Programmation Multi-cœurs — TP 2
Synchronisation blocante

Exercice 1 (Les toilettes unisexes deadlock-free)**.**
Les toilettes unisexes peuvent être utilisées par les hommes et par les femmes, sous les contraintes suivantes :

1. il ne doit jamais y avoir en même temps des hommes et des femmes aux toilettes ;
2. il ne doit jamais y avoir plus de trois personnes simultanément aux toilettes.

Un programme modélisant le problème est disponible sous Madoc. Le comportement des personnes qui accèdent aux toilettes est déjà modélisé. Les toilettes doivent implémenter l'interface `Bathroom`. La solution proposée dans la classe `NonParallelBathroom` fonctionne mais empêche tout parallélisme : il ne peut jamais y avoir plus d'une personne aux toilettes. Vous devez régler ce problème en proposant vos propres implémentations de `Bathroom`.

a. En vous inspirant de l'implémentaion des sémaphores, imaginez et implémentez une solution deadlock-free.

b. Modifiez votre solution pour la rendre starvation-free. *Indice :* des humains bien élevés organiseraient probablement une file d'attente.

Exercice 2 (Le dîner des philosophes)**.**
Des philosophes sont assis autour d'une table ronde. Devant chaque philosophe se trouve un bol de riz. Entre chaque paire de bols voisins se trouve une baguette. Ces philosophes occupent la majorité de leur temps à réfléchir, mais entre deux périodes d'intense réflexion, un philosophe a besoin de se nourrir. Pour manger, il doit commencer par prendre sa baguette de gauche et sa baguette de droite. Il peut alors manger un peu de riz. Une fois rassasié, il repose les deux baguettes. Faisons l'hypothèse que chaque philosophe mange pendant un temps fini et que le bol contient tellement de riz qu'il en restera toujours.

Un programme modélisant le problème est disponible sous Madoc. Un philosophe est une instance d'une extension de la classe abstraite `AbstractPhilosopher`. Chaque philosophe connait son identifiant (sa position sur la table) et peut accéder à sa baguette de gauche et sa baguette de droite grâce aux méthodes de l'objet `DinnerTable table`. L'implémentation de la classe `StarvingPhilosopher` ne fonctionne pas.

a. Expliquez pourquoi l'implémentation fournie ne fonctionne pas.

b. En utilisant un théorème du cours, proposez une solution deadlock-free.

c. Implémentez et testez votre solution.

d. Cette solution est-elle starvation-free ?