



AVO Übung 3

Lösung 2, CORBA

30. November 2004 (WS 2004)

Andreas I. Schmied (schmied@inf...)

1 Aufgaben

1. Bank-Applikation

- Portieren Sie die RMI-Variante (avo2a) auf die CORBA-Plattform
- TCP-Ports müssen ggf. für den Namensdienst und den Bank-ORB geändert werden!
 - z.B. falls sich mehrere Studenten einen Host teilen
- RMI over IIOP soll nicht genutzt werden

2. Weitergabe von Objektreferenzen

- Finden Sie einen Weg, um dem Client eine initiale Referenz auf das Bank-Objekt zu geben, ohne den Namensdienst zu bemühen

3. Nutzen Sie POA-Policies, um die Bankkonten bzw. die Klienten effizient zu implementieren

4. Versuchen Sie sowohl für Client als auch Server alternativ JacORB zu verwenden

2 IDL-Generate

Beispiel: Bank.idl

- IDL-Datei (erzeugt mit `rmic -idl`)

```
module infvs { module avo4 { interface Bank {  
  
    Account    getAccount(in string number);  
    Client     getClient (in string number);  
    boolean    checkCredit(in Client cli, in long long value);  
  
    readonly attribute long long balance;  
    readonly attribute ::java::util::Iterator accounts;  
    readonly attribute ::java::util::Iterator clients;  
  
}; }; }
```

- Problem ?

2 IDL-Generate

Beispiel: Bank.idl (2)

- Problem: `java.util.Iterator` u.a. Java-Klassen/Ifc. nicht direkt abbildbar!
- Korrektur

```
interface Bank { ...
  readonly attribute long cntAccounts;
  readonly attribute long cntClients;
  Account  getAccountByIndex (in long idx);
  Account  getClientByIndex  (in long idx);
  sequence<Account> getAllAccounts ();      // Wow!
}
```

also:

- Java-Code vorausschauend entwickeln oder
- von Anfang an IDL-basiert arbeiten
- IDL-Java Mapping
 - erzeugen mit `idlj -fall -td src-gen <filename>.idl`

2 IDL-Generate

Generierte Dateien (Details)

- **org.omg.CORBA.portable.InputStream/OutputStream** verwendet ("...")
- Bank.java

```
public interface Bank extends BankOperations,  
    org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity  
{...}
```

- BankOperations.java

```
public interface BankOperations { ...  
  
    Account getAccount (String number);  
    Client  getClient  (String number);  
  
    long balance ();                // readonly attribute!  
}
```

2 IDL-Generate

- BankHolder.java

```
public final class BankHolder implements ...Streamable {  
    public Bank value = null;  
  
    public BankHolder () {}  
    public BankHolder (Bank initialValue) {...}  
  
    public void _read (...InputStream i) { value = BankHelper.read (i); }  
    public void _write (...OutputStream o) { BankHelper.write (o, value); }  
    public org.omg.CORBA.TypeCode _type () { return BankHelper.type (); }  
}
```

2 IDL-Generate

- BankHelper.java

```
abstract public class BankHelper
{
    private static String _id = "RMI:infvs.avo4.Bank:0000000000000000";

    public static void insert (org.omg.CORBA.Any a, Bank that)
    {
        ...OutputStream out = a.create_output_stream ();
        a.type (type ());
        write (out, that);
        a.read_value (out.create_input_stream (), type ());
    }

    public static Bank extract (org.omg.CORBA.Any a)
    { return read (a.create_input_stream ()); }

    public static Bank read (...InputStream istream)
    { return narrow (istream.read_Object (_BankStub.class)); }

    public static void write (...OutputStream ostream, Bank value)
    { ostream.write_Object ((org.omg.CORBA.Object) value); }
```

2 IDL-Generate

```
synchronized public static org.omg.CORBA.TypeCode type ()
{ return ORB.init().create_interface_tc(BankHelper.id (), "Bank"); }
```



```
public static infvs.avo4.Bank narrow (org.omg.CORBA.Object obj)
{
    if (obj == null)                return null;
    else if (obj instanceof Bank) return (Bank)obj;
    else if (!obj._is_a (id ()))    throw new org.omg.CORBA.BAD_PARAM ();
    else {

        org.omg.CORBA.portable.Delegate delegate
        = ((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();

        _BankStub stub = new _BankStub ();
        stub._set_delegate(delegate);

        return stub;
    }
}
```

2 IDL-Generate

- BankPOA.java

```
public abstract class BankPOA extends    org.omg.PortableServer.Servant
                                   implements BankOperations,
                                   org.omg.CORBA.portable.InvokeHandler
{
    private static Hashtable _methods = new Hashtable ();

    static {
        _methods.put ("getAccount", new java.lang.Integer (0));
        _methods.put ("getClient", new java.lang.Integer (2));
        ...
    }

    public Bank _this(ORB orb)
    { return BankHelper.narrow(super._this_object(orb)); }

    public Bank _this()
    { return BankHelper.narrow(super._this_object()); }
}
```

2 IDL-Generate

```
public ...OutputStream _invoke (String $method, ...InputStream in,
                                   org.omg.CORBA.portable.ResponseHandler $rh)
{
    ...OutputStream out = null;
    Integer __method = (Integer)_methods.get ($method);
    if (__method == null) throw ...BAD_OPERATION...;

    switch (__method.intValue ()) { // Hashing: effizienter als elseif-Kette

        case 0: // infvs/avo4/Bank/getAccount(int number)
            {
                String arg0 = org.omg.CORBA.WStringValueHelper.read (in);
                Account $result = this.getAccount (arg0);
                out = $rh.createReply();
                AccountHelper.write (out, $result);
                break;
            }
            ...
        default: throw ...BAD_OPERATION...;
    }
    return out;
}
}
```

2 IDL-Generate

- `_BankStub.java`

```
public class _BankStub extends ...CORBA.portable.ObjectImpl implements Bank
{
    public Account getAccount(String number)
    {
        ...OutputStream $out = _request ("getAccount", true);
        org.omg.CORBA.WStringValueHelper.write ($out, number);
        ... InputStream $in = _invoke ($out);
        return AccountHelper.read ($in);
    }
    public long balance ()
    {
        ...OutputStream $out = _request ("_get_balance", true);
        ...InputStream $in = _invoke ($out);
        return $in.read_longlong();
    }

    private void readObject (java.io.ObjectInputStream s) ...
    {
        org.omg.CORBA.Object obj = ORB.init().string_to_object (s.readUTF());
        org.omg.CORBA.portable.Delegate delegate
        = ((org.omg.CORBA.portable.ObjectImpl) obj)._get_delegate ();
        _set_delegate (delegate);
    }
    private void writeObject (java.io.ObjectOutputStream s) ...
    { s.writeUTF (ORB.init().object_to_string (this)); }
}
```

2 IDL-Generare

- Stub nutzt Streaming-Schnittstelle
- alternativ: DSI/DII
 - erzeugen von Request/Reply-Objekten
 - anhängen von Parameter-Objekten
- Streaming/DSI/DII erzeugen untereinander kompatible Datenströme

3 Interoperable Objekt-Referenzen

Weitergabe von Objektreferenzen

- Objektreferenzen in String umwandeln

```
class org.omg.CORBA.ORB {  
    String object_to_string(Object o);  
    Object string_to_object(String s);  
}
```

- String ist üblicherweise IOR (Interoperable Object Reference)
- serialisierte org.omg.IOP.IOR-Struktur (definiert in IDL)

```
struct IOR {  
    string type_id;  
    sequence <TaggedProfile> profiles;  
};
```

3 Interoperable Objekt-Referenzen

Unterstruktur:

```

struct TaggedProfile {
    ProfileId      tag;           // org.omg.IOP.TAG_INTERNET_IOP=0
    sequence<octet> profile_data;
};

struct ProfileBody_1_1 {           // IIOP-Profile
    Version        iiop_version;
    string          host;
    unsigned short port;
    sequence<octet> object_key;
    sequence <IOP::TaggedComponent> components;
};

```

- Textprefix "IOR:" + serialisierte Daten als Hexstring (0xZZ -> "ZZ")

```

IOR:00
000000
00000021
49444c3a746573742f6c6f61642f73696d706c652f476f6f644461793a312e3000
000000
00000001
...

```

4 POA

Fähigkeiten und Konfiguration

- aktiviert/deaktiviert Objekte, vergibt OIDs
- kann Kind-POAs haben (rekursiv)
- kann DefaultServant haben: `POA.set_servant(Servant)`
- kann ServantManager haben: `POA.set_servant_manager(ServantManager)`
 - Varianten: Activator und Locator

```
interface ServantActivator { Servant incarnate(byte[] oid, POA adapter); ... }
interface ServantLocator  { ...
    Servant preinvoke (byte[] oid, POA poa, String operation, ...);
    void postinvoke(byte[] oid, POA poa, String operation, ..., Servant);
}
```

- konfigurierbar über Policies
- POAManager kontrolliert Zustand registrierter POAs (de-/activate, hold, discard, ...)
- AdapterActivator startet abh. POAs bei Bedarf

4 POA

Policies

- POA erzeugen und mittels Policies konfigurieren

IdAssignmentPolicy

OIDs von System oder Benutzerabhängig vergeben

```
void POA.activate_object_with_id(byte[] id, Servant p_servant)
```

IdUniquenessP.

ein oder mehrere OIDs pro Servant

ImplicitActivationP.

automatische Aktivierung - nur RootPOA hat diese Policy voreingestellt

LifespanP.

transiente oder persistente Servants

RequestProcessingP.

Anfrage an bestehende (aktive) Servants, DefaultServant oder ServantManager

ServantRetentionP.

Servants in ActiveObjectMap für spätere Anfragen vorhalten

4 POA

ThreadPolicy

Thread-Scheduling durch ORB oder SingleThreaded-POA

- relevante Objekte am neuen POA aktivieren

Konfiguration für Bank-Applikation

- (jeweils) eigener POA für Clients/Accounts
- `IdAssignmentPolicyValue.USER_ID`
- `POA.activate_object_with_id("KtoNr/CliNr", ...)`
- `RequestProcessingPolicyValue.USE_SERVANT_MANAGER`
- `ServantManager` als

ServantActivator

lädt angesprochene Objekte (z.B. aus DB)

ServantLocator

kann Objekte mit `pre/postinvoke` simulieren

- ...