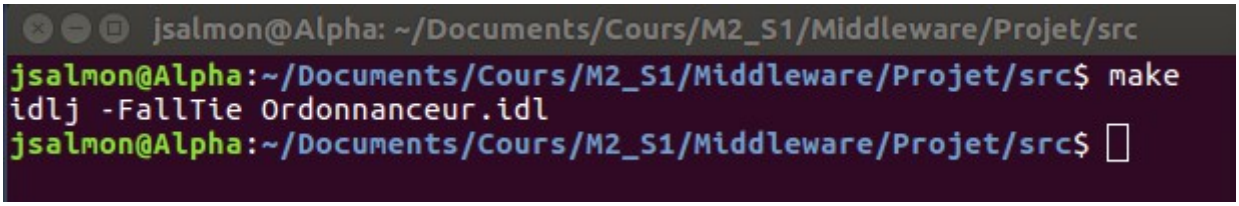
Projet CORBA

I) Utilisation :

Pour faciliter l'utilisation de l'application nous travaillions sur Eclipse, si vous ne voulez pas travailler avec
Pour utiliser l'application il faut lancer plusieurs terminals :

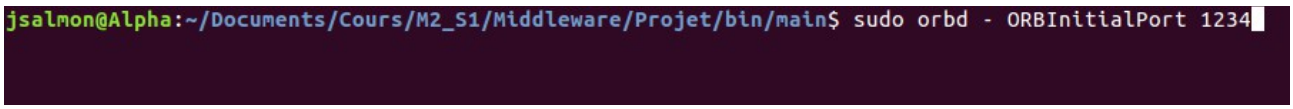
0) Compilation du point idl

Vous n'avez pas besoin mais si vous voulez vous pouvez recompiler le point idl avec la commande : « make » dans le dossier « src »



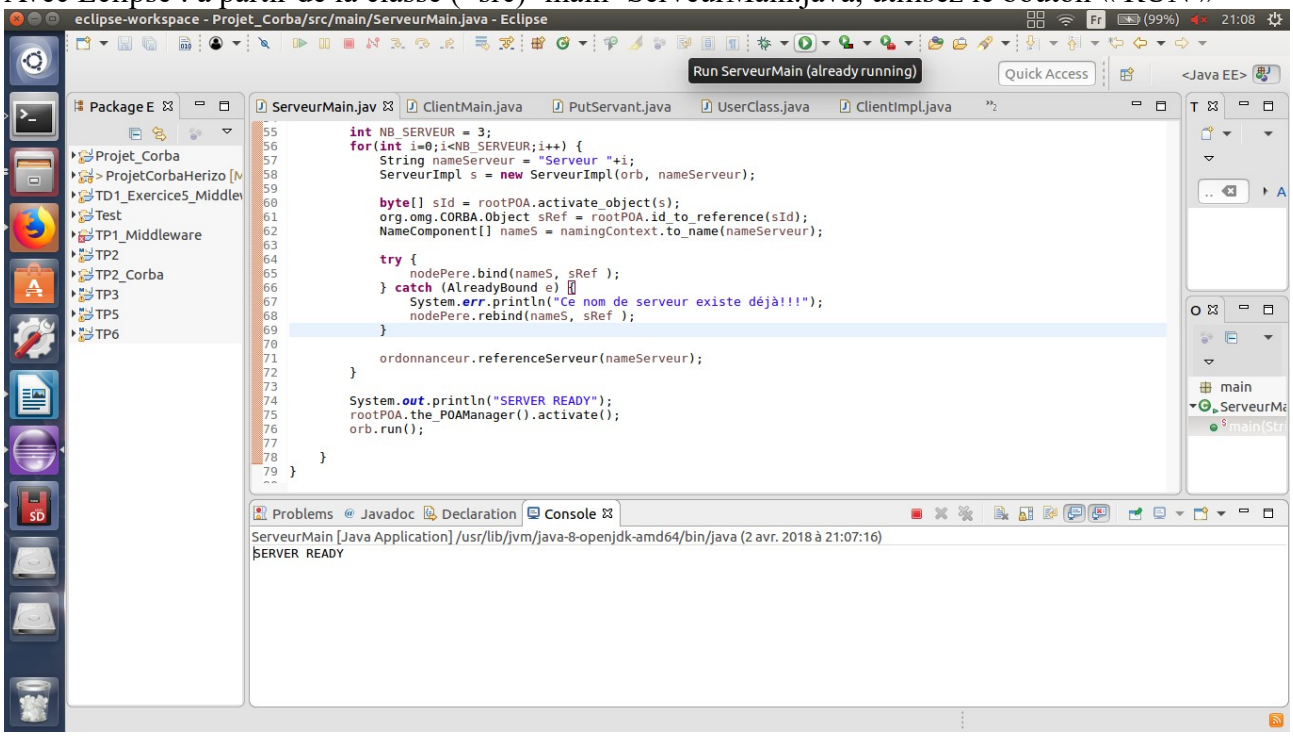
1) Terminal de nommage

Ouvrez un terminal et executez la commande :
sudo orbd - ORBInitialPort 1234



2) Terminal Ordonnanceur et Serveur

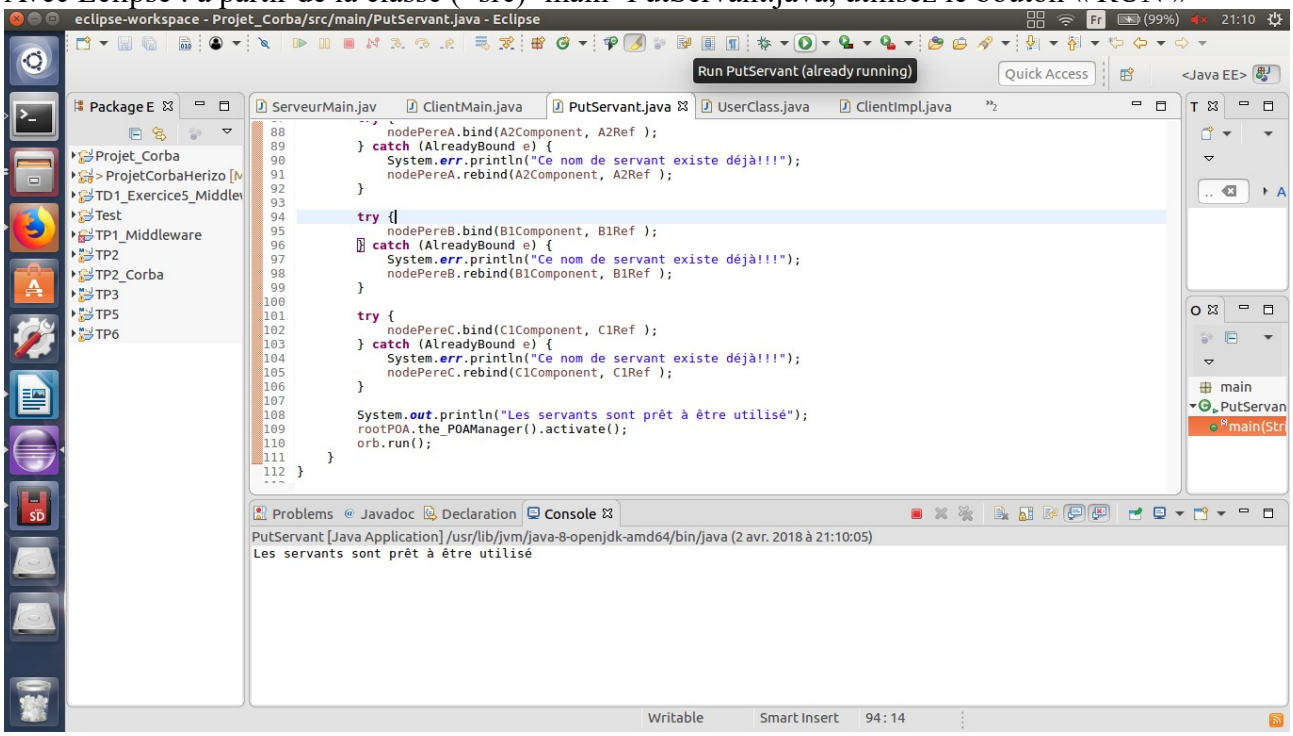
Avec Eclipse : a partir de la classe (>src)>main>ServeurMain.java, utilisez le bouton « RUN »



Cette classe permet de lancer l'ordonnanceur avec les serveurs.

3) Terminal des Servants

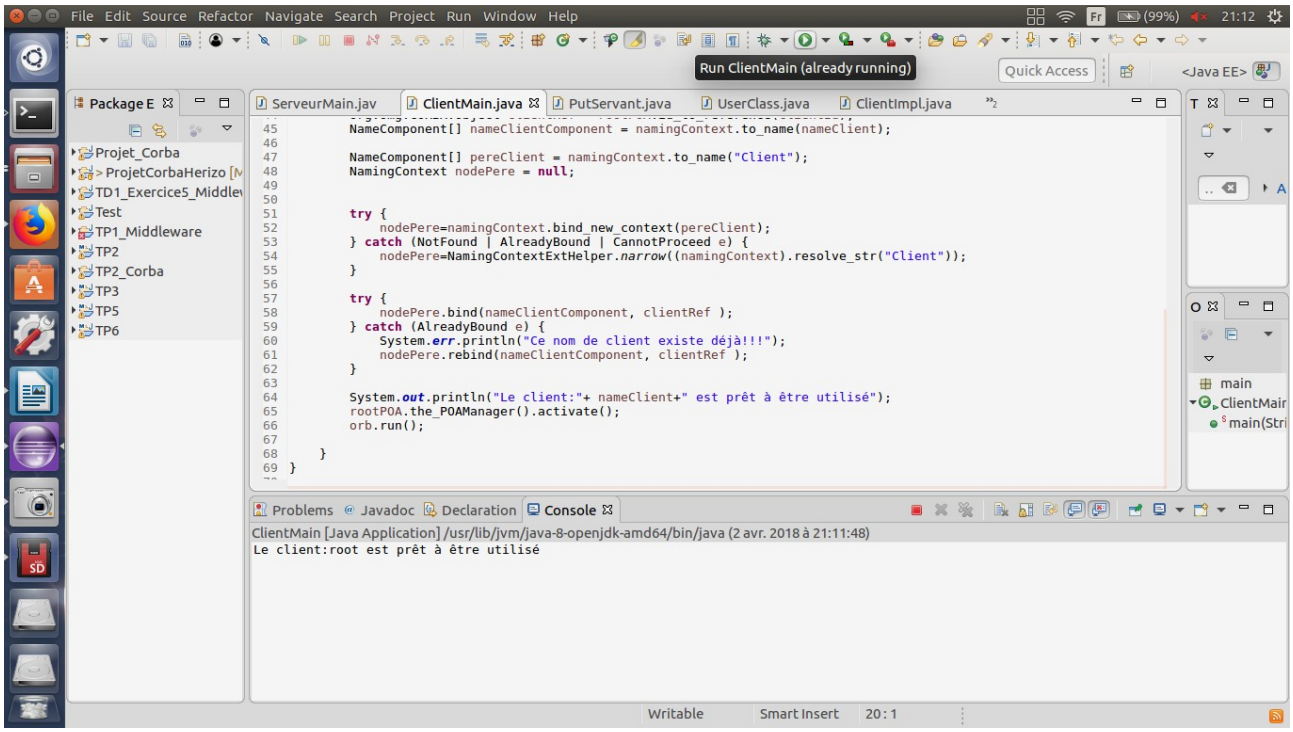
Avec Eclipse : a partir de la classe (>src)>main>PutServant.java, utilisez le bouton « RUN »



Ainsi nous avons les Servants qui sont disponible. (A1 de type A, A2 de type A, B1 de type B et C1 de type C).

4) Terminal du Client

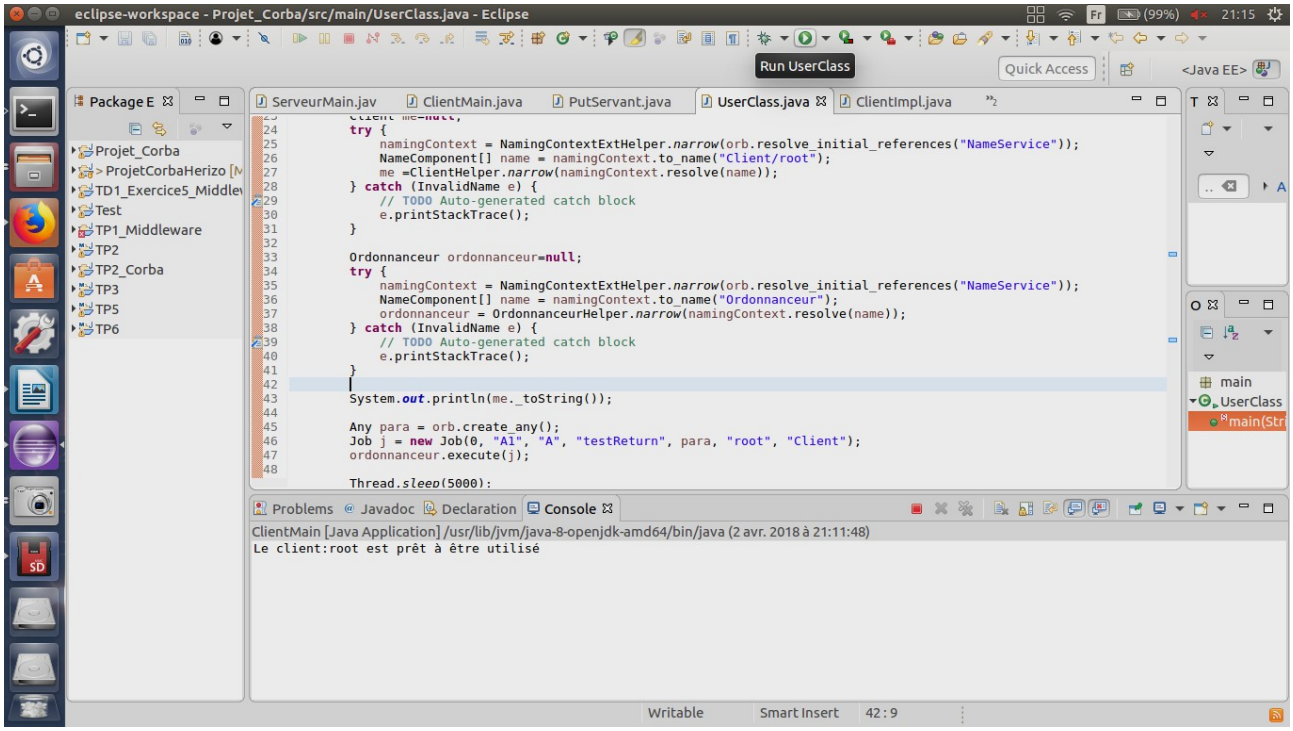
Avec Eclipse : a partir de la classe (>src)>main>ClientMain.java, utilisez le bouton « RUN »



Vous pouvez mettre votre nom d'utilisateur en argument de la commande java. Par défaut votre nom est « root ».
Par la suite toutes les interactions que vous ferez avec l'application vous sera affiché avec ce terminal.

5) Terminal d'execution

Avec Eclipse : a partir de la classe (>src)>main>UserClass.java, utilisez le bouton « RUN »



Grâce à cette classe vous pouvez faire des appelles aux différents Servants. Attention il faut d'abord récupérer l'ordonnanceur et votre compte Client.

```
public static void main(String[] args) throws org.omg.CosNaming.NamingContextPackage.InvalidName, NotFound, Canno
    ORB orb = (ORB) ORB.init(args, null);
    NamingContextExt namingContext=null;
    Client me=null;
    try {
        namingContext = NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
        NameComponent[] name = namingContext.to_name("Client/root");
        me =ClientHelper.narrow(namingContext.resolve(name));
    } catch (InvalidName e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    Ordonnanceur ordonnanceur=null;
    try {
        namingContext = NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
        NameComponent[] name = namingContext.to_name("Ordonnanceur");
        ordonnanceur = OrdonnanceurHelper.narrow(namingContext.resolve(name));
    } catch (InvalidName e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    System.out.println(me._toString());
```

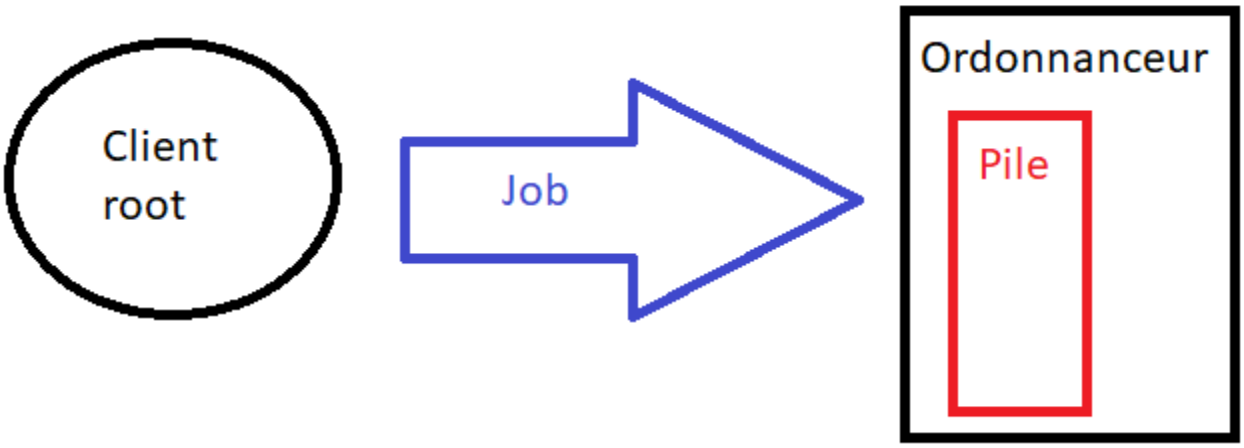
II) Déroulement et Organisation de l'application de l'application

1) Envoi d'un job.

Dans la classe UserClass.java , pour envoyer un Job il faut un id de job (0) le nom de l'objet (« A1 »), le type de l'objet (« A »), le nom de la méthode (« testEasy »), les parametttres dans un ANY, le nom du client (« root ») et le nœud dans le quelle est le client (« Client »).

```
Any para = orb.create_any();
Job j = new Job(0, "A1", "A", "testReturn", para, "root", "Client");
ordonnanceur.execute(j);
```

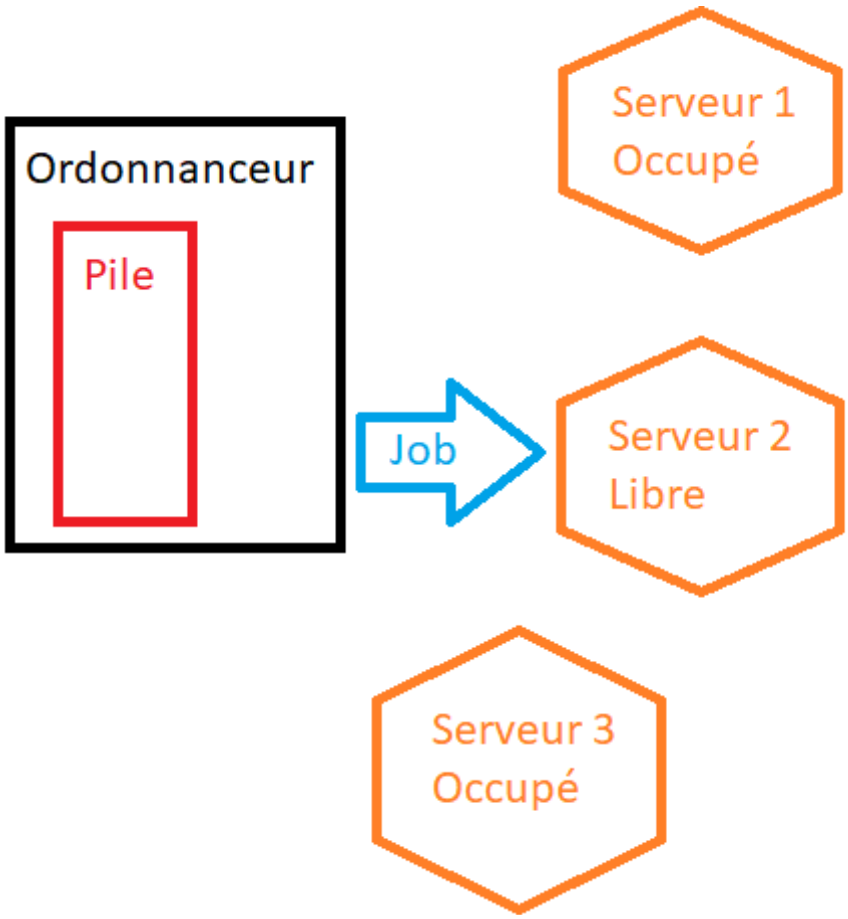
Par la suite, le job est stocké dans une pile de l'ordonnanceur.



2) Exécution du Job.

Le premier Job de la pile est pris pour être exécuter par un serveur. L'ordonnanceur récupère tous les Serveurs et leur demande si ils sont occupés ou pas.

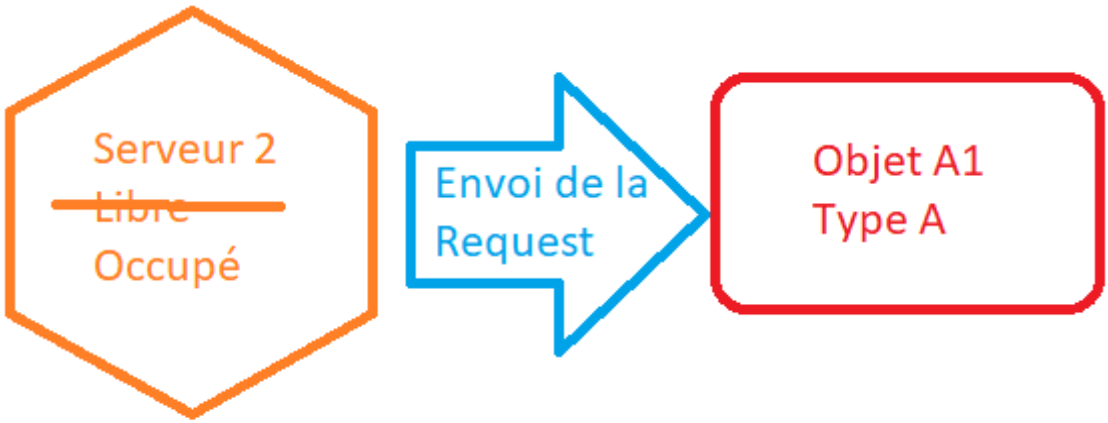
Si le serveur n'est pas occuper le serveur reçois le job.



3) Execution du Job par le serveur.

Le serveur se mets en mode occuper.

Le serveur récupère l'objet pour exécuter le Job.



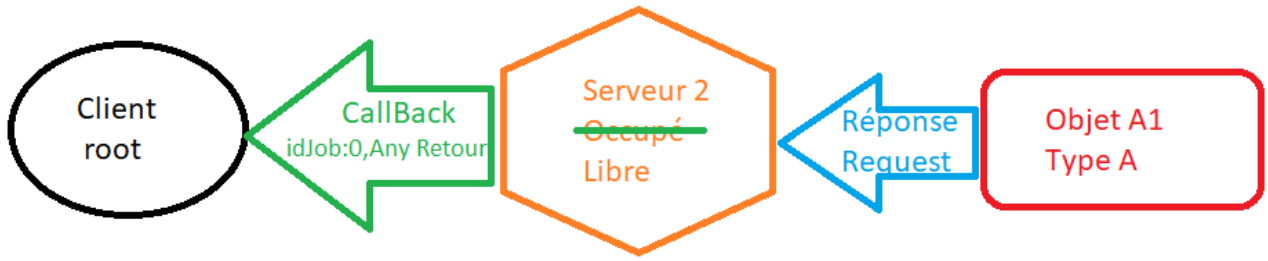
Il l'exécute en mode asynchone.

4) Réponse du Serveur au client

Le Serveur récupérer le retour le l'exécution.

Le serveur envoi la réponse en mode CallBack au Client

Le Serveur se mets en mode disponible, pour exécuter un prochain Job.



5) Lecture de la réponse du job par la Client

Pour lire la valeur de retour le client utilise ca méthode `.findRetour()`
Le client doit donner le numéro de l'id du Job pour afficher le retour.