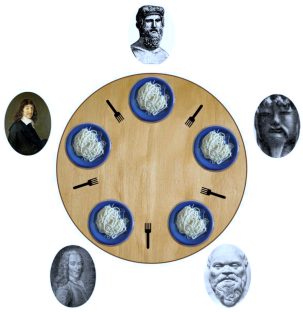
Énoncé par Edsger Dijkstra [1], le problème des philosophes et des spaghettis est un problème classique de partage de ressources en programmation parallèle. Le problème est le suivant. Cinq philosophes se trouvent autour d'une table ; chacun des philosophes a devant lui un plat de spaghetti ; à gauche de chaque plat de spaghetti se trouve une fourchette. Un philosophe n'a que trois états possibles :



① penser pendant un temps indéterminé ;

② être affamé ;

③ manger pendant un temps déterminé.

Lorsqu'un philosophe est affamé, il attend que les deux fourchettes devant lui soient libres. Dès qu'il s'aperçoit qu'une fourchette est libre, il la saisit. Dès qu'il dispose des deux fourchettes, il mange. Lorsqu'il a fini de manger, il pose les deux fourchettes et retourne penser. Mais il reviendra à table dès qu'il sera affamé. Un philosophe s'asseoit toujours à la même place : il mange donc toujours avec les mêmes fourchettes.

Exercice II.1 Modélisation des philosophes et des fourchettes Le système se compose de cinq philosophes et de cinq fourchettes.

Question 1. Écrire une classe **`Fourchette`** sous la forme d'un moniteur. Chaque objet de la classe **`Fourchette`** est disponible ou pris. C'est donc essentiellement un booléen qui fonctionne comme un verrou. La prise d'une fourchette s'effectue par la méthode **`prendre()`**, qui est bloquante si la fourchette est déjà prise, son relâchement par la méthode **`lacher()`**. Naturellement, deux philosophes ne peuvent pas tenir une même fourchette en même temps !

Question 2. Écrire une classe **`Philosophe`** qui hérite de la classe **`Thread`**. Chaque objet de cette classe dispose de deux fourchettes déposées à droite et à gauche de son assiette personnelle. Ces deux fourchettes lui sont indiquées lors de sa construction. Initialement un philosophe pense, *pendant un temps aléatoire*. Puis, il a faim et tente de saisir les deux fourchettes devant lui (dans un ordre quelconque). Lorsqu'il dispose des deux fourchettes, il mange pendant trois secondes, puis il retourne penser. Ce comportement cyclique sera décrit par la méthode **`run()`**.

Question 3. Écrire le programme principal qui installe cinq philosophes à table, avec cinq fourchettes.

Exercice II.2 Des méthodes atomiques

Question 1. Pourquoi la méthode **`prendre()`** de la classe **`Fourchette`** est-elle déclarée **`synchronized`** ? Quelle situation proscrite pourrait apparaître sinon ?

Question 2. Pourquoi la méthode **`lacher()`** est-elle aussi **`synchronized`** ? Quelle situation inattendue pourrait sinon être observée ?

Exercice II.3 Suppression de l'interblocage S'ils agissent tous de façon identique, les cinq philosophes risquent fort de se retrouver en situation d'interblocage. En effet, il suffit que chacun saisisse sa fourchette de gauche et, qu'ensuite, chacun attende que sa fourchette de droite se libère pour arriver à une situation où aucun philosophe ne peut plus jamais manger...

Question 1. Quelle approche classique, évoquée en cours, permet de supprimer l'interblocage potentiel de ce code ? Quelle ligne du programme donné sur la figure 6 peut-on modifier pour adopter cette autre approche ?

Question 2. Afin d'éviter tout interblocage, une autre approche consiste à limiter à quatre le nombre de philosophes qui sont autorisés à manger en même temps. Ceci garantit en effet qu'il y ait toujours au moins un philosophe qui peut manger. Modifiez la classe **`Philosophes`** de la figure 6 afin de limiter à 4 le nombre de philosophes qui essaient de manger en même temps. Intuitivement, il s'agit de placer quatre chaises autour de la table avec les contraintes suivantes : il faut s'asseoir pour manger et il faut se lever de sa chaise pour aller penser.

Exercice II.4 Des philosophes ambidextres Afin d'éviter l'interblocage, une solution alternative consiste à modifier le comportement des philosophes pour qu'ils saisissent les deux fourchettes « en même temps. » Modifiez les méthodes **`prendre`** et **`lacher`** de la classe **`Fourchette`** afin qu'elles permettent de prendre ou de lâcher deux fourchettes spécifiées en paramètre. Comment les philosophes doivent-ils s'adapter ?

[1] Edsger W. Dijkstra, « *Hierarchical ordering of sequential processes* », Acta Informatica, vol. 1, 1971, p. 115-138