**Note :**

**Il est important de noter que RMI est supprimé de jdk depuis jdk 17.**

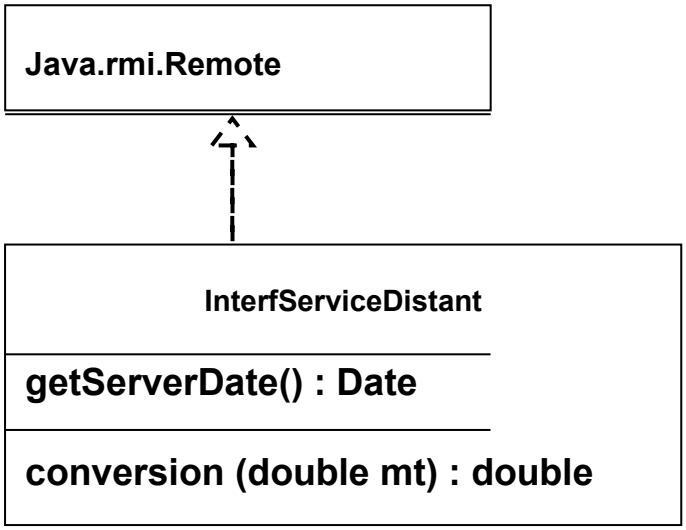
**Il est recommandé d'installer Java SE 8 (JDK 8).**

# TD 2 Java Remote Method Invocation (RMI)

Supposant qu'on veut créer un serveur RMI qui crée un objet qui offre les services distants suivant à un client RMI:

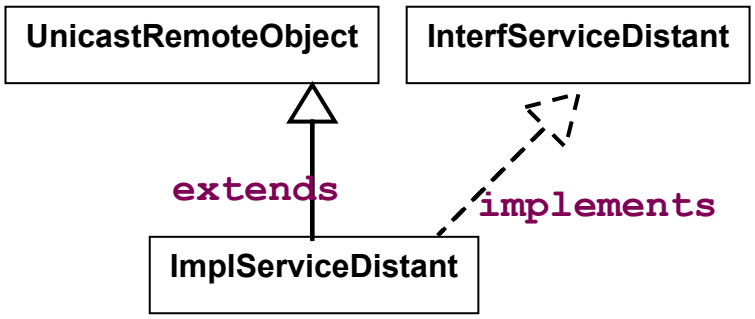
- Convertir un montant de l'euro en DH
- Envoyer au client la date du serveur.

## 1. Interface de l'objet distant




```
import java.rmi.Remote;import java.rmi.RemoteException;
import java.util.Date;
public interface InterfServiceDistant extends Remote {
  public Date getServerDate() throws RemoteException;
  public double convertEuroToDH(double montant) throws
RemoteException;
}
```

## 2. Implémentation de l'objet distant



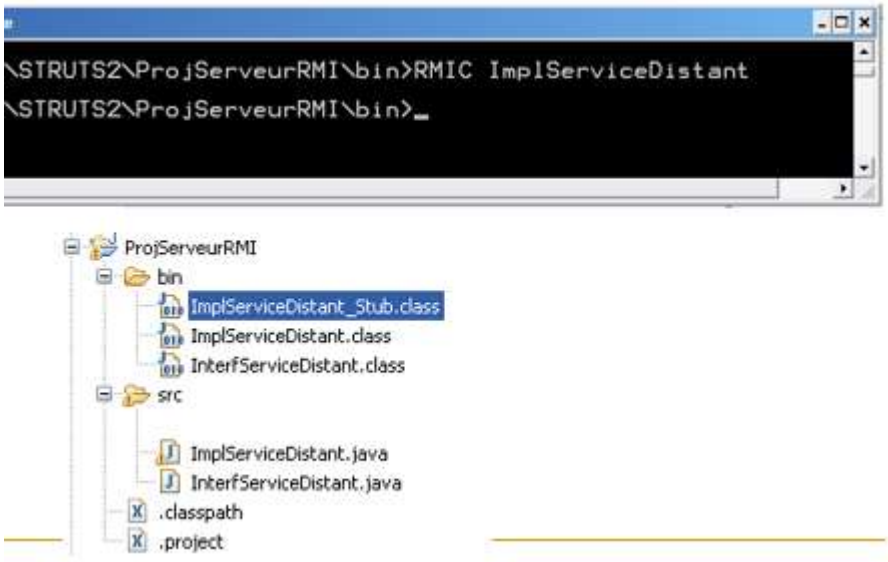

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;
public class ImplServiceDistant extends UnicastRemoteObject implements
    InterfServiceDistant {
    public ImplServiceDistant()throws RemoteException{

    }
public Date getServerDate() throws RemoteException {
return new Date();
}
  public double convertEuroToDH(double montant) throws RemoteException
    {
return montant*11.3;
}
}
```

## 3. STUBS et SKELETONS

Si l'implémentation est créée, les stubs et skeletons peuvent être générés par l'utilitaire « rmic » en écrivant la commande :

`Rmic NOM_IMPLEMENTATION ou rmic -classpath . NOM_IMPLEMENTATION`




## 4. Naming Service

Les clients trouvent les services distants en utilisant un service d'annuaire activé sur un hôte connu avec un numéro de port connu.

- RMI peut utiliser plusieurs services d'annuaire, y compris Java Naming and Directory Interface (JNDI).
- Il inclut lui-même un service simple appelé (rmiregistry).
- Le registre est exécuté sur chaque machine qui héberge des objets distants (les serveurs) et accepte les requêtes pour ces services, par défaut sur le port 1099.

Dans le cas où le système Windows ne reconnaît pas le service, elle peut être lancé manuellement avec la commande : `rmiregistry`

## 5. Serveur

- Notre serveur doit enregistrer auprès du registre RMI l'objet local dont les méthodes seront disponibles à distance.
- Cela se fait grâce à la méthode statique `bind()` ou `rebind()` de la classe Naming.
- Cette méthode permet d'associer (enregistrer) l'objet local avec un synonyme dans le registre RMI.
- L'objet devient alors disponible par le client.

```
ObjetDistantImpl od = new ObjetDistantImpl();
Naming.rebind("rmi://localhost:1099/NOM_Service",od);
```

### 5.1 Code du serveur RMI

```
import java.rmi.Naming;
public class ServeurRMI {
    public static void main(String[] args) {
     try {
// Créer l'objet distant
ImplServiceDistant od=new ImplServiceDistant();
// Publier sa référence dans l'annuaire
Naming.rebind("rmi://localhost:1099/SD",od);
     } catch (Exception e) {
          e.printStackTrace();
     }
    }}
```

## 6. Client

Le client peut obtenir une référence à un objet distant par l'utilisation de la méthode statique `lookup()` de la classe Naming.

### 6.1 Code du Client RMI

```
import java.rmi.Naming;
public class ClientRMI {
    public static void main(String[] args) {
     try {
InterfServiceDistant stub=
    (InterfServiceDistant)Naming.lookup("rmi://localhost:1099/SD");
System.out.println("Date du serveur:"+stub.getServerDate());
System.out.println("35 euro vaut "+stub.convertEuroToDH(35));
     } catch (Exception e) {
          e.printStackTrace();
     }}}
```

## 7. Compilation

Compilation du serveur Server.java ***javac*** *-cp . ServeurRMI.java*

Compilation du client Client.java ***javac*** *-cp . ClientRMI.java*

## 8. Génération du Stub coté client

***rmic*** *-classpath . ImplServiceDistant*

## 9. Lancement

- Lancement de l'annuaire *rmiregistry*
- Lancement du serveur ServerRMI *java -cp . ServeurRMI*
- Lancement du client ClientRMI *java -cp . ClientRMI*

Note :

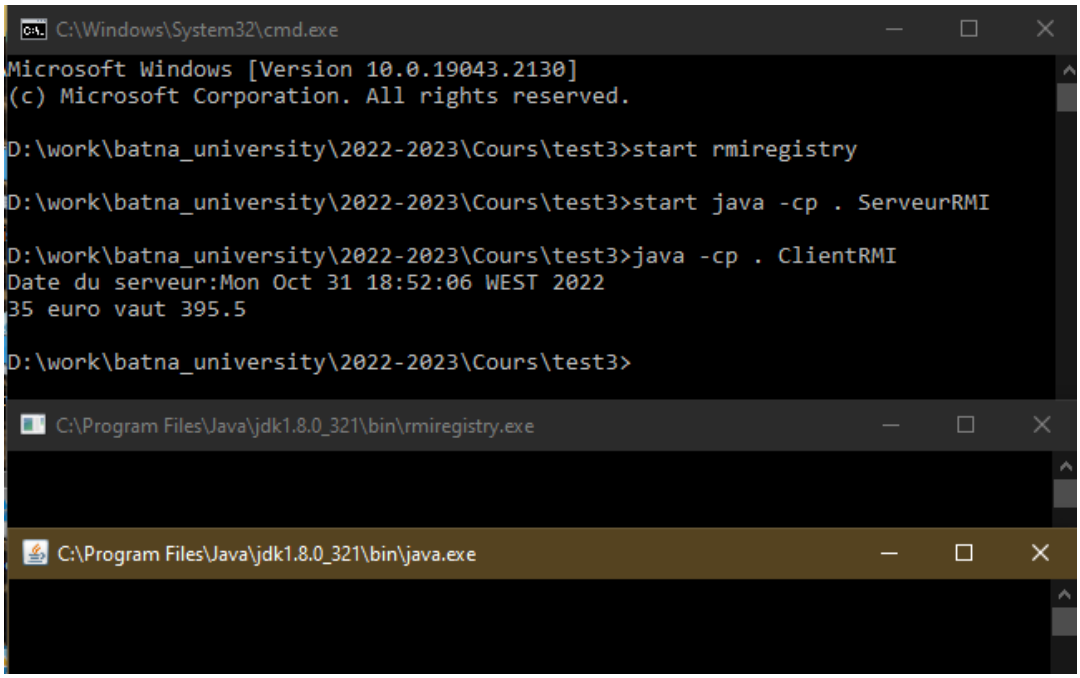
Dans le CMD Windows, vous pouvez utiliser le mot "start" avant la commande de lancement pour l'ouvrir dans une fenêtre séparée.

Example:

*start rmiregistry*

*start java -cp . ServeurRMI*

*start java -cp . ClientRMI*




## 10. TP 2 à domicile

Refaire l'exercice précédent pour permettre à un client java d'invoquer une méthode d'un objet distant (java) et récupérer le résultat qui fait l'addition de deux nombres de même type.

***Remarque**: Le TP à faire à domicile et à rendre à bentahar.info@aol.com avant le **20/11/2022.***