**Compte Rendu**

**TP6-1**

**Nour Bouguerra**

**LSI3 1TP2**

Le but de cette activité est de créer une application répartie permettant d’offrir un service de convertisseur de devise en utilisant le middleware CORBA. Un serveur permettra à des clients de se connecter et d’effectuer des opérations de conversions de

montants en euro aux dinars tunisiens. Cet atelier se déroule en 4 phases:

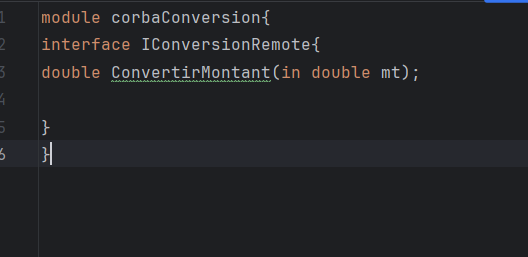
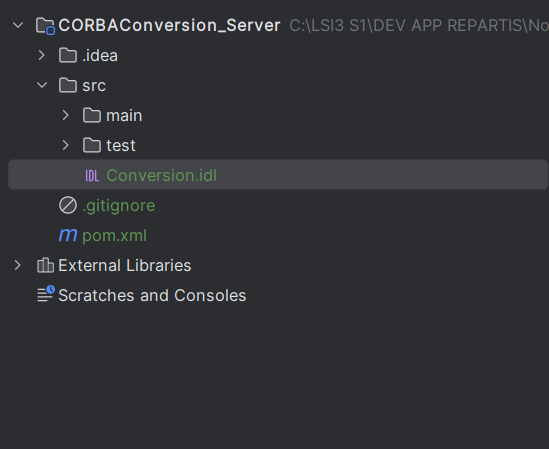
Description de la 2éme phase :

1. **Création du Projet Serveur :**
   * Un nouveau projet Java, nommé "CORBAConversion\_Server", est créé dans l'environnement de développement.

En synthèse, cette phase prépare le serveur en créant l'interface IDL, en générant les fichiers Java correspondants, en implémentant le serviteur de l'objet distant, et en configurant l'ORB et l'annuaire JNDI. Ces étapes sont cruciales pour mettre en place une infrastructure CORBA permettant aux clients de se connecter et d'effectuer des opérations de conversion de devises.

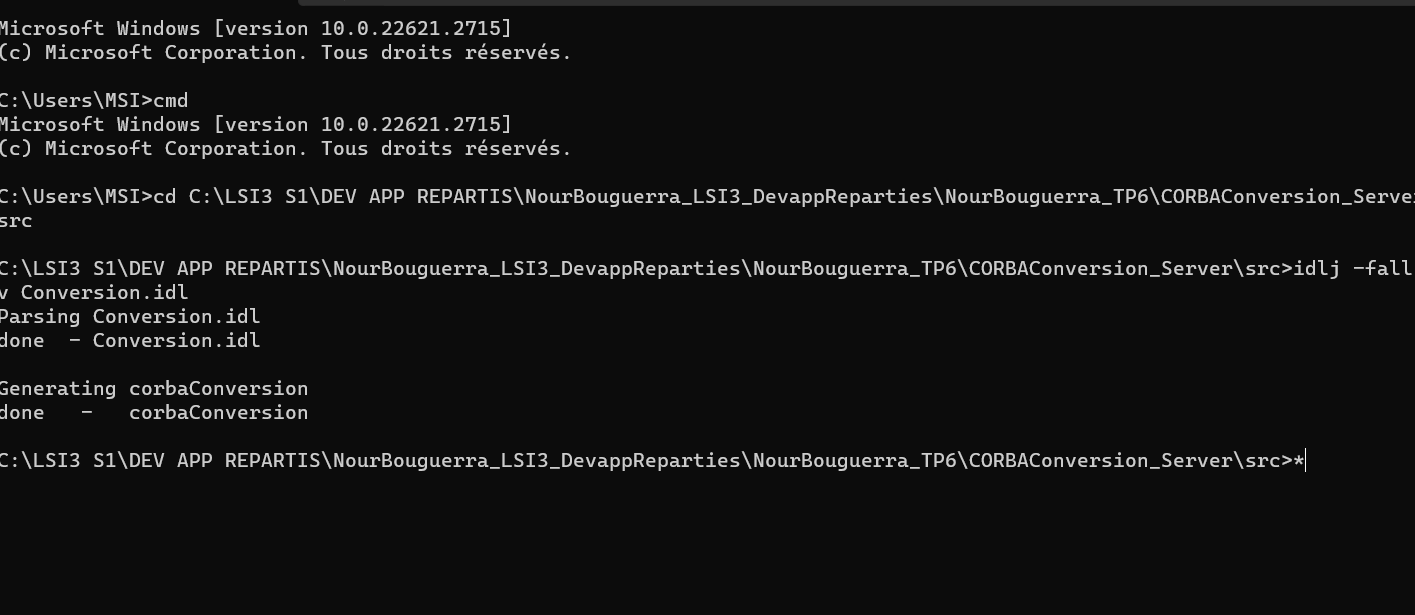
**2.Écriture de l'Interface en IDL :**

* + Un fichier IDL, nommé "Conversion.idl", est créé dans le répertoire src du projet.
  + Ce fichier déclare un module "corbaConversion" contenant une interface "IConversionRemote".
  + L'interface a une méthode abstraite "conversionMontant" prenant un paramètre "mt" de type float et retournant un double.



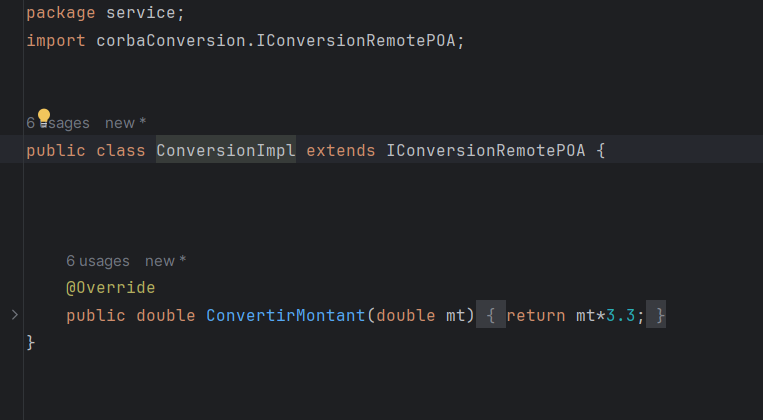
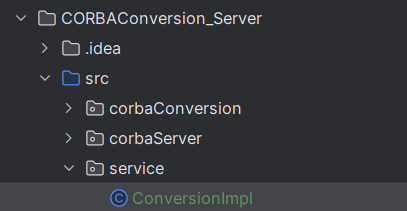
**3.Compilation IDL :**

* + Deux options sont fournies pour compiler le fichier IDL.
  + Par ligne de commande, on peut utiliser la commande **idlj** avec les arguments appropriés.
  + En utilisant Eclipse, une configuration est créée pour exécuter la compilation IDL à l'aide d'une nouvelle configuration externe.



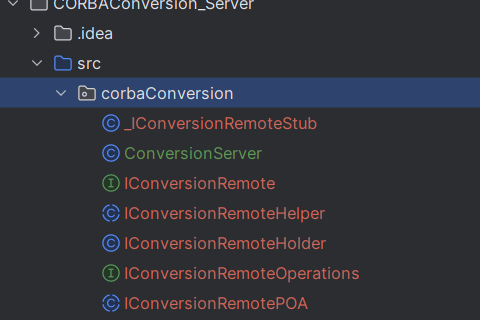
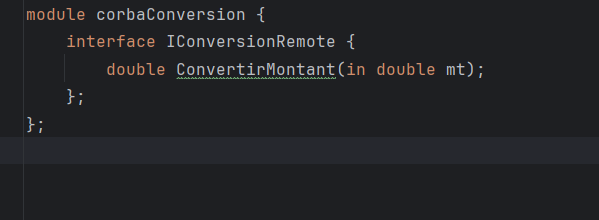
**4.Implémentation du Serviteur (Server-side) :**

* + Un package "service" est créé, contenant une classe "ConversionImpl" implémentant l'interface "IConversionRemotePOA".
  + La méthode "conversionMontant" est implémentée dans cette classe.

**5.Résultat de la Compilation IDL :**

* + La compilation génère plusieurs fichiers Java correspondant à l'interface IDL.
  + Ces fichiers comprennent la souche (STUB), l'interface des méthodes distantes, l'interface Java de l'objet distant, etc.

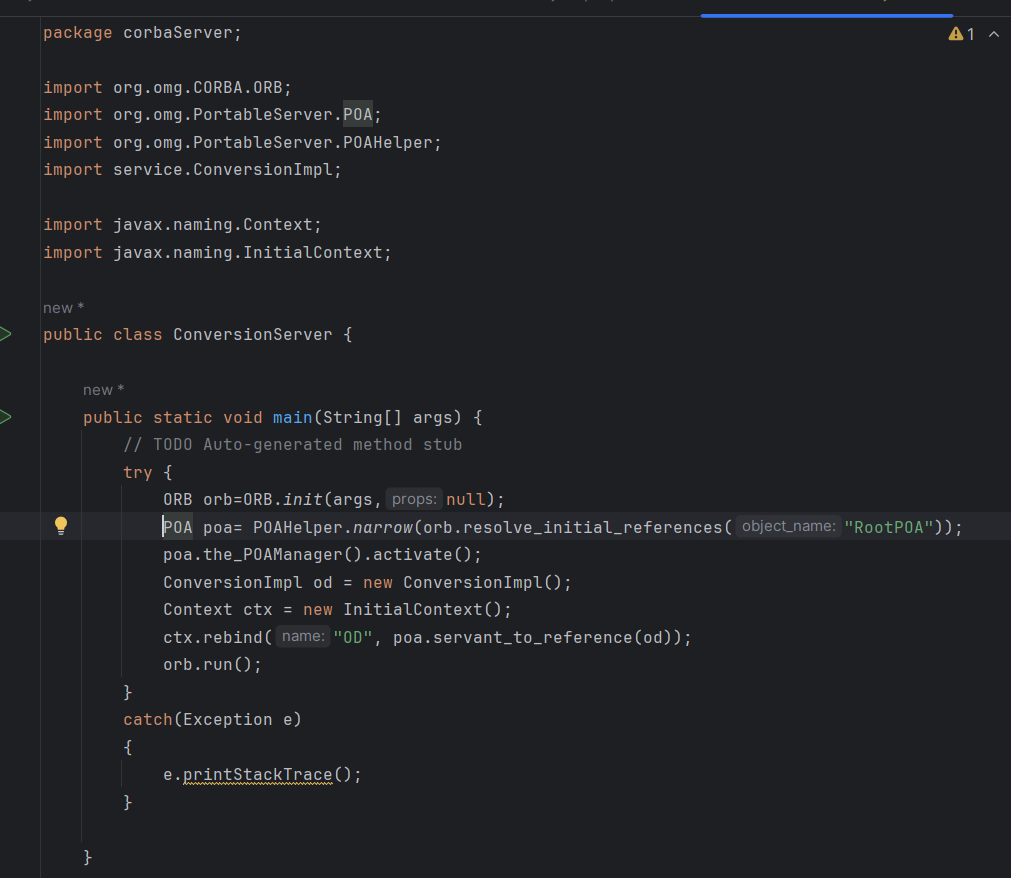


1. **\_IConversionRemoteStub.java (Souche/STUB) :**
   * Ce fichier contient la souche (STUB) de l'objet distant. Le STUB est utilisé du côté client pour effectuer des appels distants comme s'il s'agissait d'appels locaux.
2. **IConversionRemoreOperations.java (Interface des méthodes distantes) :**
   * Ce fichier contient l'interface des méthodes distantes qui sont définies dans l'interface IDL. Il est partagé à la fois par la souche (STUB) et le skeleton.
3. **IConversionRemote.java (Interface Java de l'objet distant) :**
   * Cette interface représente l'objet distant du côté client. Elle hérite de plusieurs interfaces, notamment :
     + **IConversionRemoteOperations**: L'interface des méthodes distantes.
     + **org.omg.CORBA.Object**: Interface de base pour tous les objets CORBA.
     + **org.omg.CORBA.portable.IDLEntity**: Interface pour les entités IDL.
4. **IConversionRemoteHelper.java (Helper pour le transtypage) :**
   * Ce fichier contient une classe utilitaire (**IConversionRemoteHelper**) qui facilite le transtypage des références des objets CORBA vers leurs propres types à travers la méthode **narrow**.
5. **IConversionRemoteHolder.java (Classe utilitaire pour les inouts) :**
   * Cette classe utilitaire (**IConversionRemoteHolder**) est utilisée pour la gestion des inouts, qui sont des paramètres de méthode qui sont à la fois en entrée et en sortie.
6. **IConversionRemotePOA.java (Skeleton du serveur) :**
   * Ce fichier contient le skeleton (squelette) de l'objet distant du côté serveur. Le skeleton est responsable de la réception des appels distants et de la délégation de ces appels à l'implémentation réelle de l'objet distant.
7. **Implémentation de l'objet distant (servant) :**
   * La classe **ConversionImpl** est créée dans le package **service** et elle hérite de **IConversionRemotePOA**, représentant le servant (l'objet distant côté serveur).
   * La méthode **conversionMontant** définie dans l'interface **IConversionRemoteOperations** est implémentée dans cette classe. Cette méthode contiendra la logique de conversion des montants.

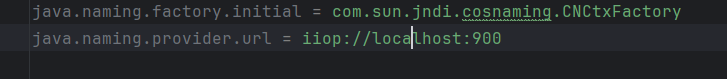
En résumé, cette étape met en place toutes les classes nécessaires du côté serveur pour gérer les appels distants et implémente la logique métier dans la classe **ConversionImpl**. Ces fichiers générés et la structure de l'implémentation sont essentiels pour le bon fonctionnement de l'application répartie CORBA

**Configuration du Serveur :**

* + Un package "corbaServer" est créé, contenant une classe "ConversionServer" avec une méthode main.
  + L'ORB (Object Request Broker) est initialisé.
  + Le POA (Portable Object Adapter) est configuré et activé.
  + Un serviteur de l'objet distribué est créé et enregistré pour être disponible dans l'ORB.
  + L'ORB est mis en attente de requêtes des clients.



* + L'annuaire JNDI est configuré.

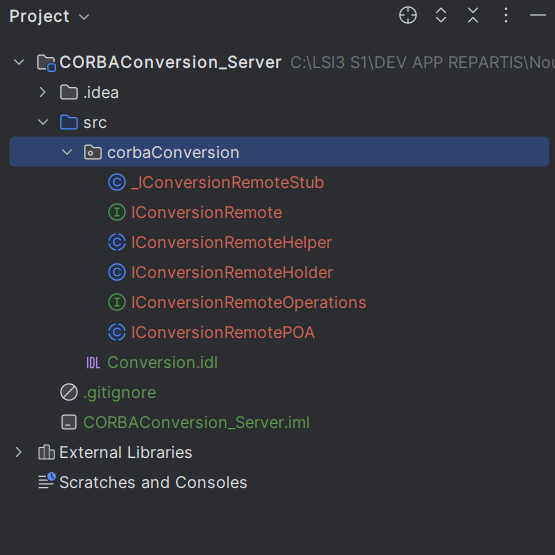


**Description de la 3ème phase : Préparation du Client**

**Création du Projet Client :**

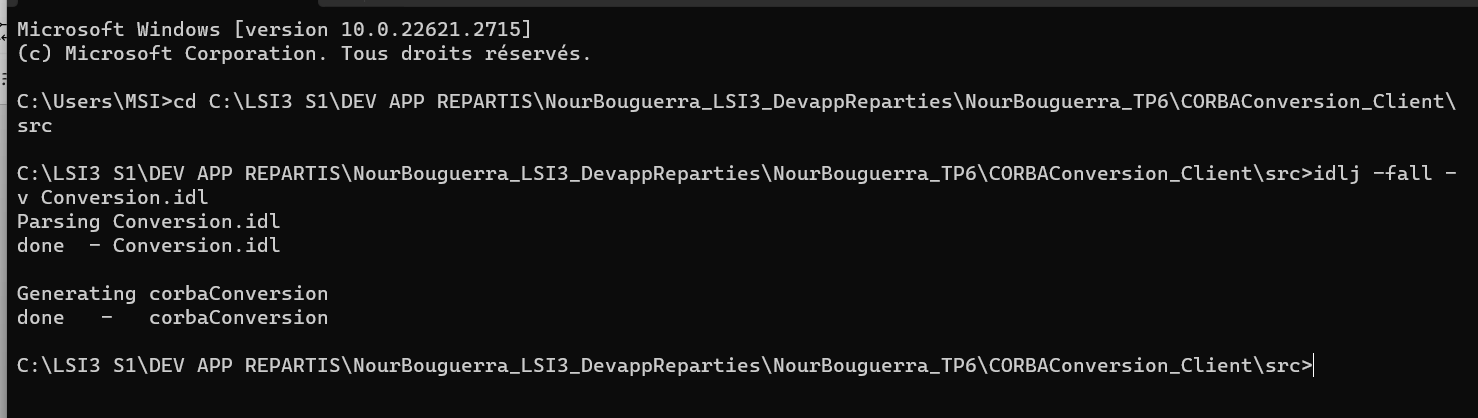
**Copie des Fichiers :**

* On va Copié les fichiers "Conversion.idl" et "jndi.properties" dans le répertoire "CORBAConversion\_Client/src" de notre nouveau projet.



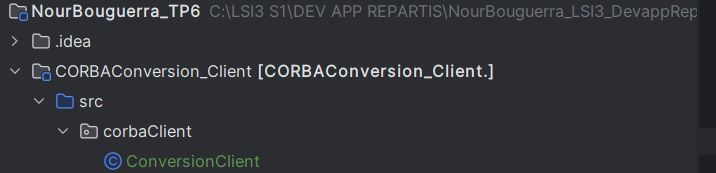
**Compilation du Fichier IDL :**

* Le fichier "Conversion.idl" est compilé à nouveau dans le projet client. La compilation génère les classes Java nécessaires pour le client en utilisant la même procédure que celle décrite dans la 2ème phase.



**Développement du Client :**

* On va crée un package "corbaClient" dans le répertoire src du projet client.
* À l'intérieur de ce package, on va crée une classe appelée "ConversionClient" avec une méthode main (**public static void main()**).





a) **Récupération des Propriétés de l'Annuaire JNDI :**

* Utilisons les propriétés spécifiées dans le fichier "jndi.properties" pour configurer l'accès à l'annuaire JNDI. Ces propriétés peuvent inclure des informations telles que le contexte initial.

b) **Récupération de la Référence de l'Objet :**

* Utilisons l'annuaire JNDI pour récupérer la référence de l'objet distant en utilisant son nom lié.

c) **Conversion de la Référence vers le Type de l'Objet (IConversionRemote) :**

* Utilisons le helper généré (**IConversionRemoteHelper**) pour effectuer le transtypage de la référence obtenue vers le type de l'objet distant (**IConversionRemote**). Cela permet de créer le STUB.

d) **Appel de la Méthode à Distance à travers le STUB :**

* Utilisons le STUB pour appeler la méthode distante (dans ce cas, "conversionMontant") de l'objet distant. Cette étape implique l'invocation d'une méthode à distance comme si elle était une méthode locale.

En résumé, cette phase prépare le client en créant un nouveau projet, copiant les fichiers nécessaires, compilant le fichier IDL, et développant une classe client qui récupère la référence de l'objet distant, effectue le transtypage, et appelle la méthode distante via le STUB. Ces étapes sont essentielles pour établir la communication client-serveur dans une architecture CORBA.

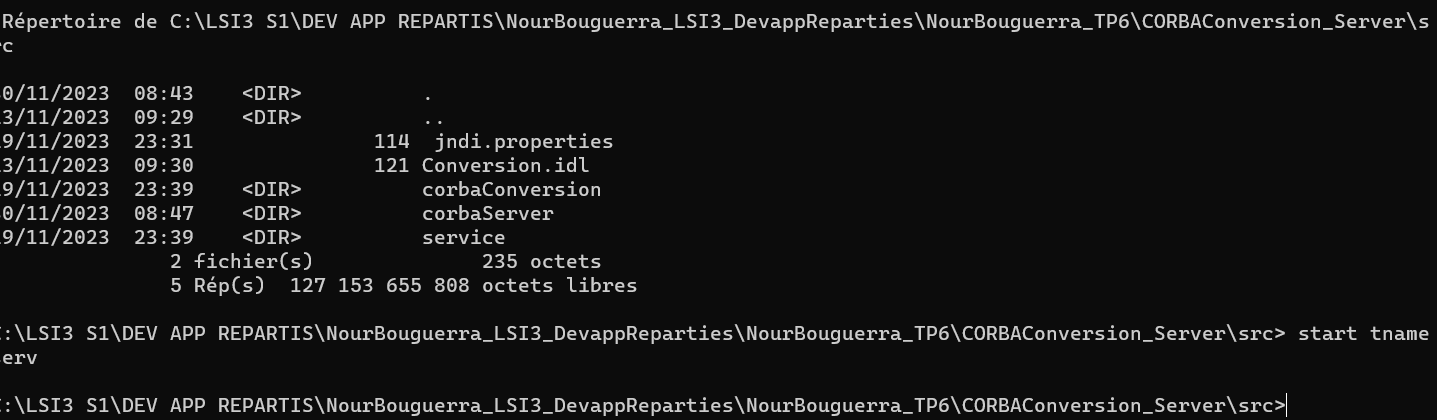
**Description de la 4ème phase** :

Exécution de l'application Pour exécuter l’application, il faut suivre les étapes suivantes :

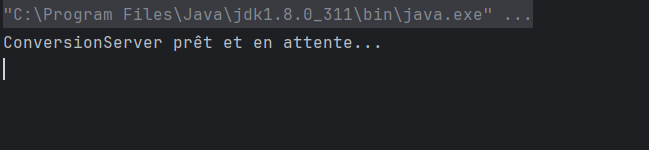
1. Lanceons l’annuaire.

• Par ligne de commande. Pour ce faire :

1. Placez-vous dans le dossier du projet du serveur,
2. ii. Tapez la commande tnameserv ou bien start tnameserv



2. Lancez le serveur.



3. Lancez le client