisation du déplacement d’un train.** 2](#_Toc125303873)

[⊳ Question 1.3 : **Donnez le code des méthodes identifiées. Pour valider le bon fonctionnement de vos méthodes, vous pouvez afficher l’état du train chaque fois qu’il change de position.** 2](#_Toc125303874)

[Exercice 2 Plusieurs trains sur la ligne 3](#_Toc125303875)

[⊳ Question 2.1 : **Modifiez votre programme pour qu’il puisse y avoir plusieurs trains actifs (en déplacement) sur la ligne.** 3](#_Toc125303876)

[⊳ Question 2.2 : **Identifiez les variables qui permettent d’exprimer l’invariant de sûreté pour la ligne de trains.** 3](#_Toc125303877)

[⊳ Question 2.3 : **À l’aide des variables identifiées, exprimez l’invariant de sûreté.** 3](#_Toc125303878)

[⊳ Question 2.4 : **Quelles sont les actions « critiques » que peut effectuer un train ?** 3](#_Toc125303879)

[⊳ Question 2.5 : **Dans quelles classes ces actions doivent être ajoutées ?** 3](#_Toc125303880)

[⊳ Question 2.6 : **Selon la méthode de construction d’une solution de synchronisation donnée plus haut. Quelles autres méthodes faut-il ajouter et dans quelle classe ?** 3](#_Toc125303881)

[⊳ Question 2.7 : **Ajoutez les méthodes identifiées dans les classes correspondantes.** 3](#_Toc125303882)

[⊳ Question 2.8 : **Modifiez maintenant le comportement d’un train pour qu’il utilise les méthodes ajoutées. Testez le bon fonctionnement de votre solution en démarrant l’exécution d’un, puis deux, puis trois trains.** 3](#_Toc125303883)

[Exercice 3 - Éviter les interblocages 3](#_Toc125303884)

[⊳ Question 3.1 : **Identifiez les variables qui permettent d’exprimer la nouvelle condition.** 3](#_Toc125303885)

[⊳ Question 3.2 : **À l’aide des nouvelles variables, identifiez la nouvelle condition pour l’invariant de sûreté.** 3](#_Toc125303886)

[⊳ Question 3.3 : **Quelle est la classe responsable de la gestion de ces variables ?** 3](#_Toc125303887)

[⊳ Question 3.4 : **Utilisez la méthode de construction d’une solution de synchronisation présentée dans l’exercice précédent pour tenir compte de cette nouvelle condition.** 3](#_Toc125303888)

[⊳ Question 3.5 : **Modifiez les méthodes leave et enter de la classe Section pour tenir compte de la nouvelle condition. Testez votre solution.** 3](#_Toc125303889)

[Exercice 4 - Gare intermédiaire 4](#_Toc125303890)

[⊳ Question 4.1 : **Modifiez votre code pour permettre d’ajouter des gares intermédiaires** 4](#_Toc125303891)

[⊳ Question 4.2 : **Constatez que vous devez ajouter un nouvel invariant de sûreté pour éviter un interblocage si la gare intermédiaire a n places et qu’il y a n + 2 trains. Déterminer ce nouvel invariant.** 4](#_Toc125303892)

[⊳ Question 4.3 : **Modifiez votre code pour l’assurer.** 4](#_Toc125303893)

# Exercice 1 - Le comportement d’un train

## ⊳ Question 1.1 : **Dans le diagramme de classes précédent, quel sera le rôle de chaque classe dans la réalisation du déplacement d’un train ?**

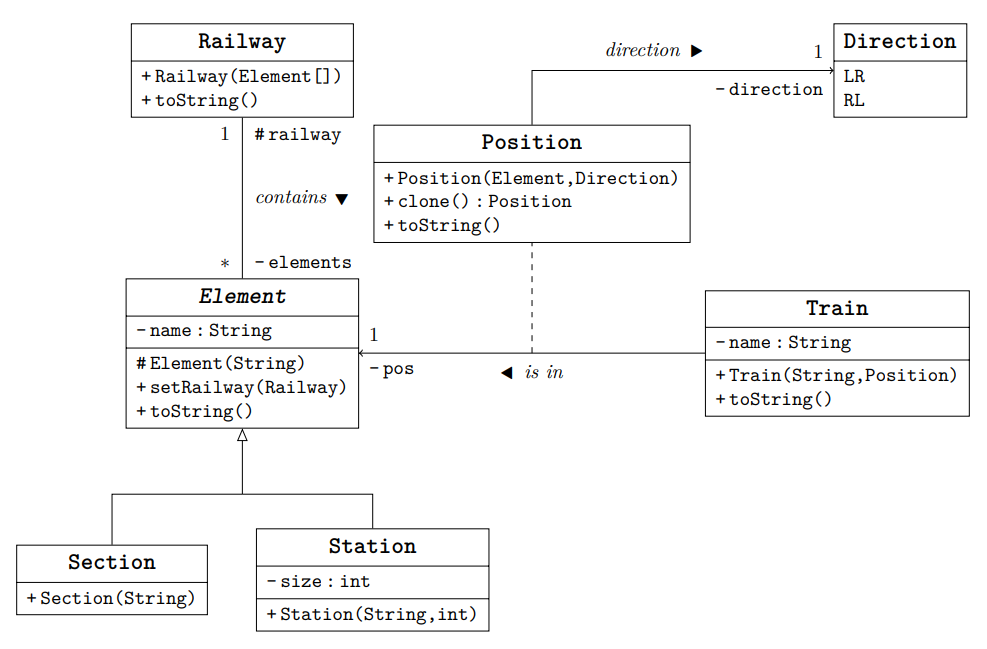


Diagramme de classe initial (fourni dans le sujet)

## ⊳ Question 1.2 : **Modifiez le diagramme de classes initial en ajoutant les méthodes et/ou attributs nécessaires à la réalisation du déplacement d’un train.**

Les méthodes et attributs ajoutées sont écrits en bleu.

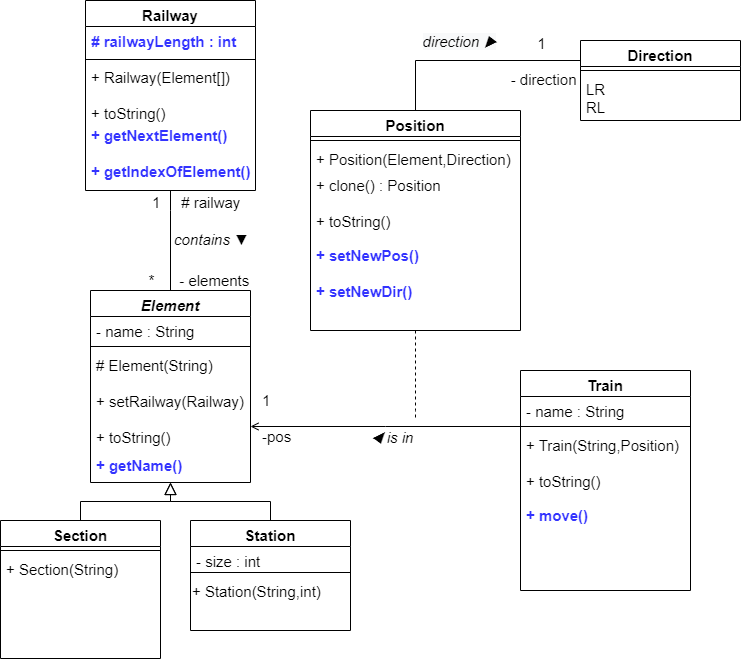


Diagramme de classe – Déplacement d’un unique train

Pour réaliser le déplacement d'un train :

- dans la classe "train", la méthode "move()" est appelé :

- la méthode "setNewPos()" de la classe Position

change la position du train :

- la méthode "getNextElement()" de la classe Railway

permet d'obtenir cette nouvelle position.

- la méthode "setNewDir()" de la classe Position

change la direction du train s'il est en bout de ligne :

- la méthode "getIndexOfElement()" de la classe Railway

permet de vérifier si le train est en bout de ligne.

## ⊳ Question 1.3 : **Donnez le code des méthodes identifiées. Pour valider le bon fonctionnement de vos méthodes, vous pouvez afficher l’état du train chaque fois qu’il change de position.**

Les méthodes ajoutées sont toutes dans le dossier src/train :

- move() | classe Train

- setNewPos() | classe Position

- setNewDir() | classe Position

- getNextElement() | classe Railway

- getIndexOfElement() | classe Railway

- getName() | classe Element

On ajoute la méthode "getName()" à la classe Element afin d'afficher la position du train à tout instant.

Le comportement du train est celui attendu (on appelle la méthode move() 20 fois dans le Main).

# Exercice 2 - Plusieurs trains sur la ligne

## ⊳ Question 2.1 : **Modifiez votre programme pour qu’il puisse y avoir plusieurs trains actifs (en déplacement) sur la ligne.**

## ⊳ Question 2.2 : **Identifiez les variables qui permettent d’exprimer l’invariant de sûreté pour la ligne de trains.**

## ⊳ Question 2.3 : **À l’aide des variables identifiées, exprimez l’invariant de sûreté.**

## ⊳ Question 2.4 : **Quelles sont les actions « critiques » que peut effectuer un train ?**

## ⊳ Question 2.5 : **Dans quelles classes ces actions doivent être ajoutées ?**

## ⊳ Question 2.6 : **Selon la méthode de construction d’une solution de synchronisation donnée plus haut. Quelles autres méthodes faut-il ajouter et dans quelle classe ?**

## ⊳ Question 2.7 : **Ajoutez les méthodes identifiées dans les classes correspondantes.**

## ⊳ Question 2.8 : **Modifiez maintenant le comportement d’un train pour qu’il utilise les méthodes ajoutées. Testez le bon fonctionnement de votre solution en démarrant l’exécution d’un, puis deux, puis trois trains.**

# Exercice 3 - Éviter les interblocages

## ⊳ Question 3.1 : **Identifiez les variables qui permettent d’exprimer la nouvelle condition.**

## ⊳ Question 3.2 : **À l’aide des nouvelles variables, identifiez la nouvelle condition pour l’invariant de sûreté.**

## ⊳ Question 3.3 : **Quelle est la classe responsable de la gestion de ces variables ?**

## ⊳ Question 3.4 : **Utilisez la méthode de construction d’une solution de synchronisation présentée dans l’exercice précédent pour tenir compte de cette nouvelle condition.**

## ⊳ Question 3.5 : **Modifiez les méthodes leave et enter de la classe Section pour tenir compte de la nouvelle condition. Testez votre solution.**

# Exercice 4 - Gare intermédiaire

## ⊳ Question 4.1 : **Modifiez votre code pour permettre d’ajouter des gares intermédiaires**

## ⊳ Question 4.2 : **Constatez que vous devez ajouter un nouvel invariant de sûreté pour éviter un interblocage si la gare intermédiaire a n places et qu’il y a n + 2 trains. Déterminer ce nouvel invariant.**

On arrive dans un état d’interblocage lorsque :

* Un train est en déplacement gauche-droite vers la gare du milieu.
* Un train est en déplacement droite-gauche vers la gare du milieu.
* La gare du milieu est pleine (elle contient n train à quai).

## ⊳ Question 4.3 : **Modifiez votre code pour l’assurer.**