# Séance 01 : TP - Découverte CORBA

- Quelques ORBs disponibles gratuits :
  - ORB disponible dans le Java Development Kit
  - JacORB
    - https://www.jacorb.org/download.html
  - OpenORB
    - https://sourceforge.net/projects/openorb/files/

counter poa: sckeleton server
stub : cote client

- Le développement d'une application basée sur CORBA commence toujours par la définition de l'interface IDL.

- Exemple d'un compteur simple :
  - Attribut IDL de type « long » (« readonly » car n'est pas destiné à être manipulé directement.)
  - Deux opérations « inc() » et « dec() » invoquées pour incrémenter et décrémenter la valeur courante du compteur.

Counter.idl

```
1  // Counter.idl
2    interface Counter
3    {
4      readonly attribute long value;
5      void inc();
6      void dec();
7  };
```

- Le compilateur IDL utilisé dépends aussi bien du produit que de la plateforme utilisée (hardware, système opératoire) ainsi que le langage de programmation spécifique.

- Le compilateur IDL de JDK :

```
idlj –fall Counter.idl
```

- Le paramètre « -fall » permets de générer les souches (proxy côté client), ainsi que les squelettes (proxy côté serveur)
  - (`–fclient` parameter : Génération côté client uniquement)
  - (`–fserver` parameter : Génération côté serveur uniquement)
  - (`–fallTIE` parameter : Approche par délégation)
  - (`–fserverTIE` : Délégation côté serveur)

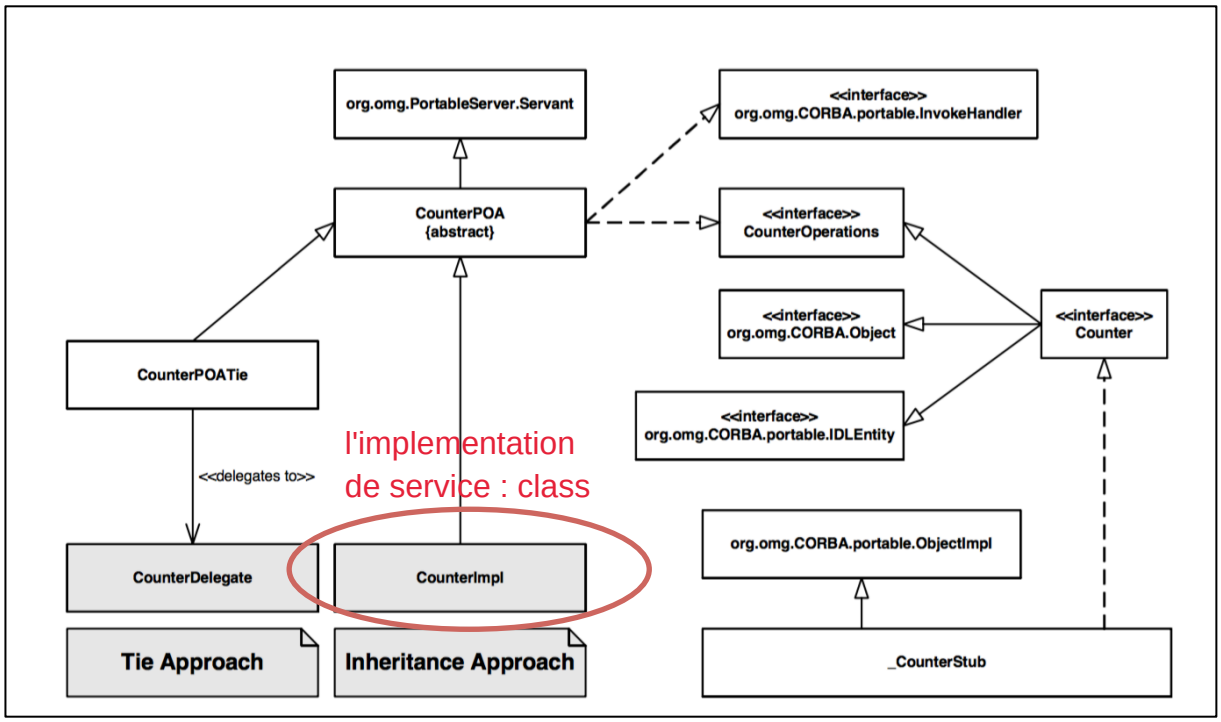
- Autres configurations :
  Altération de la variable d'environnement pour l'invocation de « idlj »

- Fichiers générés (paramètre –fall)
  - `CounterPOA.java`
  - `_CounterStub.java`

- `CounterHolder.java`
- `CounterHelper.java`
- `Counter.java`
- `CounterOperations.java`

Lorsque l'exécution a lieu sur un seul poste, l'exécution du client et du serveur doit se faire sur leurs propres fenêtres de commande avec leurs propres configurations de variables d'environnement.

- Implémentation de « Counter » utilisant l'approche par héritage (JDK) :




L'implémentation fournie par les programmeurs doit hériter de la classe POA.

Notre classe d'implémentation sera nommée « `Counter-Impl` » (Convention CORBA : `<Interface>Impl ou <Interface>_Impl`)

- « CounterPOA » est une classe abstraite qui implémente les opérations de l'interface de l'interface « `CounterOperations` » mais ne comporte aucune déclaration des méthodes héritées. La classe « `CounterImpl` » doit implémenter toutes les méthodes déclarées au niveau de l'interface « Operations »

CounterOperations.java

```
/**
* CounterOperations.java .
* Generated by the IDL-to-Java compiler (portable), version "3.2"
* from Counter.idl
* jeudi 17 mai 2018 05 h 21 WET
*/


// Counter.idl
public interface CounterOperations
{
  int value ();
  void inc ();
  void dec ();
} // interface CounterOperations
```

CounterImpl.java

```
// CounterImpl.java
public class CounterImpl extends CounterPOA {
  private int count;
  public CounterImpl() {
count = 0; }
  public void inc() {
    count++;
  }
  public void dec() {
count--; }
  public int value() {
    return count;
} }
```

- Implémentation du serveur (Approche par héritage)

recupperer
l'adaptater obj

narrow pour
faire la convwersion

```java
// Server.java
import java.io.*;
import java.util.Properties;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import static java.lang.System.*;
public class Server {
  public static void main(String[] args) {
    try {
      Properties props = getProperties();
      ORB orb = ORB.init(args, props);
      org.omg.CORBA.Object obj = null;
      POA rootPOA = null;
      try {
        obj = orb.resolve_initial_references("RootPOA");
        rootPOA = POAHelper.narrow(obj);
      } catch (org.omg.CORBA.ORBPackage.InvalidName e) { }
      CounterImpl c_impl = new CounterImpl();
      Counter c = c_impl._this(orb);
      try {
        FileOutputStream file =
          new FileOutputStream("Counter.ref");
        PrintWriter writer = new PrintWriter(file);
        String ref = orb.object_to_string(c);
        writer.println(ref);
        writer.flush();
        file.close();
        out.println("Server started."
          + " Stop: Ctrl-C");
      } catch (IOException ex) {
        out.println("File error: "
          + ex.getMessage());
exit(2); }
      rootPOA.the_POAManager().activate();
      orb.run();
    } catch(Exception ex) {
      out.println("Exception: " + ex.getMessage());
exit(1); }
} }
```

- Initialisation de l' ORB « ORB.init() »
- Déterminer la référence de rootPOA
  Changement de type (`org.omg.CORBA.Object` → `org.omg.PortableServer.POA` ) à travers la méthode « narrow » de la classe POAHelper
- Création d'une instance « CounterImpl » (Servant), associer avec l'ORB qui contient rootPOA, puis l'activer (« _this() »)
- Transformer IOR construit à travers l'instance de « CounterImpl » à une chaîne de caractère qui sera stockée au niveau du fichier « Counter.ref »
- Activation du Manger

  Orb.run() : Serveur prêt à recevoir les requêtes

- Implémentation du client (Approche par héritage)

```java
// Client.java
import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import static java.lang.System.*;
public class Client {
  public static void main(String[] args) {
    try {
      Properties props = getProperties();
      ORB orb = ORB.init(args, props);
      String ref = null;
      org.omg.CORBA.Object obj = null;
      try {
        Scanner reader =
          new Scanner(new File("Counter.ref"));
        ref = reader.nextLine();
      } catch (IOException ex) {
        out.println("File error: " + ex.getMessage());
exit(2); }
      obj = orb.string_to_object(ref);
      if (obj == null) {
        out.println("Invalid IOR");
exit(4); }
      Counter c = null;
      try {
        c = CounterHelper.narrow(obj);
      } catch (BAD_PARAM ex) {
        out.println("Narrowing failed");
exit(3); }
      int inp = -1;
      do {
        out.print("Counter value: " + c.value()
          + "\nAction (+/-/e)? ");
        out.flush();
        do {
          try {
            inp = in.read();
          } catch (IOException ioe) { }
        } while (inp != '+' && inp != '-' && inp != 'e');
        if (inp == '+')
          c.inc();
        else if (inp == '-')
          c.dec();
      } while (inp != 'e');
    } catch (Exception ex) {
      out.println("Exception: " + ex.getMessage());
      exit(1);
} }
```

Initialisation de l'ORB

Lecture de l'IOR du serveur à partir du fihcier « `Counter.ref` »

Conversion en type « `orb.string_to_object()` »

Reconversion vers le type « Counter » à travers la méthode « narrow » de « CounterHelper » )

Invocation des méthodes à travers des entrées '+' ou '-'

- Execution :

  L'objet IOR du serveur est enregistré au niveau du fichier « `Counter.ref`» , Celui-ci contient toutes les informations qu'un client peut nécessiter à distance afin de localiser l'objet serveur et invoquer les opérations : (Adresse IP, Numéro du port du poste serveur, Référence de l'objet etc.. )

  Le fichier «`Counter.ref`» doit être récupéré au niveau du poste client.

  On verra que le service de nommage fourni par CORBA offre une meilleure alternative.