PROJET

Consignes du projet

- Le projet se fera par **des groupes de 3 ou 4** étudiants**.**
- Le rapport doit être concis, imprimer en **recto-verso et sans reliure**. Il comprendra : l'objectif du projet, le diagramme des classes, la description des classes, l'explicitation et la justification des choix techniques, une documentation utilisateur de l'utilisation de l'application, et les copies écran de l'exécution. En annexe du rapport, le code Java des exercices des TPs.
- Les **soutenances** auront lieu la semaine du **16/01/2012**
- La présentation du projet lors de la soutenance ne devra pas excéder **cinq minutes**
- Le **rapport** doit être **rendu le 04/01/2012 avant 12h** en **B526**
- Le **code** source des classes (.zip ou .tar) doit être **rendu le 03/01/2012** par email à l'adresse haddadjoyce@gmail.com. **Le sujet de l'email fera apparaître : 2012/M1/ProjetSAR/Noms des 3 ou 4 étudiants**

Introduction

Tous les algorithmes de contrôle de flux vu en cours supposent que le tampon est géré par le site consommateur. Il s'agit dans ce projet de mettre en place une implantation répartie d'un nouvel algorithme de contrôle de flux qui soit une généralisation du modèle producteur-consommateur dans lequel un site T est responsable de la gestion du tampon, p sites de production et k sites de consommation.

Afin de résoudre le problème dû à l'asynchronisme des sites de production et des sites de consommation, l'algorithme doit asservir les producteurs et les consommateurs des messages par des informations fournies par le site T. Pour cela, chaque site, producteur ou consommateur, communique avec le site T.

L'algorithme consiste à faire circuler des autorisations de production entre les producteurs et le site T et des autorisations de consommation entre les consommateurs et le site T.
Lorsqu'un producteur veut déposer un message dans le tampon, il appelle `Produire(m)`.
Lorsqu'un consommateur veut retirer un message du tampon, alors il appelle la primitive `Consommer()`.

Un producteur qui souhaite produire un message enverra une demande de production au site T. En réponse, le site T envoie soit une autorisation de production signifiant qu'au moins une place est libre dans le tampon soit un refus signifiant que le tampon est plein. Dans le premier cas, le site producteur envoie le message produit à T.

Un consommateur qui souhaite retirer un message enverra une demande de consommation au site T. En réponse, le site T envoie un message si le tampon n'est pas vide ou un refus dans le cas contraire.

Bien entendu, les consommateurs ne pourrons consommer que si le tampon n'est pas vide et les producteurs ne pourront produire que si le tampon n'est pas plein. On suppose que le site T gère le tampon de manière circulaire.

L'application Serveur : Le site T

Elle suppose l'existence d'un objet réparti, le tampon : il s'agira d'un tableau de taille donnée en paramètre lors de la construction de l'objet. Deux méthodes permettent d'accéder à l'objet:

1. `insérer (message)` // dépôt d'un objet dans le tampon ;
2. `extraire ()` // extraction d'un objet du tampon ;

D'autres méthodes peuvent être nécessaires.

Etape 1 : Spécifier l'interface de l'objet

Il s'agit de spécifier sous la forme d'une interface java **l'ensemble** des méthodes qui pourront être invoquées sur l'objet. Cette interface, `TamponInterface`, sera connue à la fois du côté du client et du côté du serveur.

Etape 2 : Implanter l'interface du côté serveur

Cette implantation, sous forme d'une classe `TamponInterfaceImpl`, permettra l'instanciation d'objets sur lesquelles des méthodes seront invocables à distance par les producteurs et les consommateurs.

Dans la classe `TamponInterfaceImpl` implémentant l'interface `TamponInterface`, différents objets seront utilisés :

- un `tableau d'objets` dont le nombre d'éléments est fixé lors de l'instanciation de la classe et qui correspond au tampon;
- des entiers `taille`, `in`, `out` et `nbmess` qui correspondent respectivement à la taille du tampon, l'emplacement à utiliser lors du prochain dépôt, l'emplacement à utiliser lors de la prochaine extraction et le nombre d'éléments actuellement dans le tampon.

La classe contient par ailleurs l'implémentation des deux méthodes `insérer` et `extraire`. L'implémentation d'autres méthodes peuvent être nécessaires.

Etape 3 : Définir l'objet afin de le rendre visible

C'est le premier rôle de l'application serveur. Un second est de faire connaître l'existence de l'objet par l'intermédiaire d'un service de nommage.

Implantation, de la classe `ServeurTampon`, d'un serveur de tampon.

L'application Client : les Producteurs et les Consommateurs

Réaliser les appels distants au travers d'applications clientes comme suit :

1. La classe `Producteur` : Le tampon est partagé par les différents producteurs. La méthode `Produire` permet de déposer un objet dans le tampon. La méthode run implante une boucle de demandes de dépôt dans le tampon. Entre chaque dépôt le thread est endormi durant une période de durée aléatoire.

2. La classe `Consommateur` : Le tampon est partagé par les différents consommateurs. La méthode `Consommer` permet de retirer un objet du tampon. La méthode run implante une boucle de demandes de lecture dans le tampon. Entre chaque lecture le thread respecte une période de sommeil de durée aléatoire.
3. Une classe principale : la méthode statique main de cette classe a comme paramètre le nom de l'objet et en second paramètre optionnel l'adresse de la machine abritant le serveur (et l'objet) [la machine locale est considérée si le second paramètre est omis]. Elle crée un certain nombre d'instances des classes `Producteur` et `Consommateur` (au moins deux de chacune des classes).

Génération des fichiers nécessaires à l'application

Générer les fichiers .class et construire le talon et le squelette nécessaires à l'application.
Tester l'exécution de l'application producteurs/consommateurs répartie sur deux machines distantes.

Bonus

La réalisation d'une interface graphique sera un plus.