Thème 3 - TD

Objectifs

— section critique
— Méthodes synchronisées

Exercice

Exercice 5 – Producteur/Consommateur : cas de base

Nous reprenons le problème du producteur/consommateur et nous voulons maintenant programmer un système complet : la zone de stockage est accédée à la fois par un producteur qui dépose des données et par un consommateur qui les retire.

Question 1
Complétez la classe `Buffer` écrite dans le cas du producteur simple pour permettre des retraits. Lorsqu'un retrait n'est pas possible, un message d'erreur doit être affiché. Vous ajouterez aussi un accesseur permettant de connaître le nombre d'éléments présents dans le stock.

Question 2
Écrivez une classe `Consommateur` qui permet d'exécuter le consommateur dans un thread séparé. On supposera que le nombre de retraits effectués est le même que le nombre de dépôts et correspond à la taille du stock.

Question 3
Écrivez un programme de test permettant l'exécution concurrente du producteur et du consommateur. Le programme fonctionne-t-il de manière satisfaisante (i.e., pas de retrait dans une case vide ou de dépôt dans une case pleine) ?

Question 4
Proposez une solution pour résoudre le problème mis en évidence à la question précédente. Le tampon est-il vide à la fin de l'exécution ?

Pour garantir que les éléments déposés sont retirés, nous nous proposons maintenant de construire une solution dans laquelle les opérations demandées sont bloquantes : un appel à la méthode `depot` (resp. `retrait`) ne se termine que lorsque la valeur a effectivement pu être écrite (resp. lue) dans le stock.

Tant que la condition pour exécuter une opération n'est pas remplie, le processus qui la demande est "mis en sommeil" (appel à la méthode `sleep` de la classe `Thread`) avant de pouvoir retenter sa chance. Il sort de ces sommeils successifs lorsque l'état du stock rend l'opération possible. Nous imposons en outre que les accès aux cases du tampon se fassent dans l'ordre : le producteur remplit successivement les cases $0, 1, \ldots, n-1, 0, \ldots$ et le consommateur les vide dans le même ordre.

Question 5
Proposez une nouvelle version de la méthode `depot` dans laquelle, lorsque le producteur ne peut pas produire, il est mis en sommeil pendant 10 millisecondes avant de refaire une tentative.

Ajoutez aussi un constructeur à la classe producteur permettant de fixer, à la création du producteur, le nombre de dépôts qu'il va effectuer.

Question 6
Proposez une version du retrait “symétrique” de celle que vous venez d’écrire pour le dépôt. Ajoutez aussi un constructeur au consommateur permettant de fixer le nombre d’opérations qu’il exécute.

Question 7
Modifiez le programme de test pour que le producteur et le consommateur effectuent un nombre d’opérations identique, mais supérieur au nombre de cases du tampon.

Est-il nécessaire d’ajouter un mécanisme d’exclusion mutuelle pour que ce programme fonctionne de manière satisfaisante ? Qu’en est-il si on a plusieurs producteurs et/ou plusieurs consommateurs ?

Question 8
Proposez un mécanisme permettant de garantir la cohérence des accès au stock, même lorsqu’il y a plusieurs producteurs et plusieurs consommateurs.