Thème 4 - TME

Exercices

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 5 – Implémentation du problème du barbier

Question 1

Proposez une implémentation multi-threads du problème du barbier.

Exercice 6 – Le pliage de la capote du Spider

La firme automobile UPM*Car* met au point son nouveau spider 3I001 (voir figure 3) pour lequel il faut concevoir un logiciel permettant de gérer une ouverture et une fermeture rapides de la capote. Nous nous intéressons ici à la phase d'ouverture (lorsqu'on replie la capote dans le coffre), et plus précisément à la synchronisation entre le mouvement de la capote et celui des vitres.

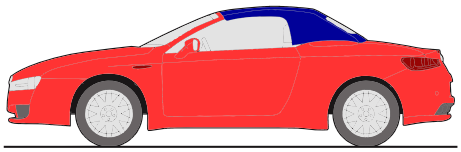

FIGURE 3 – Le nouveau spider 3I001 de chez UPM*Car*

La capote ne peut être rangée que lorsque les vitres sont ouvertes (en position basse). Pour obtenir de bonnes performances, les ingénieurs souhaitent paralléliser le mouvement des vitres. Il s'agit donc de réaliser une solution à trois threads : un thread principal qui pilote l'opération, et un thread pour le moteur de chaque vitre.

Afin d'obtenir un programme lisible, nous allons utiliser des types énumérés pour

- la position d'une vitre : `HAUTE`, `BASSE` ;
- l'opération demandée sur une vitre : `MONTER`, `DESCENDRE`, `NIL` s'il n'y a pas d'opération à effectuer ;
- le côté de la voiture concerné : `GAUCHE`, `DROITE` ;

(voir `http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html` pour la manipulation des types énumérés).

Question 1

Donnez le code Java pour les trois types énumérés.

Nous faisons l'hypothèse qu'il existe une classe `MoteurVitre` (avec un constructeur `MoteurVitre(Cote c)`) dont une instance permet de piloter une vitre. Cette classe dispose d'un accesseur `getPosition` permettant de savoir si la vitre est en position haute ou basse, et d'une méthode `demander` qui prend en paramètre l'opération demandée au moteur (lever ou baisser la vitre, même si on ne considère pour l'instant que la descente).

Le rôle du gestionnaire (pour l'instant) est de créer et démarrer les threads correspondant aux deux moteurs et de leur demander, si la vitre qu'ils pilotent est en position haute, de la faire descendre.

Question 2
Proposez une implémentation du gestionnaire de la capote. On ne se formalisera pas du fait que cette implémentation utilise des méthodes non encore écrites de la classe `MoteurVitre`.

Nous nous intéressons enfin à la programmation du moteur d'une vitre. Le comportement du moteur doit respecter les contraintes suivantes :

De manière répétitive (nous considérons pour l'instant que le moteur une fois lancé reste indéfiniment actif, il doit pouvoir traiter une infinité de demandes) :

- il attend de recevoir une demande de mouvement en provenance du gestionnaire, et ne fait rien tant que cette demande n'est pas arrivée ;
- il exécute l'opération demandée (qu'il s'agisse d'une montée ou d'une descente). Le temps nécessaire à cette opération sera simulé *dans l'exécution du moteur* par un délai aléatoire.

Question 3
Programmez une méthode permettant de réaliser l'attente par le moteur de la demande du gestionnaire.

Question 4
A quel moment l'attente se termine-t-elle ? Proposez une implémentation de la méthode `demander` (on stockera l'opération demandée dans une variable d'instance du moteur). Pourquoi l'exécution de l'opération (montée ou descente) ne doit-elle pas être intégrée dans cette méthode ?

Question 5
Donnez le code de la méthode `run` de la classe `MoteurVitre`.

Exercice 7 – La capote du Spider, le retour

Nous reprenons l'exercice du pliage de la capote du spider, et nous voulons maintenant exécuter l'opération complète : une fois les vitres baissées, le gestionnaire peut activer le pliage de la capote, que nous implémenterons comme un simple affichage. Une fois la capote repliée, le gestionnaire s'assure que chaque moteur remet la vitre qu'il pilote dans sa position initiale.

Question 1
Comment le gestionnaire peut-il savoir que les vitres sont en position basse ?

Question 2
Proposez une nouvelle implémentation de la classe `MoteurVitre` et de la classe `GestionnaireCapote`. Le gestionnaire devra arrêter les moteurs lorsque l'exécution complète du pliage de la capote est terminée.

Question 3
En supposant que l'opération de pliage de la capote prenne un certain temps, y aurait-il un intérêt à l'implémenter dans un thread différent de celui du gestionnaire ?