TP Noté 2009 - 2 h

PAR : Programmation d'Applications Réparties
Troisième année

Tous documents autorisés. La qualité de rédaction de votre code, des commentaires associés et du compte-rendu est aussi importante que l'exactitude du code.
Pensez à indiquer dans votre compte rendu les commandes à lancer pour exécuter votre projet, ainsi que l'endroit depuis où elles doivent être lancées.

Comment rendre votre copie

Vous devez rendre un compte-rendu avec vos sources. Un simple fichier texte décrivant votre travail et argumentant vos choix suffit, mais vous pouvez faire un .pdf si vous préfèrez (pas de .doc .odt ou autre). Quand vous avez fini votre TP, lancez le script suivant depuis neptune :

`/home/EqPedag/quinson/bin/rendre_projet PAR`

★ Exercice 1. Service d'agenda simple.

On souhaite réaliser un service (simpliste) de calendrier partagé. Il est composé d'un serveur central sur lequel un objet `CalServeur` est invocable à distance. L'interface de cet objet est donnée plus bas. Chaque calendrier d'utilisateur (créé par un appel distant à CalServeur.createCal) est représenté par un objet de type `Cal` (placé sur le serveur).

Pour simplifier l'implémentation de la classe `Cal`, on ne représente pas des rendez-vous heure par heure, mais simplement si les gens sont libres un jour donné.

CalServeurInterface

```
1 package exam;
2 import java.rmi.Remote;
3
4 public interface CalServeurInterface extends Remote {
5   public CalInterface createCal(String nom) throws RemoteException;
6 }
```

CalInterface

```
1 package exam;
2 import java.rmi.Remote;
3
4 public interface CalInterface extends Remote {
5   public Boolean estLibre(int jour) throws RemoteException;
6   public Boolean libere(int jour) throws RemoteException;
7   public Boolean occupe(int jour) throws RemoteException;
8 }
```

▷ **Question 1.** Écrivez les classes `CalServeur` et `Cal` implémentant les interfaces données. Vous êtes libre d'ajouter aux classes les champs nécessaires à cela. En particulier, il est conseillé de doter `Cal` du champ `boolean jourLibre[365]`

▷ **Question 2.** Écrivez une classe de test cliente créant un ou plusieurs calendriers sur le serveur, et testant les différentes fonctionnalités.

★ Exercice 2. Services centraux.

▷ **Question 1.** On souhaite maintenant ajouter un service permettant de vérifier si un groupe de personne est libre à un instant donné. Ajoutez pour cela à la classe `CalServeur` la methode suivante : `boolean intersectionLibre(String[ ] nom, int jour)` Vous pourrez être amenés à modifiez le reste de la classe.

▷ **Question 2.** Implémentez une méthode à `CalServeur` pour poser une réunion à un groupe de personne.
`void reserve(String[ ] nom, int jour)`

En cas de conflit d'emploi du temps, une exception adéquate devra être levée.

Il est rappelé que les objets RMI s'exécutent en parallèle les uns des autres, et votre solution devra tenir compte de la possibilité qu'une personne peut être disponible au début de l'exécution de `reserve()`, et être retenue par ailleurs en cours d'exécution (par exemple parce qu'un autre client a fait un appel à `occupe()` de cet objet entre temps).

▷ **Question 3.** Augmentez votre classe de test pour tester les nouvelles fonctionnalités.

★ **Exercice 3. Services aux clients.**

On souhaite maintenant que les clients puissent «s'enregistrer» à un ou plusieurs calendriers. S'ils le font, un message s'affichera sur la sortie standard de leur programme chaque fois que le calendrier auquel ils sont abonnés est modifié sur le serveur.

▷ **Question 1.** Implémentez cette fonctionnalité, et augmentez votre classe de test pour vérifier son bon fonctionnement.