tp5

Lectures préalables :	Thèmes du TP :
• Le tutorial de Sun extrait de la documentation du jdk /docs/guide/rmi/index.html • Le support de cours	Question 1 • Services auprès de clients Question 2 • un "chat"

- **Visualisez le sujet en ouvrant `index.html` du répertoire qui a été créé à l'ouverture de `tp5.jar` par BlueJ; vous aurez ainsi accès aux différents liens qui sont proposés pour vous aider, et aux applettes.**
- **Soumettez chaque question à l'outil d'évaluation jnews.**

Accès aux propriétés-Java de la machine hébergeant le serveur RMI

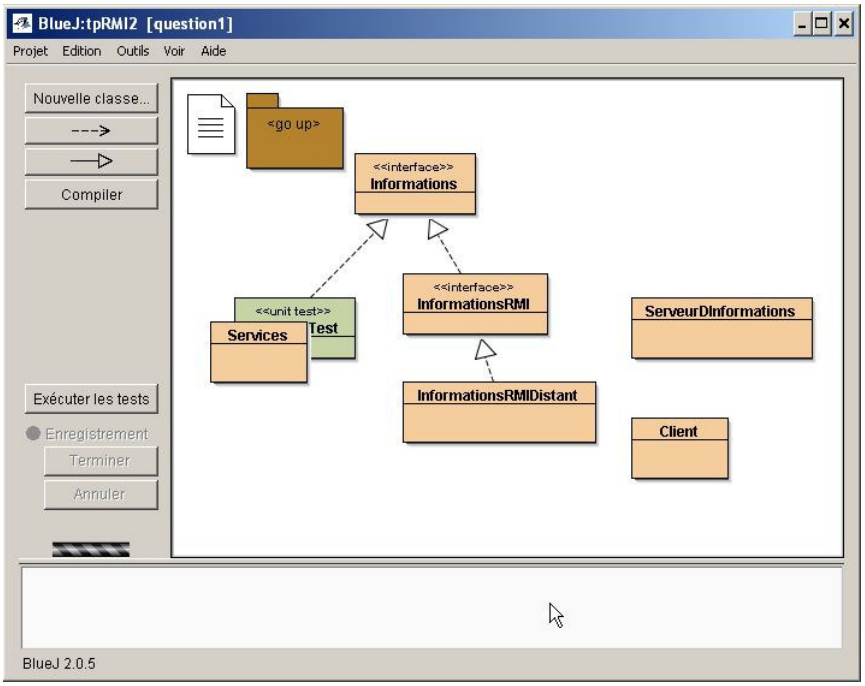
question1) Les clients souhaitent obtenir l'heure et certaines propriétés du serveur. Proposer une implémentation de ces services selon le protocole rmi: en respectant cette interface :

```
public interface Informations{

 public String getDate() throws Exception;
 public Properties getProperties() throws Exception;
}
```

- Seules les classes InformationsRMIDistant, ServeurDInformations et Client sont à compléter

- Une première version "locale", puis une seconde "répartie" avec l'usage des classes du paquetage java.rmi devront être proposées, le pattern Adaptateur sera utilisé, voir le support cité dans l'encadré. Les clients se contenteront d'afficher l'heure du serveur ainsi que le type du processeur (os.arch) et le type du système d'exploitation (os.name).



Un diagramme des classes en notation BlueJ/UML

un exemple de trace sur la console : deux clients successifs du serveur **rmi://localhost:1099/informations/**

```
le serveur: informations a demarre
16/03/05 19:15
x86
Windows 2000
16/03/05 19:15
x86
Windows 2000
```

Les commandes depuis une console ici sous windows, la variable d'environnement PATH contient le chemin des exécutables du jdk

exemple : c:\> set PATH=c:/jdk1.6/bin;%PATH%

exécution du service de nommage, par défaut port 1099	**c:\> start rmiregistry**
depuis le répertoire de ce tp, démarrage du serveur de .class, ici sur le port 8086	**c:\tp_rmi\> start java -cp . ServeurWeb8086**
démarrage du service RMI	**c:\tp_rmi\> start java -cp . -Djava.security.policy=policy.all -Djava.rmi.server.codebase=http://localhost:8086/ question1.ServeurDInformations**
exécution d'un client	**c:\tp_rmi\> start java -cp . -Djava.security.policy=policy.all question1.Client**

extraits de la documentation de SUN

`java.rmi.server.codebase`
This property specifies the locations from which classes that are published by this JVM (for example, custom classes that implement an interface that is the declared parameter type of a remote method call) may be downloaded. The value of this property is a a space-separated list of URLs that will be the codebase annotation for all classes loaded from the CLASSPATH of (and subsequently marshalled by) this JVM.

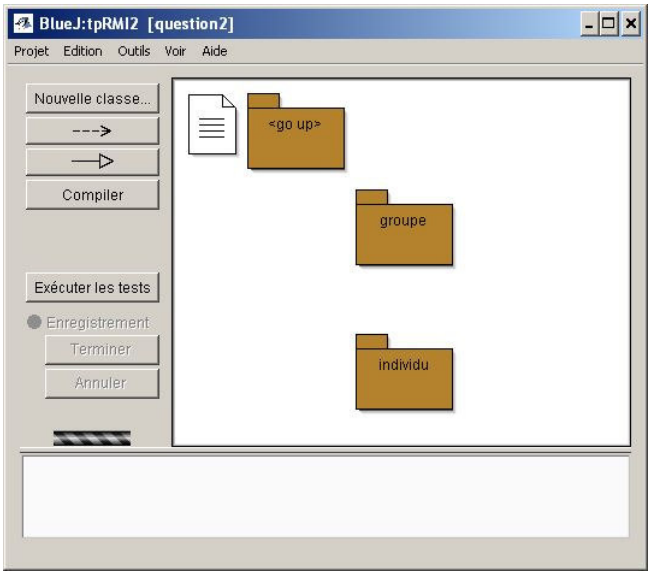
`java.rmi.server.useCodebaseOnly`
If this value is `true`, automatic loading of classes is prohibited *except* from the local CLASSPATH and from the `java.rmi.server.codebase` property set on this JVM. Use of this property prevents client JVMs from dynamically downloading bytecodes from other codebases.

The `java.security.policy` property is used to specify the policy file that contains the permissions you intend to grant.
policy.all du tp

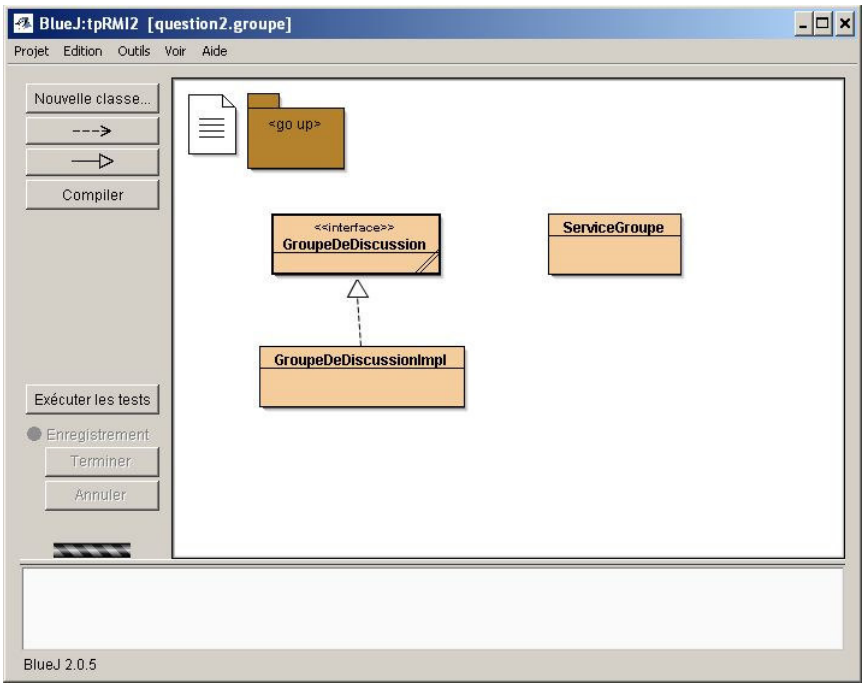
```
grant {
permission java.security.AllPermission "", "";
};
```

un Chat avec RMI

question2) Complétez toutes les classes afin de disposer d'un "chat" élémentaire au protocole RMI, voir l'annexe 3 du support



Le groupe de discussion est un service RMI

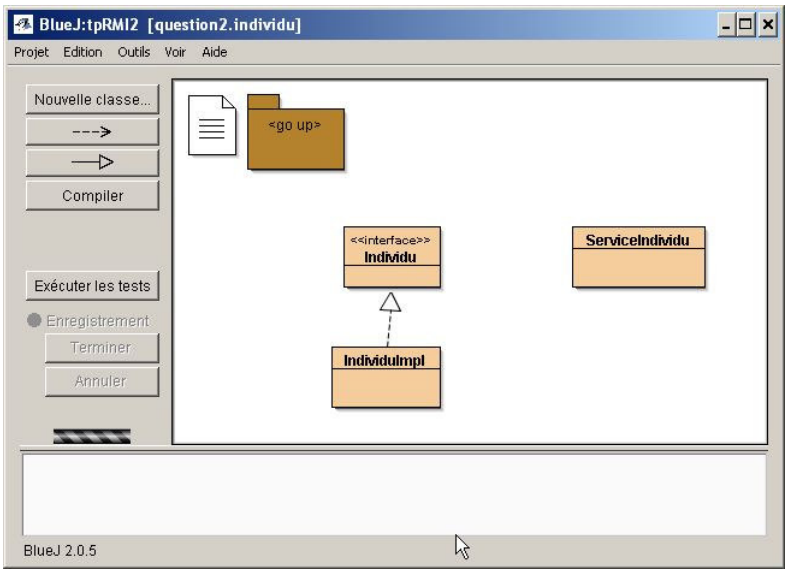



```
public interface GroupeDeDiscussion extends Remote{

  public void participer(Individu individu) throws RemoteException;
  public void sortir(Individu individu) throws RemoteException;

  public void saluer(Individu individu) throws RemoteException;
  public void parler(Individu individu, String phrase) throws RemoteException;
  public void chuchoter(Individu source, String phrase, Individu destinataire) throws RemoteExceptior

  public Set<Individu> listeDesParticipants() throws RemoteException;
}
```



L'"individu" est un également un service RMI

```
public interface Individu extends Remote, Serializable{

  public String nom() throws RemoteException;

  public void entendre(String phrase) throws RemoteException;


}
```

Testez votre "chat" en utilisant plusieurs machines avec plusieurs groupes et plusieurs clients