

SISTEMAS DISTRIBUIDOS

PROGRAMA DE INGENIERIA DE SISTEMAS

ING. DANIEL EDUARDO PAZ PERAFÁN

Aplicaciones CORBA

Invocación estática

Pasos a seguir:

1. Definir los objetos del servidor mediante IDL (operaciones, struct, excepciones, entre otras)
2. Precompilar los IDL para obtener el stub, esqueletos del servidor, interfaces y soportes
3. Añadir la implementación de los objetos
 - Registrar la referencia al servant
 - Registrar los objetos en el IR
 - Registrar los objetos en el Rep. de Implementaciones
4. Obtener la referencia al objeto CORBA desde el cliente
5. Compilar el código del servidor y cliente

CORBA

Invocación estática

1. Declarar la interfaz

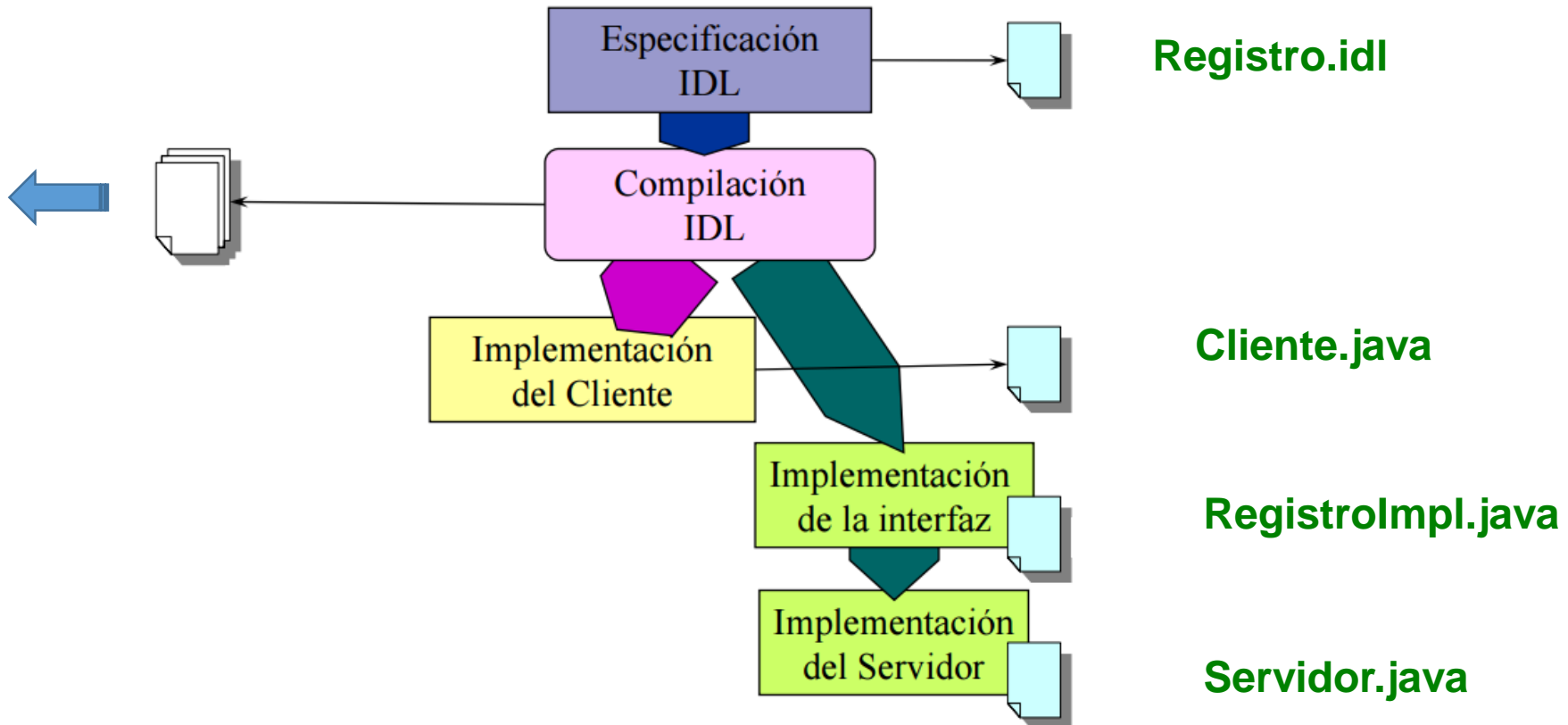
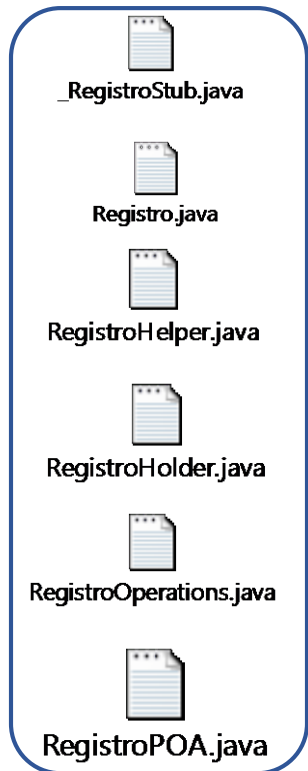
```
module SegundoEjemplo {  
  
    interface Registro {  
  
        attribute long cantidadMaximaDepositos;  
  
        boolean registrarDeposito(in string identificacion, in float cantidad);  
        float cantidadCuenta(in string identificacion);  
        long cantidadDepositosRegistrados();  
    };  
};
```

CORBA

Invocación estática

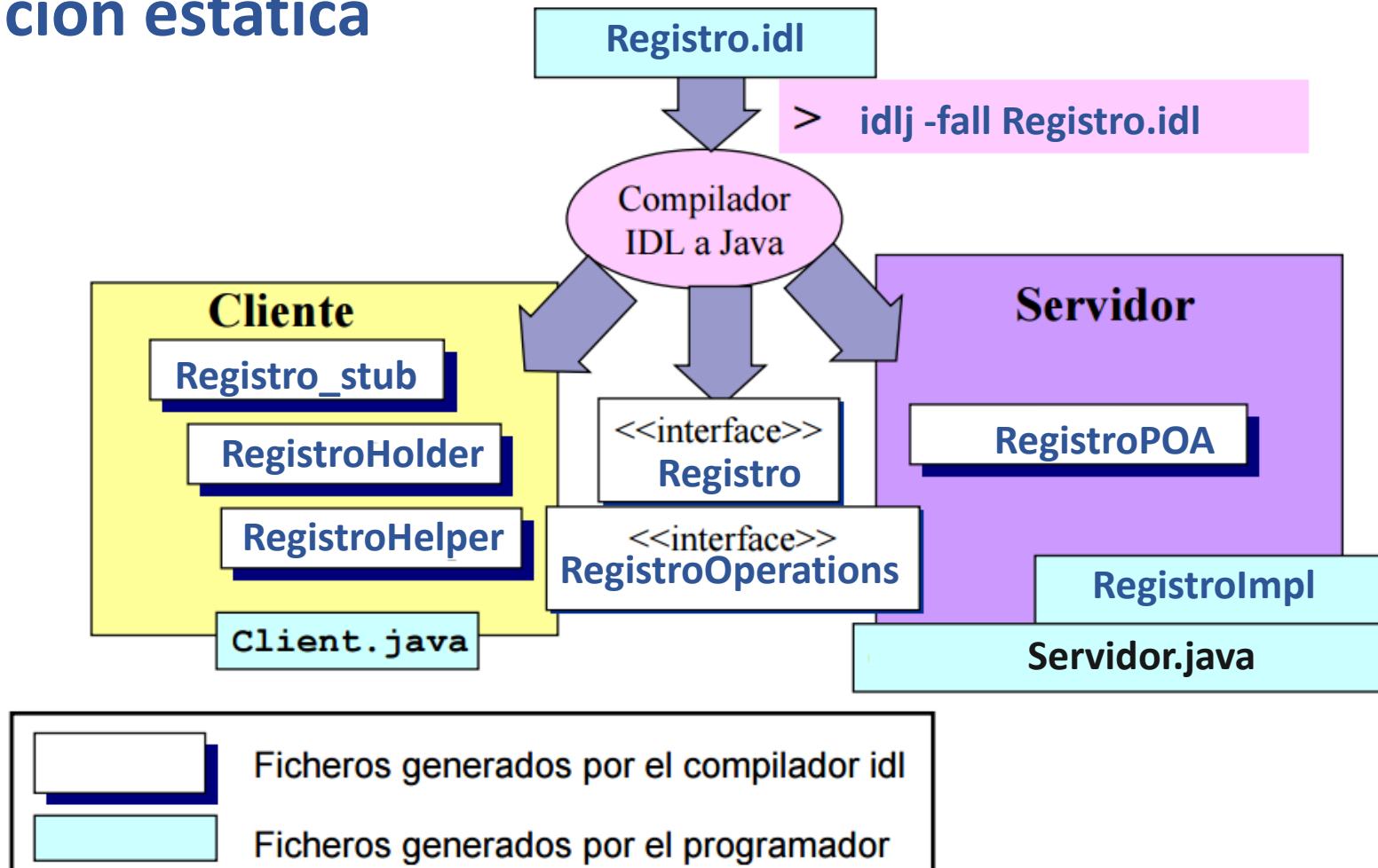
Invocación estática

Package sop_corba



CASO DE ESTUDIO ESPECIFICACIÓN DE CORBA EN JAVA

Invocación estática



Código generado por el compilador IDL

Registro.java y RegistroOperations.java

- ❖ **RegistroOperations:** Interfaz Java que define todos los métodos correspondientes a la interfaz idl. Es la que implementa el servant.

```
public interface RegistroOperations{  
    int cantidadMaximaDepositos ();  
    void cantidadMaximaDepositos (int newCantidadMaximaDepositos);  
    boolean registrarDeposito (String identificacion, float cantidad);  
    float cantidadCuenta (String identificacion);  
    int cantidadDepositosRegistrados ();  
}
```

- Cada atributo se corresponde con dos métodos: uno para leer y actualizar
- Cada operación se corresponde con un método Java

- ❖ **Registro:** Interfaz java que extiende a la anterior. Es la que implementa el stub.

```
public interface Registro extends RegistroOperations,  
                                org.omg.CORBA.Object,  
                                org.omg.CORBA.portable.IDLEntity  
{ ... }
```

Código generado por el compilador IDL

_RegistroStub.java

- ❖ Clase java que implementa el stub de la interfaz Registro en el lado del cliente
 - Realiza marshalling (serialización) para los parámetros de cada método de la interfaz antes de pasárselos al ORB. (También unmarshalling en la recepción).

```
public class _RegistroStub extends org.omg.CORBA.portable.ObjectImpl implements
                                                                    SegundoEjemplo.Registro {

    public int cantidadMaximaDepositos (){...}
    public void cantidadMaximaDepositos (int newCantidadMaximaDepositos){...}
    public boolean registrarDeposito (String identificacion, float cantidad){...}
    public float cantidadCuenta (String identificacion){...}
    public int cantidadDepositosRegistrados (){...}

    ...
}
```

Código generado por el compilador IDL

RegistroHelper.java

- ❖ La clase Helper tiene métodos estáticos que pueden ser útiles para el programador del cliente y el servidor.
- ❖ El método narrow() es estándar y sirve para convertir un objeto genérico de tipo CORBA a uno del tipo Registro.

```
abstract public class RegistroHelper {  
    public static SegundoEjemplo.Registro narrow (org.omg.CORBA.Object obj)  
    {  
        if (obj == null)  
            return null;  
        else if (obj instanceof SegundoEjemplo.Registro)  
            return (SegundoEjemplo.Registro)obj;  
        ...  
    }  
}
```


Código generado por el compilador IDL

RegistroHolder.java

- ❖ Clase Java que se utiliza si es necesario pasar objetos Registro como parámetros out o inout en operaciones de otra interfaz

```
public final class RegistroHolder implements org.omg.CORBA.portable.Streamable{

    public SegundoEjemplo.Registro value = null;

    public void _read (org.omg.CORBA.portable.InputStream i) {...}

    public void _write (org.omg.CORBA.portable.OutputStream o) {...}

    public org.omg.CORBA.TypeCode _type () {...}

}
```

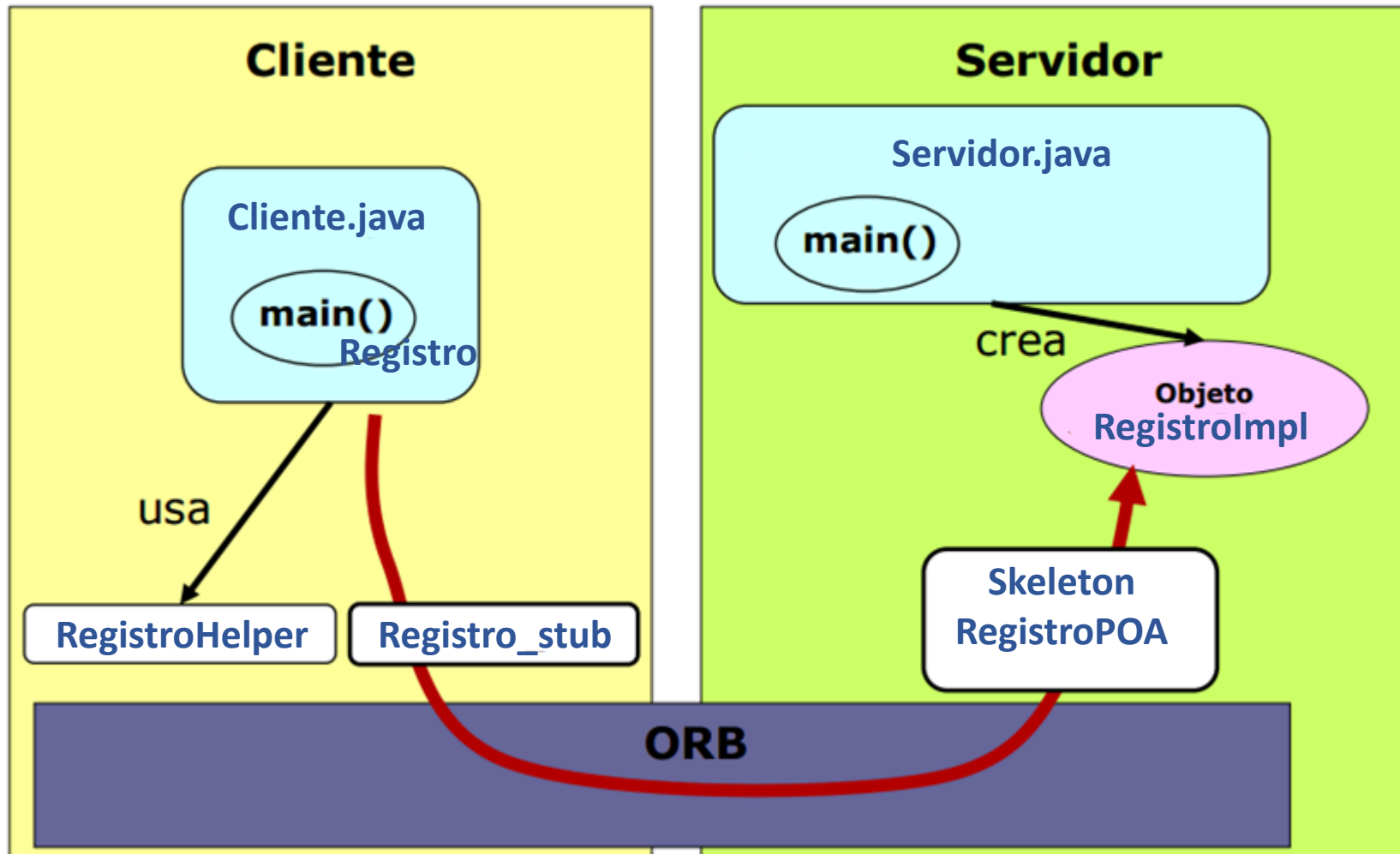
Código generado por el compilador IDL

RegistroPOA.java

- ❖ Clase abstracta que sirve de base para la implementación del servant (código del servidor que implementa las operaciones definidas en la interfaz idl)
- ❖ Esta clase es la que contiene el skeleton del servidor

```
public abstract class RegistroPOA extends org.omg.PortableServer.Servant
    implements SegundoEjemplo.RegistroOperations,
    org.omg.CORBA.portable.InvokeHandler
{
    ...
}
```

Arquitectura de la aplicación



Programación del servidor

Tareas a realizar

1. Programación de servants.

- ❖ La implementación de una o varias interfaces IDL
 - Se debe crear una clase que implementa las operaciones definidas en la interfaz.
 - Hay dos maneras de implementar una interfaz:
 - Por herencia de la clase `interfacePOA`
 - Por delegación, usando una clase `_tie_`

2. Creación del programa principal (Server)

- ❖ Contiene el método `main` que se encarga de:
 - Inicializar el ORB y POA
 - Crear objetos servant que implementan las interfaces
 - Pasar el control al ORB

Programación del servidor

Modelos de implementación servants

Herencia

Se hereda de la clase interfacePOA → **RegistroPOA.java**

Ventajas

- El adaptador de objetos (clase interfacePOA) ya nos da una implementación de base
- El adaptador de objetos (clase interfacePOA) hace las funciones de skeleton

Desventaja

- Este tipo de implementación restringe la herencia con esta clase y no con otra.
- Los métodos de la interfaz se definen en la clase de implementación

*** Al principio se reconocía como enfoque BOA**

Programación del servidor

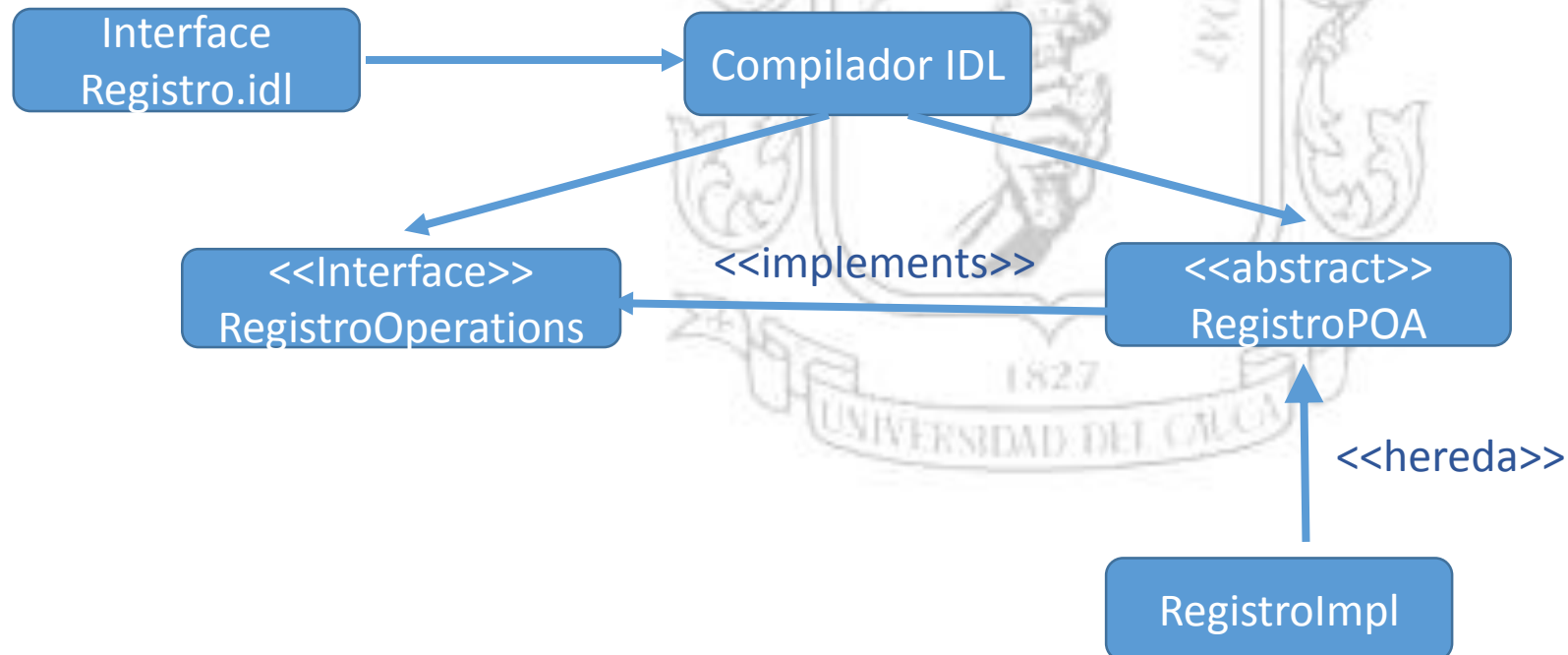
Modelos de implementación servants

Herencia

Se hereda de la clase interface POA



RegistroPOA.java



Programación del servidor

Modelos de implementación servants

Delegación

- Se implementa la interfaz Operations (que sería de una clase Java normal que implementa las operaciones de la interfaz)
- Se debe generar una clase TIE la cual hace las funciones de skeleton y delega las peticiones a nuestra implementación
- Objetos de la clase TIE se encargan de realizar las tareas del skeleton y delegan la realización de la operación al objeto de implementación).
- La implementación no hereda de la clase interfacePOA. Esto le da flexibilidad
- Tiene la ventaja que la implementación puede ser una clase pre-existente.

Por delegación (enfoque TIE)

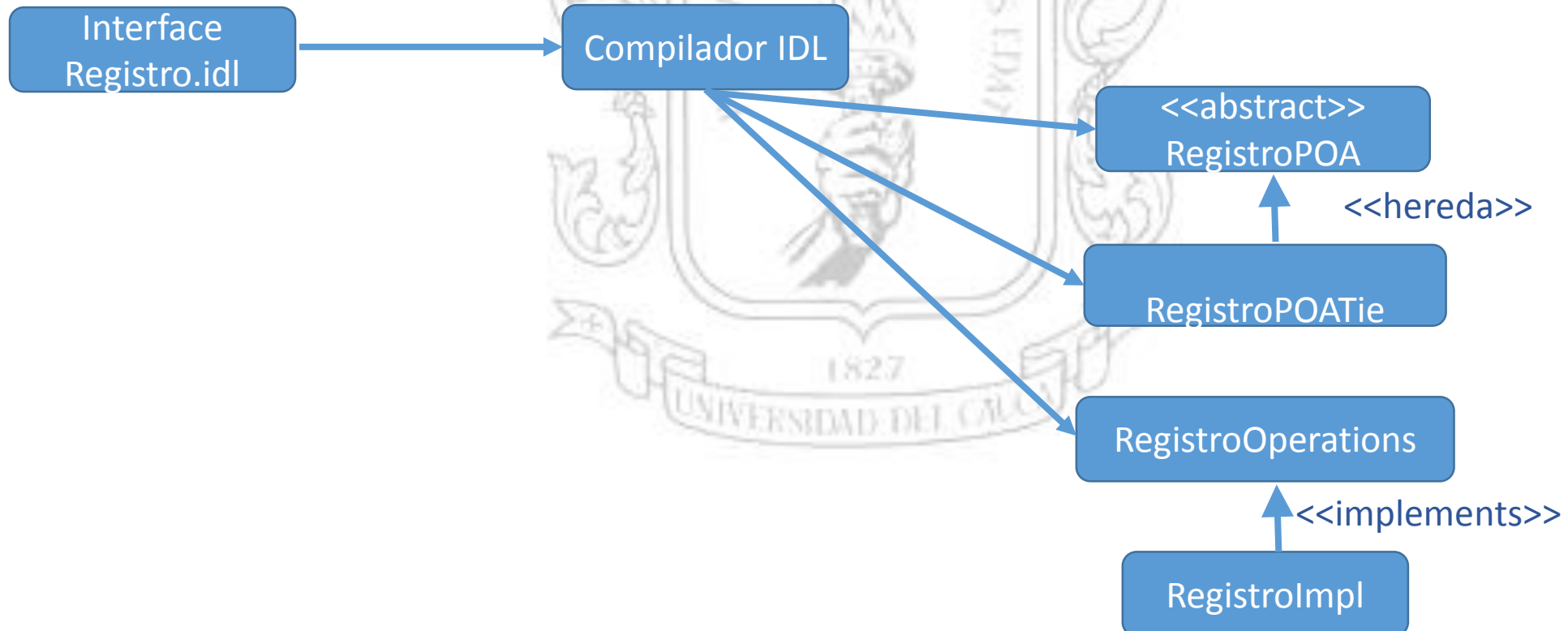
Programación del servidor

Modelos de implementación servants

Delegación

Se hereda de la clase interfacePOA

➔ RegistroPOA.java



Programación del servidor Modelos de implementación servants

Como se enlaza el objeto CORBA al objeto real

Herencia

`Idlj –fall nombreInterface.idl`

Genera un conjunto de soportes para implementar un objeto CORBA por herencia

Delegación

`Idlj –fallTIE nombreInterface.idl`

Genera un conjunto de soportes para implementar un objeto CORBA por delegación

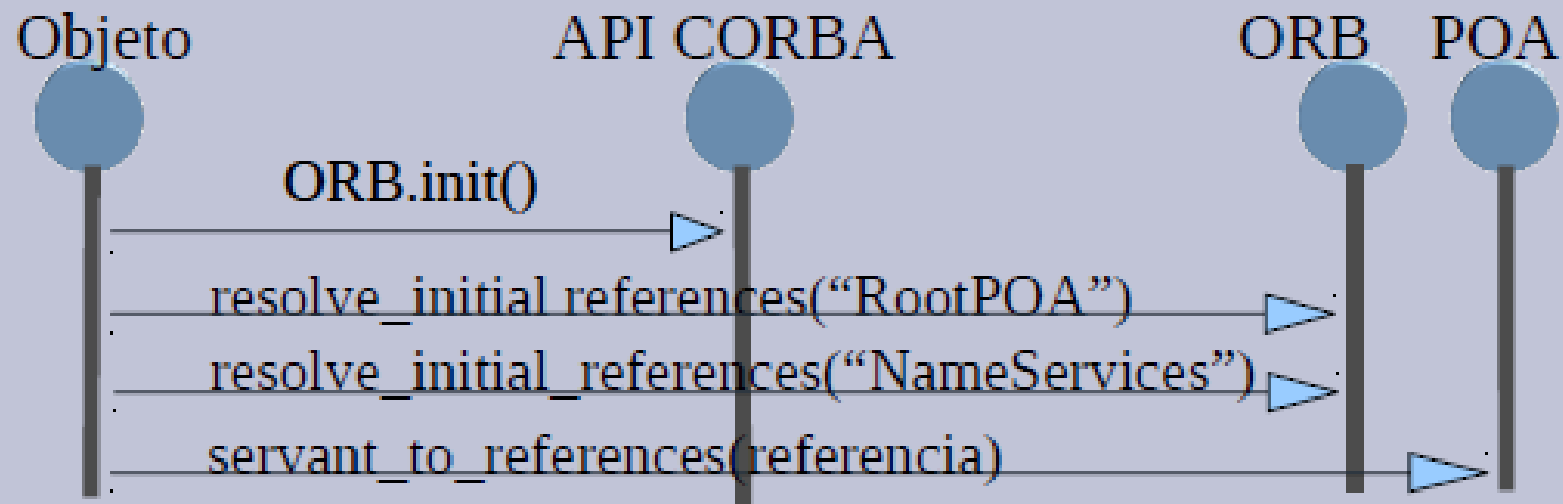
Programación del servidor

Conectarse al ORB

- ❖ Para conectarse al ORB es necesario invocar al método estático `org.omg.CORBA.orb.init()` que devuelve un objeto orb
- ❖ El método `init()` sin parámetros devuelve un orb único, también se puede invocar al método con parámetros para funcionalidad adicional:
- ❖ Sobre este objeto se pueden invocar el resto de los métodos que definen la funcionalidad del orb

Programación del servidor

- Como un componente encuentra un ORB?



- * Obtener una referencia del ORB
- * Obtener un puntero al adaptador de objetos
- * Descubrir los servicios disponibles (IR, servicio de nombres ...)
- * Obtener una referencia del objeto de servicio que se quiere

Ejemplo servidor CORBA por herencia

Desarrollo del servidor (servant por herencia)

```
public class RegistroImpl extends SegundoEjemplo.RegistroPOA{  
  
    private final List<Deposito> listaDepositos;  
    int cantidadMaximaDepositos;  
  
    public RegistroImpl()  
    {  
        System.out.println("Creando un objeto de tipo RegistroImpl");  
        cantidadMaximaDepositos=0;  
        listaDepositos=new ArrayList<>();  
    }  
    @Override  
    public int cantidadMaximaDepositos() {  
        return cantidadMaximaDepositos;  
    }  
  
    @Override  
    public void cantidadMaximaDepositos(int newCantidadMaximaDepositos) {  
        this.cantidadMaximaDepositos=newCantidadMaximaDepositos;  
    }  
}
```

```
@Override  
public boolean registrarDeposito(String identificacion, float cantidad) {  
    boolean bandera=false;  
    Deposito objNuevoDeposito= new Deposito();  
    bandera=listaDepositos.add(objNuevoDeposito);  
    return bandera;  
}  
  
@Override  
public float cantidadCuenta(String identificacion) {  
    float cantidad=0;  
    for(Deposito objDeposito: listaDepositos)  
    {  
        if (objDeposito.getIdentificacion().compareTo(identificacion)==0)  
            cantidad+=objDeposito.getCantidad();  
    }  
    return cantidad;  
}  
  
@Override  
public int cantidadDepositosRegistrados() {  
    return listaDepositos.size();  
}
```

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import org.omg.PortableServer.*;  
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;  
public static void main(String args[])  
{
```

```
try{
```

```
    System.out.println("1. Crea e inicia el orb");  
    ORB orb = ORB.init(args, null);
```

```
    System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");  
    org.omg.CORBA.Object objPoa = null;  
    objPoa=orb.resolve_initial_references("RootPOA");  
    POA rootPOA = POAHelper.narrow(objPoa);
```

```
    System.out.println("3. Activa el POAManager");  
    rootPOA.the_POAManager().activate();
```

```
    ...
```

1. Crear e inicializar el ORB

La clase ORB hace parte del paquete org.omg.CORBA

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import org.omg.PortableServer.*;  
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;  
public static void main(String args[])  
{
```

```
    try{  
        System.out.println("1. Crea e inicia el orb");  
        ORB orb = ORB.init(args, null);
```

2. Obtener la referencia al POA raíz.
Para encontrar el POA raíz se utiliza el nombre estándar
"RootPOA"

```
        System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");  
        org.omg.CORBA.Object objPoa = null;  
        objPoa=orb.resolve initial references("RootPOA");
```

```
        POA rootPOA = POAHelper.narrow(objPoa);
```

```
        System.out.println("3. Activa el POAManager");  
        rootPOA.the_POAManager().activate();
```

2.1 Convertir la referencia de un objeto genérico a una
referencia al rootPOA

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import org.omg.PortableServer.*;  
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;  
public static void main(String args[])  
{  
    try{  
        System.out.println("1. Crea e inicia el orb");  
        ORB orb = ORB.init(args, null);  
  
        System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");  
        org.omg.CORBA.Object objPoa = null;  
        objPoa=orb.resolve_initial_references("RootPOA");  
        POA rootPOA = POAHelper.narrow(objPoa);
```

```
        System.out.println("3. Activa el POAManager");  
        rootPOA.the_POAManager().activate();
```

3. Activar el gestor de invocaciones del POA, lo que permite recibir invocaciones a los objetos asociados.

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```



4. Crear el objeto servant

```
System.out.println("5. Obtiene la referencia al objeto servant ");  
org.omg.CORBA.Object obj =  
rootPOA.servant_to_reference(ObjServant);  
System.out.println("6. Convierte la referencia de un objeto generico a una referencia al servant ");  
Registro href = RegistroHelper.narrow(obj);  
  
System.out.println("7. Obtiene una referencia al servicio de nombrado por medio del orb");  
org.omg.CORBA.Object objRefNameService =  
    orb.resolve_initial_references("NameService");
```


Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```

```
System.out.println("5. Obtiene la referencia al objeto servant ");  
org.omg.CORBA.Object obj =  
rootPOA.servant_to_reference(ObjServant);
```

5. Crear la referencia al objeto CORBA por medio del objeto rootPOA

```
System.out.println("6. Convierte la referencia de un objeto generico a una referencia al servant ");
```

```
Registro href = RegistroHelper.narrow(obj);
```

6. Convertir la referencia almacenada en obj, a una referencia al servant

```
System.out.println("7. Obtiene una referencia al serv");  
org.omg.CORBA.Object objRefNameService =  
orb.resolve_initial_references("NameService");
```

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```

```
System.out.println("5. Obtiene la referencia al objeto");  
org.omg.CORBA.Object obj =  
rootPOA.servant_to_reference(ObjServant);  
System.out.println("6. Convierte la referencia de objeto a href");  
Registro href = RegistroHelper.narrow(obj);
```

```
System.out.println("7. Obtiene una referencia al servicio de nombrado por medio del orb");  
org.omg.CORBA.Object objRefNameService =  
    orb.resolve_initial_references("NameService");
```

7. Obtener una referencia al servicio de nombrado por medio del orb.

Para encontrar el servicio de nombrado se utiliza el nombre estándar "NameService"

");

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

```
System.out.println("9. Construir un contexto de nombres que identifica al servant");  
String identificadorServant = "identificadorServant";
```

```
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

```
System.out.println("10. Realiza el binding de la referencia de objeto en el N_S");  
refContextoNombrado.rebind(path, href);
```

```
orb.run();
```

8. Convertir la referencia genérica a una referencia correspondiente a NamingContextExt

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

```
System.out.println("9. Construir un contexto de nombres que ident  
String identificadorServant = "identificadorServant";  
  
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

```
System.out.println("10. Realiza el binding de la referencia de o  
refContextoNombrado.rebind(path, href);
```

```
orb.run();
```

9. Construir el contexto de nombres que identificara al servant

Cada componente del contexto se identifica por el par **identificador, tipo**

El cliente debe obtener una referencia al objeto remoto utilizando el mismo **contexto de nombres**

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

```
System.out.println("9.Construir un contexto de nombres que identifica al servant");  
String identificadorServant = "identificadorServant";
```

```
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

10. Realizar el binding de la referencia del objeto y el identificador en el N_S



```
System.out.println("10.Realiza el binding de la referencia de objeto en el N_S");  
refContextoNombrado.rebind(path, href);
```

```
orb.run();
```

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);

System.out.println("9.Construir un contexto de nombres que identifica al servant");
String identificadorServant = "identificadorServant";

NameComponent [] path = new NameComponent[1];
path[0] = new NameComponent();
path[0].id = identificadorServant;
path[0].kind = "tipoServicio";

System.out.println("10.Realiza el binding de la referencia de objeto en el N_S");
refContextoNombrado.rebind(path, href);
```

`orb.run();`



11. Ceder control al ORB para que escuche peticiones orb.run();

Programación del cliente

Tareas a realizar

❖ Modelos de implementación

- Clase java con método main
- Applet de java con método init

❖ Tareas a realizar

1. Conectar el cliente al orb (Inicializar ORB)
2. Obtener referencia a un objeto CORBA que implemente interfaz Registro usando el servicio de nombres (solución estándar)
3. Usar el objeto CORBA como si fuese local

Programación del cliente

Obtención de refs. a objetos corba

- El ORB es capaz de convertir de un string a una ref a un objeto corba y viceversa.
- La ref a un objeto corba encapsula
 - Dirección de red del proceso servidor
 - Un identificador único (puesto por el servidor) que identifica la implementación concreta a la que va dirigida la petición
- Al obtenerse una ref a un objeto corba en realidad obtenemos una ref a un objeto java que implementa en el cliente el representante (stub) del proceso servidor
- Cuando el cliente obtiene ref a objeto corba
 - El ORB instancia un proxy (stub) en el lenguaje apropiado en el espacio del cliente. El cliente no puede instanciar estas referencias lo hace siempre el ORB
 - Una vez creado el stub, el cliente realiza operaciones sobre él.
 - El stub hace marshalling de las peticiones y se las pasa al ORB
 - El ORB localiza al servidor y establece las conexiones de transporte necesarias de forma transparente para el cliente.
 - Con la respuesta del servidor se realiza el proceso inverso

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import SegundoEjemplo.*;
```

Paquetes a utilizar propios de la especificación de CORBA

```
public static void main(String args[])  
{  
    try  
    {
```

```
        System.out.println("1. Crea e inicia el ORB");  
        ORB orb = ORB.init(args, null);
```

1. Crear e inicializar el ORB
La clase ORB hace parte del paquete org.omg.CORBA

```
        System.out.println("2. Obtiene una referencia al servicio de nombrado por medio del orb");  
        org.omg.CORBA.Object objRefNameService = orb.resolve_initial_references("NameService");
```

```
        System.out.println("3. Convierte la ref genérica a ref de NamingContextExt");  
        NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import SegundoEjemplo.*;
```

Paquetes a utilizar propios de la especificación de CORBA

```
public static void main(String args[])  
{
```

```
    try  
    {
```

```
        System.out.println("1. Crea e inicia el ORB");  
        ORB orb = ORB.init(args, null);
```

```
        System.out.println("2. Obtiene una referencia al servicio de nombrado por medio del orb");  
        org.omg.CORBA.Object objRefNameService = orb.resolve_initial_references("NameService");
```

```
        System.out.println("3. Convierte la ref genérica a ref de NamingContextExt");  
        NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

2. Obtener una referencia al servicio de nombrado por medio del orb.
Para encontrar el servicio de nombrado se utiliza el nombre estándar "NameService"

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import SegundoEjemplo.*;
```



Paquetes a utilizar propios de la especificación de CORBA

```
public static void main(String args[])  
{
```

```
    try  
    {
```

```
        System.out.println("1. Crea e inicia el ORB");  
        ORB orb = ORB.init(args, null);
```

```
        System.out.println("2. Obtiene una referencia al servicio de nombrado por medio del orb");  
        org.omg.CORBA.Object objRefNameService = orb.resolve_initial_references("NameService");
```

```
        System.out.println("3. Convierte la ref genérica a ref de NamingContextExt");  
        NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);
```

3. Convertir la referencia genérica a una referencia correspondiente a NamingContextExt

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
  
import SegundoEjemplo.*;
```

Paquetes a utilizar propios de la especificación de CORBA

```
System.out.println("4. Resuelve la referencia del objeto en el N_S.");  
  
String identificadorServant = "identificadorServant";  
  
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

4.1 Construir el **contexto de nombres** que identificara al servant

```
org.omg.CORBA.Object objRef= refContextoNombrado.resolve(path);
```

4.2 Localizar la referencia del objeto remoto en el naming Service

```
System.out.println("5. Convierte la referencia de un objeto generico a una referencia al servant ");  
Registro objSolucion = RegistroHelper.narrow(objRef);
```

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import SegundoEjemplo.*;
```



Paquetes a utilizar propios de la especificación de CORBA

```
System.out.println("4. Resuelve la referencia del objeto en el N_S.");
```

```
String identificadorServant = "identificadorServant";
```

```
NameComponent [] path = new NameComponent[1];
```

```
path[0] = new NameComponent();
```

```
path[0].id = identificadorServant;
```

```
path[0].kind = "tipoServicio";
```

```
org.omg.CORBA.Object objRef= refContextoNombrado.resolve(path);
```

5. Convierte la referencia de un objeto genérico a una referencia al servant



```
System.out.println("5. Convierte la referencia de un objeto generico a una referencia al servant ");
```

```
Registro objSolucion = RegistroHelper.narrow(objRef);
```

Ejemplo cliente CORBA

Desarrollo del cliente de objetos (main)

```
Registro objSolucion = RegistroHelper.narrow(objRef);

System.out.println("Invocación de los métodos como si fueran locales");

objSolucion.cantidadMaximaDepositos(3);
objSolucion.registrarDeposito("1", 250000);
objSolucion.registrarDeposito("1", 500000);
objSolucion.registrarDeposito("2", 1500000);
float saldoCuenta=objSolucion.cantidadCuenta("1");
System.out.println("El saldo de la cuenta para la identificacion 1 es: " + saldoCuenta );
int cantidadDepositos= objSolucion.cantidadDepositosRegistrados();
System.out.println("La cantidad de depositos registrados es: " + cantidadDepositos);
```

6. Invocación de los métodos como si fueran locales, por medio de la referencia almacenada en objSolucion



Compilación y ejecución del cliente y servidor CORBA

4. Compilar los códigos fuente

Compilar los soportes que fueron generados por el precompilador idlj en el paso 1, por medio de los siguientes comandos:

Para compilar los soportes:

```
javac -d ../bin sop_corba/*.java
```

Para compilar los fuentes del servidor:

```
javac -d ../bin servidor/*.java
```

Para compilar los soportes:

```
javac -d ../bin cliente/*.java
```

Compilación y ejecución del cliente y servidor CORBA

5. Lanzar el Object Request Broker Daemon (ORBD)

Herramienta que se utiliza para permitir a los clientes localizar e invocar objetos persistentes en servidores dentro del entorno CORBA.

Lanzar el orbd mediante el comando:

```
orbd -ORBInitialHost dir_IP -ORBInitialPort N_Puerto
```

6. Lanzar el servidor.

Ubicarse en el directorio bin/ . Lanzar el Servidor mediante el comando:

```
java servidor.Servidor -ORBInitialHost dir_IP -ORBInitialPort N_Puerto
```

dir_IP y **N_Puerto** son la dirección ip y puerto donde se encuentra escuchando el servicio orbd.

Compilación y ejecución del cliente y servidor CORBA

7. Lanzar el cliente

Ubicarse en el directorio bin/. Lanzar el Cliente mediante el comando:

```
java cliente.Cliente -ORBInitialHost dir_IP -ORBInitialPort N_Puerto
```

dir_IP y N_Puerto son la dirección ip y puerto donde se encuentra escuchando el servicio ordb.

Programación del servidor

Modelos de implementación servants

Delegación

- Se implementa la interfaz Operations (que sería de una clase Java normal que implementa las operaciones de la interfaz)
- Se debe generar una clase TIE la cual hace las funciones de skeleton y delega las peticiones a nuestra implementación
- Objetos de la clase TIE se encargan de realizar las tareas del skeleton y delegan la realización de la operación al objeto de implementación).
- La implementación no hereda de la clase interfacePOA. Esto le da flexibilidad
- Tiene la ventaja que la implementación puede ser una clase pre-existente.

Por delegación (enfoque TIE)

Ejemplo servidor CORBA por delegación

Desarrollo del servidor (servant por delegación)

```
public class RegistroImpl implements RegistroOperations{
```

```
    private final List<Deposito> listaDepositos;  
    int cantidadMaximaDepositos;
```

```
    public RegistroImpl()  
    {
```

```
        super();  
        System.out.println("Creando un objeto de tipo RegistroImpl");  
        cantidadMaximaDepositos=0;  
        listaDepositos=new ArrayList<>();  
    }
```

```
    @Override
```

```
    public int cantidadMaximaDepositos() {  
        return cantidadMaximaDepositos;  
    }
```

```
    @Override
```

```
    public void cantidadMaximaDepositos(int newCantidadMaximaDepositos) {  
        this.cantidadMaximaDepositos=newCantidadMaximaDepositos;  
    }
```

```
    @Override
```

```
    public boolean registrarDeposito(String identificacion, float cantidad) {  
        boolean bandera=false;  
        Deposito objNuevoDeposito= new Deposito();  
        bandera=listaDepositos.add(objNuevoDeposito);  
        return bandera;  
    }
```

```
    @Override
```

```
    public float cantidadCuenta(String identificacion) {  
        float cantidad=0;  
        for(Deposito objDeposito: listaDepositos)  
        {  
            if (objDeposito.getIdentificacion().compareTo(identificacion)==0)  
                cantidad+=objDeposito.getCantidad();  
        }  
        return cantidad;  
    }
```

```
    @Override
```

```
    public int cantidadDepositosRegistrados() {  
        return listaDepositos.size();  
    }
```

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import org.omg.PortableServer.*;  
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;  
public static void main(String args[])  
{
```

```
try{
```

```
    System.out.println("1. Crea e inicia el orb");  
    ORB orb = ORB.init(args, null);
```

```
    System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");  
    org.omg.CORBA.Object objPoa = null;  
    objPoa=orb.resolve_initial_references("RootPOA");  
    POA rootPOA = POAHelper.narrow(objPoa);
```

```
    System.out.println("3. Activa el POAManager");  
    rootPOA.the_POAManager().activate();
```

```
    ...
```

1. Crear e inicializar el ORB

La clase ORB hace parte del paquete org.omg.CORBA

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;
public static void main(String args[])
{
```

```
    try{
        System.out.println("1. Crea e inicia el orb");
        ORB orb = ORB.init(args, null);
```

2. Obtener la referencia al POA raíz.
Para encontrar el POA raíz se utiliza el nombre estándar "RootPOA"

```
        System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");
        org.omg.CORBA.Object objPoa = null;
        objPoa=orb.resolve initial references("RootPOA");
```

```
        POA rootPOA = POAHelper.narrow(objPoa);
```

```
        System.out.println("3. Activa el POAManager");
        rootPOA.the_POAManager().activate();
```

2.1 Convertir la referencia de un objeto genérico a una referencia al rootPOA

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
import org.omg.CosNaming.*;  
import org.omg.CORBA.*;  
import org.omg.PortableServer.*;  
import org.omg.PortableServer.POA;
```

Paquetes a utilizar propios de la especificación de CORBA

```
import SegundoEjemplo.*;  
public static void main(String args[])  
{  
    try{  
        System.out.println("1. Crea e inicia el orb");  
        ORB orb = ORB.init(args, null);  
  
        System.out.println("2. Obtiene la referencia al poa raiz, por medio del orb ");  
        org.omg.CORBA.Object objPoa = null;  
        objPoa=orb.resolve_initial_references("RootPOA");  
        POA rootPOA = POAHelper.narrow(objPoa);
```

```
        System.out.println("3. Activa el POAManager");  
        rootPOA.the_POAManager().activate();
```

3. Activar el gestor de invocaciones del POA, lo que permite recibir invocaciones a los objetos asociados.

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```

4. Crear el objeto servant

```
System.out.println("5. Crea el objeto tie y se registra una referencia al objeto servant");  
RegistroPOATie objTIE= new RegistroPOATie(ObjServant);
```

```
System.out.println("6. Obtiene la referencia al orb ");  
Registro referenciaORB = objTIE._this(orb);
```

Ejemplo servidor CORBA


Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```

```
System.out.println("5. Crea el objeto tie y se registra una referencia al objeto servant");  
RegistroPOATie objTIE= new RegistroPOATie(ObjServant);
```

```
System.out.println("6. Obtiene la referencia al orb ");  
Registro referenciaORB = objTIE._this(orb);
```

5. Crea el objeto tie y se registra una referencia al objeto servant mediante su constructor



Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("4. Crea el objeto servant");  
RegistroImpl ObjServant = new RegistroImpl();
```

```
System.out.println("5. Crea el objeto tie y se registra una referencia al objeto servant");  
RegistroPOATie objTIE= new RegistroPOATie(ObjServant);
```

```
System.out.println("6. Obtiene la referencia al orb ");  
Registro referenciaORB = objTIE._this(orb);
```

6. Los servant se activan con el método `_this()`, el cual devuelve una referencia del objeto CORBA asociado

Ejemplo servidor CORBA

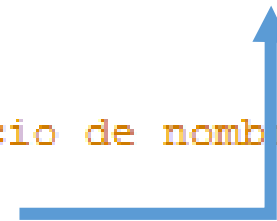
Desarrollo del servidor de objetos (main)

```
System.out.println("5. Crea el objeto tie y se  
RegistroPOATie objTIE= new RegistroPOATie(ObjS  
  
System.out.println("6. Obtiene la referencia a  
Registro referenciaORB = objTIE._this(orb);
```

```
System.out.println("7. Obtiene una referencia al servicio de nombrado por medio del or  
org.omg.CORBA.Object objRefNameService =  
    orb.resolve_initial_references("NameService");
```

7. Obtener una referencia al servicio de nombrado por medio del orb.

Para encontrar el servicio de nombrado se utiliza el nombre estándar "NameService"



Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow (objRefNameService);
```

```
System.out.println("9.Construir un contexto de nombres que identifica al servant");  
String identificadorServant = "identificadorServant";
```

```
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

```
System.out.println("10.Realiza el binding de la referencia de objeto en el N_S");  
refContextoNombrado.rebind(path, referenciaORB);
```

```
System.out.println("El Servidor esta listo y esperando ...");  
orb.run();
```

8. Convertir la referencia genérica a una referencia correspondiente a NamingContextExt

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow (objRefNameService);
```

```
System.out.println("9.Construir un contexto de nombres que ident  
String identificadorServant = "identificadorServant";  
  
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

```
System.out.println("10.Realiza el binding de la referencia de ob  
refContextoNombrado.rebind(path, referenciaORB);
```

```
System.out.println("El Servidor esta listo y esperando ...");  
orb.run();
```

9. Construir el **contexto de nombres** que identificara al servant

Cada componente del contexto se identifica por el par **identificador, tipo**

El cliente debe obtener una referencia al objeto remoto utilizando el mismo **contexto de nombres**

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");  
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow (objRefNameService);
```

```
System.out.println("9.Construir un contexto de nombres que identifica al servant");  
String identificadorServant = "identificadorServant";
```

```
NameComponent [] path = new NameComponent[1];  
path[0] = new NameComponent();  
path[0].id = identificadorServant;  
path[0].kind = "tipoServicio";
```

10. Realizar el binding de la referencia del objeto y el identificador en el N_S

```
System.out.println("10.Realiza el binding de la referencia de objeto en el N_S");  
refContextoNombrado.rebind(path, referenciaORB);
```

```
System.out.println("El Servidor esta listo y esperando ...");  
orb.run();
```

Ejemplo servidor CORBA

Desarrollo del servidor de objetos (main)

```
System.out.println("8. Convierte la ref genérica a ref de NamingContextExt");
NamingContextExt refContextoNombrado = NamingContextExtHelper.narrow(objRefNameService);

System.out.println("9.Construir un contexto de nombres que identifica al servant");
String identificadorServant = "identificadorServant";

NameComponent [] path = new NameComponent[1];
path[0] = new NameComponent();
path[0].id = identificadorServant;
path[0].kind = "tipoServicio";

System.out.println("10.Realiza el binding de la referencia de objeto en el N_S");
refContextoNombrado.rebind(path, href);
```

`orb.run();`



11. Ceder control al ORB para que escuche peticiones orb.run();

Aplicaciones CORBA

Invocación dinámica

Pasos para invocar operaciones dinámicamente

1. Obtener la descripción del método del almacén de interfaces.
 2. Crear la lista de argumentos. Utilizando la operación `create_list` y `add_arg`.
 3. Crear la petición. Debe especificar la referencia al objeto, el nombre del método y la lista de argumentos.
 4. Invocar la petición y esperar por los resultados.
 5. Recuperar los resultados.
- * El servidor y la implementación son los mismos! Lo que cambia es el cliente.

Aplicaciones CORBA

Invocación dinámica

Client Object

1. **Obtener el nombre de la interfaz**

```
get_interface()
```

2. **Obtener la descripción del método**

```
lookup_name ()
```

```
describe ()
```

3. **Crear la lista de argumentos**

```
create_list ()
```

```
add_item ()...add_item ()...add_item ()
```

4. **Crear la petición**

```
create_request ()
```

5. **Invocar el método remoto**

3 formas de hacer la invocación remota

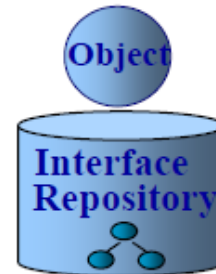
- invocación síncrona:
- o bien
- síncrona diferida:
- o bien
- asíncrona ("oneway"):

```
invoke ()
```

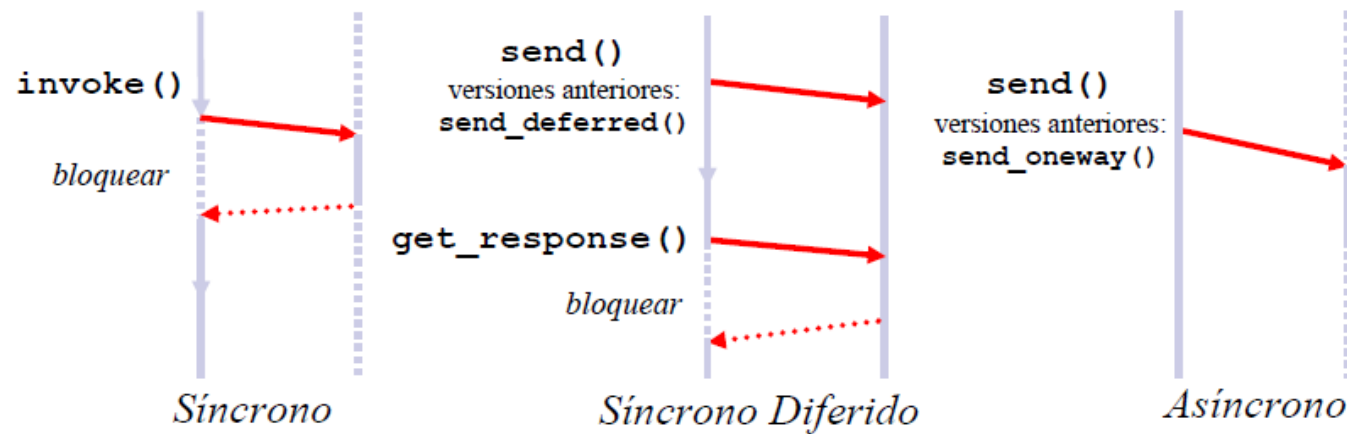
```
send ()
```

```
get_response ()
```

```
send ()
```



Aplicaciones CORBA Invocación dinámica



- ❖ **Invocación síncrona (bloqueante):** se envía la petición y se obtiene un resultado. (semántica de “*exactly-once*” y si hay una excepción “*at-most-once*”)
- ❖ **Invocación sincrónica diferida (no bloqueante):** disponible únicamente con invocación dinámica. Antes de invocar `get_response()`, se puede invocar `poll_response()` para saber si la respuesta está lista.
- ❖ **Invocación asíncrona o oneway:** Se utiliza en métodos sin respuesta. semántica de “*best-effort*”