Thème 6 - TME

Exercice

Lors de chaque TME, vous devrez créer un répertoire pour chaque exercice, y mettre les fichiers s'y rapportant et soumettre ce répertoire à la fin du TME.

Exercice 9 – Gestion d'un garage à locomotives

Nous considérons une zone de stationnement de locomotives composée de N hangars pouvant chacun accueillir une locomotive.

Une locomotive accède à la zone de stationnement via un segment de voie appelé segment d'accueil. Elle est ensuite orientée vers son hangar de destination au moyen d'un segment de voie tournant.

L'architecture du système est représentée dans la Figure 4.

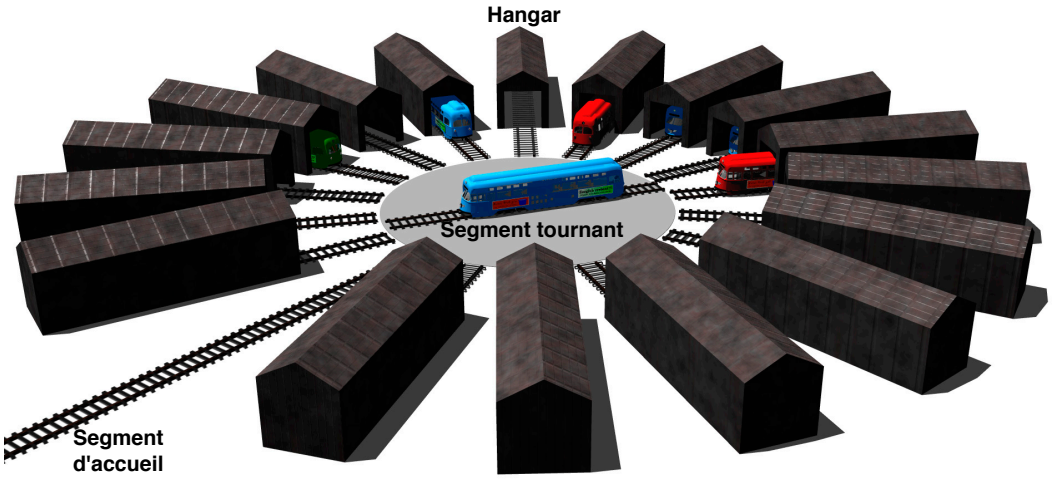

FIGURE 4 – Architecture du système pour $N = 17$

Nous souhaitons modéliser le mouvement des locomotives dans la zone de stationnement au moyen d'une application Java concurrente. Pour cela, nous considérons un système composé des éléments suivants :

- N locomotives,
- N hangars regroupés au sein d'un pool de hangars,
- le segment tournant.
- le segment d'accueil.

Pour accéder à un hangar, une locomotive commence par réserver le segment d'accueil. Elle appelle ensuite le segment tournant qui se positionne devant le segment d'accueil : la locomotive peut alors entrer dans le segment tournant. Celui-ci interroge l'état du pool de hangars pour récupérer un numéro de hangar disponible (il y en a toujours un).

La locomotive libère ensuite le segment d'accueil, attend que le segment tournant se positionne devant le hangar qui lui a été attribué, entre dans le hangar et libère le segment.

Une locomotive de notre système peut être modélisée par une classe `Loco` dont la méthode `run` est donnée ci-dessous :

```
public void run() {
   try {
      sAccueil.reserver();
      sTournant.appeler(0);
      sTournant.attendrePositionOK();
      sTournant.entrer(id);
      sAccueil.liberer();
      sTournant.attendrePositionOK();
      pHangars.getHangar( sTournant.getPosition() ).entrer(id);
      sTournant.sortir(id);
   }
   catch (InterruptedException e) {
      System.out.println("Loco " + id + " interrompue (ne devrait pas arriver)");
   }
```

Le paramètre de la méthode `appeler` correspond à la position (numéro de hangar, `0` pour le segment d'accueil) à laquelle doit se rendre le segment tournant. Un identifiant de locomotive est transmis à certaines méthodes pour permettre des affichages explicites.

Le but de cet exercice est de programmer les méthodes utilisées ici (et d'autres si nécessaire) de manière à garantir une progression synchronisée de l'ensemble des composants du système.

Question 1

Donnez l'implémentation de la classe `SegAccueil` dont une instance représente le segment d'accueil.

Question 2

L'ensemble des hangars est regroupé au sein d'un pool de hangars. Ce pool doit offrir (entre autres) une méthode permettant de rechercher un hangar libre à l'intérieur du pool et renvoyant sa position dans le pool.

Donnez l'implémentation de la classe `Hangar` et de la classe `PoolHangars`.

Nous nous intéressons maitenant à l'évolution de la locomotive et du segment tournant. Il faut garantir que ces deux éléments progressent de manière coordonnée pour que l'état du système reste cohérent. La méthode `run` de la classe `SegTournant` est donnée ci-dessous :

```
public void run() {
   try {
      while (true) {
         attendreAppel();
         seDeplacer();
         attendreEntree();
         seDeplacer();
         attendreVide();
      }
   }
   catch (InterruptedException e) {
      System.out.println("Terminaison sur interruption du segment tournant");
   }
}
```

Nous devons par exemple garantir que lorsque le segment tournant exécute son premier déplacement, il a bien été appelé par une locomotive.

Question 3

Écrivez les deux méthodes de la classe `SegTournant` correspondant à la procédure d'appel : `appeler` et `attendreAppel`. Vous préciserez les variables d'instance nécessaires, notamment celles utilisées pour la synchronisation.

Question 4
Écrivez les deux méthodes de la classe `SegTournant` correspondant au mouvement du segment : `seDeplacer` et `attendrePositionOK`. Vous préciserez les variables d'instance nécessaires, notamment celles utilisées pour la synchronisation. Vous simulerez le déplacement du segment par une attente.

Question 5
Écrivez les deux méthodes de la classe `SegTournant` correspondant à l'entrée d'une locomotive sur le segment : `entrer` et `attendreEntree`. Vous préciserez les variables d'instance nécessaires, notamment celles utilisées pour la synchronisation.

Question 6
Écrivez les deux méthodes de la classe `SegTournant` correspondant à la libération du segment par la locomotive : `sortir` et `attendreVide`. Vous préciserez les variables d'instance nécessaires, notamment celles utilisées pour la synchronisation.

Question 7
Complétez la classe `SegTournant` (le constructeur, les méthodes `getPosition` et `run`). Complétez la classe `Loco` et écrivez un programme de test permettant de vérifier le bon fonctionnement du système.