1 TD6 : RMI

Dès votre arrivée en TD, pensez à déposer votre QCM rempli à votre encadrant. La partie à rendre est la dernière page de votre document (à découpe suivant les pointillés).

Après l'appel, c'est l'occasion de poser les questions concernant des éléments non compris du cours. Pensez à recopier le corrigé du QCM sur votre document (partie à concerver à la page précédante). Vous pourrez l'utiliser le jour de l'examen. Une page de notes personnelles est à votre disposition en fin de TD.

Ce TD porte sur la réalisation d'un programme réparti utilisant l'appel de méthode à distance de Java (RMI pour *Remote Method Invocation*). Le programme consistera en un serveur de vote du genre sélection de date commune pour une réunion.

1.1 compréhension de RMI

La compréhension de RMI demande des heures de travail. Nous ne ferons durant ce TD qu'un survol pour permettre la réalisation d'un appel à distance.

L'objetif est de simuler le même comportement localement et à distance. Pour cela, on décrit le comportement local (voir figure 1). Le serveur définit une interface qu'il s'engage à réaliser.

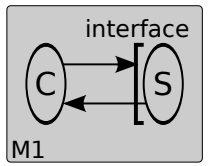

FIGURE 1 – Client local

Le but de RMI est de remplacer les appels locaux en appels à distance via des objets de liaison. C'est donc la même interface que le client utilisera sans se rendre compte qu'il fera appel à un objet de liaison.

Les objets de liaison communiqueront via un socket (voir figure 2).

Pour retrouver ces objets un dépositaire d'interfaces est utilisé : le *rmiregistry*.

L'utilisation de RMI se fait en 2 temps :

- lors de la compilation, on indique l'interface du serveur qu'il faudra pouvoir traiter.
- au moment de l'exécution, la recherche d'un serveur fournira soit le serveur lui-même soit un représentant local. Dans tous les cas, le client doit passer par l'interface qui garantit la transparence de traitement.

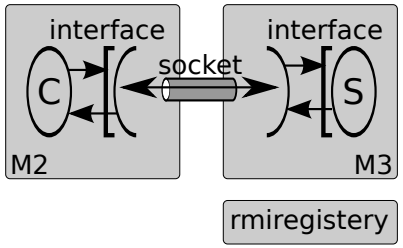

FIGURE 2 – Client distant

1.1.1 Structure du serveur

Il faut définir la liste des opérations offertes par le serveur. Par exemple :

```
public interface ICalc extends Remote {
    public int add (int a, int b) throws RemoteException;
}
```

Notez l'héritage de l'interface `Remote` et le fait que toutes méthodes déclancheables à distance peut engendrer un problème de communication `throws RemoteException`.

Ensuite, le serveur doit proposer cette interface. Il doit également définir un mode de communication par socket comme `UnicastRemoteObject`.

```
public class CalcImpl extends UnicastRemoteObject implements ICalc {

    public CalcImpl () throws RemoteException {
        super ();
    }

    public int add (int a, int b) throws RemoteException {
        System.out.println ("Addition de " + a + " et " + b);
        return a+b;
    }
}
```

Le constructeur doit lui aussi pouvoir lever une exception en cas de problème de communication `throws RemoteException`.

Nous nous limiterons dans ce TD à l'utilisation de type de base java pour les paramètres formels des méthodes. L'utilisation d'objets est possible suivant certaines contraintes.

Enfin, le serveur doit être déclaré auprès d'un dépositaire d'interface (`rmiregistry`) sur le même ordinateur qui exécute la machine virtuelle Java du serveur.

```
    public static void main (String args[]) {
        System.setSecurityManager (new RMISecurityManager ());
        try {
            ICalc obj = new CalcImpl ();

            Naming.rebind ("rmi://"+args[0]+"/CalcServeur", obj);
            System.out.println ("CalcServeur enregistre");
        } catch (Exception e) {
            System.out.println ("CalcServeur err : "+e);
        }
    }
```

Reportez-vous à la documentation de `Naming` pour plus d'informations. Sachez que cet annuaire est en relation avec le dépositaire qui est exécuté dans le processus `rmiregistry`.

L'URL fournie est de la forme `rmi://host/CalcServeur`, avec `host`, un nom de machine ou une adresse IP suivie éventuellement d'un numéro de port.

1.1.2 Structure du client

Le client va commencer par interroger le dépositaire ce qui provoquera la création d'un objet de liaison de même interface que le serveur dans sa machine virtuelle.

```
public static void main (String args[]) {
System.setSecurityManager (new RMISecurityManager ());

try {
ICalc od = (ICalc) Naming.lookup ("rmi://"+args[0]+"/CalcServeur");
System.out.println (od.add (Integer.parseInt (args[1]),
Integer.parseInt (args[2])));
} catch (Exception e) {
System.out.println ("CalcClient Err : " + e);
}
}
```

Ensuite le client manipule l'objet trouvé comme s'il s'agissait d'un serveur local.

1.1.3 Compilation

Au moment de la compilation, il faudra créer les objets de liaison avec la commande `rmic`.

```
# compilation serveur et client
cd src
javac -d ../class *.java

# création des objets de liaison
cd ../class
rmic -d . CalcImpl
```

Il faut également établir une stratégie de sécurité pour autoriser les connexions réseau. Par exemple en créant le fichier "policy.txt".

```
grant {
// Allow everything for now
permission java.security.AllPermission;
};
```

1.1.4 Exécution

Sur la machine du serveur, il faut lancer le dépositaire et enregistrer le serveur.

```
cd class
rmiregistry &
java -Djava.security.policy=policy.txt CaclServer
```

Sur la machine du client, il faut lancer le client

```
cd class
java -Djava.security.policy=policy.txt CaclClient
```

1.2 Compréhension du sujet

Le logiciel résultant de ce TD possède les propriétés suivantes :

- le serveur est configuré pour un nombre de choix et un nombre de votants.
- le serveur gère le nom des clients ayant voté
- le serveur permet de voter en indiquant un nom de votant et un choix (un booléen indique si le vote est recevable).
- le serveur permet de connaître le taux de participation (réel).
- le serveur permet de connaître le pourcentage de vote pour chacun des choix (chaîne de caractères).

1.3 Analyse du logiciel

Question : Quelles sont les opérations du serveur ? Donnez l'interface

Réponse : ✍

Question : Quelles sont les données gérées par le serveur ?

Réponse : ✍

Question : Ecrivez le constructeur du serveur.

Réponse : ✍

Question : Ecrivez le calcul du taux de participation.

Réponse : ✍

Question : Ecrivez l'affichage du pourcentage de résultat par choix dans une chaîne de caractères.

Réponse : ✍

Question : Ecrivez un lanceur qui crée et enregistre un serveur de vote.

Réponse : ✍

Question : Ecrivez un client qui prend un nom et un choix de vote.

Réponse : ✍

Page de notes personnelles : ✍

2 TP6 : RMI

- Votre compte-rendu est à rendre en fin de TP.
- Vous ne devez pas rendre un listing, mais un document répondant aux questions posées.
- Vous préparerez le compte-rendu à l'avance (copier/coller les questions) pour gagner du temps.
- Vous rappelerez vos prénoms et noms, groupe TD, date et identifierez clairement le TP.
- Vous recopierez le texte des questions (et numéro) à chaque fois.
- Vous répondrez par des phrases.
- Vous recopierez les commandes que vous avez saisies pour obtenir un résultat et plus généralement ce que vous saisissez dans un terminal.
- Quand c'est nécessaire, vous pouvez faire une copie d'écran (uniquement la fenêtre concernée).
- N'oubliez pas de rendre votre TP imprimé en 2 colonnes de texte par côté de papier et de faire le TP en binôme.

2.1 Validation des outils

Téléchargez les programmes `ICalc.java`, `CalcImpl.java`, `CalcServeur.java`, `CalcClient.java` et testez-les.

2.2 Réalisation du logiciel

Consultez l'API Java concernant les classes : TreeSet, Naming

Question : Écrivez l'interface de vote.

Réponse : ✍

Question : Écrivez le serveur avec des méthodes vide ou qui rends une valeur constante.

Réponse : ✍

Question : Écrivez le client.

Réponse : ✍

Question : Écrivez le lanceur.

Réponse : ✍

Question : Vérifiez le fonctionnement. Quelles sont les lignes de compilation ?

Réponse : ✍

Question : Ajoutez la méthode de calcul du taux de participation.

Réponse : ✍

Question : Ajoutez la méthode de présentation du résultat du vote.

Réponse : ✍

Question : Ajoutez la méthode de calcul du taux de vote.

Réponse : ✍

Question : Ajoutez la méthode de vote.

Réponse : ✍

2.3 Test du logiciel

Question : Donnez des exemples de fonctionnement.

Réponse : ✍

Question : Quelles sont les limites ? Quelles sont les améliorations à apporter ?

Réponse : ✍