C überblick über die 12. übung

- Besprechung der 7. Aufgabe
- .NET Anwendungen bauen und binden

C.1 Besprechung der 7. Aufgabe

- 25 Punkte:
  - 9 Punkte Client
  - 16 Punkte Server (8 davon eigener POA)
- Abgegebene Lösungen: 19
- Häufigster Fehler im SS2002:

  FEHLENDE SYNCHRONISATION

  - CORBA ist Multithreaded, mehrere Clients können gleichzeitig auf den selben Servant zugreifen!

C.2 Lösung der Aufgabe 7

- library.server.Main (1/4)

```
package library.server;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.PortableServer.*;
import library.*;
import library.database.BookDatabase;
import java.io.*;

public class Main {
  public static void main( String[] args ) {
    try {
      // Initialize ORB
      ORB orb = ORB.init( args, null );

      // Initialize Book Database
      BookDatabase database = new BookDatabase()

      // Find the root POA & start it
      POA rootPOA =POAHelper.narrow(
                   orb.resolve_initial_references("RootPOA");
      rootPOA.the_POAManager().activate();
```


C.2 Lösung der Aufgabe 7

- library.server.Main (2/4)

```
      // Set policies for POA that uses servant locator
      Policy[] policies = new Policy[] {
         rootPOA.create_id_assignment_policy(
             IdAssignmentPolicyValue.USER_ID),
         rootPOA.create_servant_retention_policy(
             ServantRetentionPolicyValue.NON_RETAIN),
         rootPOA.create_request_processing_policy(
             RequestProcessingPolicyValue.USE_SERVANT_MANAGER)
      };
      POA bookPOA = rootPOA.create_POA("bookPOA",
           rootPOA.the_POAManager(), policies);
      // null => neuer POAManager => muss aktiviert werden!

      // Create a Librarian
      LibrarianServant libServ = new LibrarianServant(
                                        rootPOA, bookPOA, database);
      Librarian lib = LibrarianHelper.narrow(
                        rootPOA.servant_to_reference(libServ));
      // deprecated: lib = libServ._this(orb)
      // bzw: poa.id_to_reference(poa.activate_object(.))
```

C.2 Lösung der Aufgabe 7

- library.server.Main (3/4)

```
        // Register at name service
        // Read the reference
        BufferedReader br =
         new BufferedReader(
         new FileReader(
            "/proj/i4oovs/pub/aufgabe7/Nameservice.ior" ) );
        String ior = br.readLine();
        // Create a stub object
        org.omg.CORBA.Object obj_root_context =
                             orb.string_to_object( ior );

        // Get name service reference
        org.omg.CORBA.Object obj_root_context =
           orb.resolve_initial_references("NameService");

        NamingContext root_context =
           NamingContextHelper.narrow(obj_root_context);
```


C.2 Lösung der Aufgabe 7

- library.server.LibrarianServant (1/4)

```
package library.server;

import library.*;
import library.database.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import java.util.*;

public class LibrarianServant extends LibrarianPOA
{
    private POA     my_poa, book_poa;
    private BookDatabase my_db;
    private POA     book_poa;
    private Vector  my_cards;
    private ORB     my_orb;

    public LibrarianServant( ... ) { ... }
```


C.2 Lösung der Aufgabe 7

- library.server.Main (4/4)

```
        // Register new Namingcontext
        try {
        root_context.bind_new_context(  new NameComponent[]
                          { new NameComponent("oovs", "")});
        } catch (.....AlreadyBound e)
        { System.err.println("   already registered");}
        // Create a name as an array of NameComponents
        NameComponent name[] = new NameComponent[2];
        name[0] = new NameComponent("felser", "");
        name[1] = new NameComponent("Librarian", "");
        // Register the Librarian object
        root_context.rebind(name, lib);

        System.out.println( "server up and running." );
        // Wait for requests
        orb.run();
    } catch( Exception ex )
    { ex.printStackTrace();}
}
```


C.2 Lösung der Aufgabe 7

- library.server.LibrarianServant (2/4)

```
public Card issueCard( String owner ) throws DeniedCard
{
    synchronized( my_cards ) {
        Card c = searchCard( owner );
        if(c!=null) throw new DeniedCard("Card exists");

        try {
          CardServant cServ = new CardServant(owner,my_poa);
          c = CardHelper.narrow(
                      my_poa.servant_to_reference(cServ));
          my_cards.addElement( c );
          return c;
        }
        catch( Exception ex ) {
           throw new DeniedCard( "Cannot create Card" );
        }
    }
}
```

C.2 Lösung der Aufgabe 7

- library.server.LibrarianServant (3/4)

```
public Card findCardByOwner( String owner ) throws NoSuchCard
{
    Card c = searchCard( owner );
    if( c == null )throw new NoSuchCard();
    return c;
}


private Card searchCard( String owner ) {
    synchronized( my_cards ) {
        Enumeration e = my_cards.elements();
        while(e.hasMoreElements()) {
            try {
                Card c = (Card) e.nextElement();
                if( c.owner().equals( owner ) ) return c;
            } catch( SystemException ex ) {
                // This card is no accessible, so ignore it
            }
        }
    }
    return null;
}
```


C.2 Lösung der Aufgabe 7

- library.server.CardServant (1/2)

```
package library.server;
import library.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

public class CardServant extends CardPOA {
    private java.util.Vector bookvect = new Vector();
    public synchronized Book[] borrowedBooks() {
        Books[] ret = new Book[bookvect.size()];
        for(int i=0; i<bookvect.size(); i++) {
            ret[i] = (Book)bookvect.elementAt(i);
        }
        return ret;
    }
    public synchronized returnBook(Book b)
        throws BookNotBorrowed
    {
        if(bookvect.remove(b)==true) {
            try { b.unsetBorrowed(); } catch(...)
        } else throw new BookNotBorrowed();
    }
```


C.2 Lösung der Aufgabe 7

- library.server.LibrarianServant (4/4)

```
public Book[] findBookByTitle( String title ) {
    Vector matches = new Vector();
    Book[] books = null;
    try {
        int[] bookids = my_db.booksByTitle(s_title);
        books = new Book[bookids.length];
        for(int i=0; i<bookids.length; i++) {
            byte[] oid = BookServantLocator.int2oid(bookids[i]);
            org.omg.CORBA.Object o =
                book_poa.create_reference_with_id(oid,
                    BookHelper.id()); //"IDL:library/Book:1.0";
            books[i] = BookHelper.narrow(o);
        }
    } catch(Exception ex) {
        ex.printStackTrace();
        return new Book[0];
    }
    return books;
}
```


C.2 Lösung der Aufgabe 7

- library.server.CardServant (2/2)

```
    public synchronized void borrowBook( Book b )
        throws BookBorrowed, CardExpired
    {
        Card c = CardHelper.narrow(
            my_poa.servant_to_reference(this));
        b.setBorrowed(c);

        bookvect.addElement( b );
    }
}
```

C.2 Lösung der Aufgabe 7

- library.servant.BookServant (1/2)

```
package library.server;
import library.*;
import java.util.*;

public class BookServant extends BookPOA
{
    private ...;               // Instanzvariablen
    public BookServant( ... )  // Konstruktor, mit Init.
    public String title();
       [...]                   // Lesen v. Attributen

    public synchronized Card borrowedBy()
                                throws BookNotBorrowed
    {
        if( my_card != null ) {
            return my_card;
        }
        else { throw new BookNotBorrowed();  }
    }
```


C.2 Lösung der Aufgabe 7

- library.server.BookServantLocator (1/5)

```
package library.server;

import ...

public class BookServantLocator extends ServantLocatorPOA
    // oder: extends LocalObject implements ServantLocator
{
    ORB myorb;
    BookDatabase mydb;

    static final int MAXSIZE=10;
    Hashtable hash = new Hashtable();
    Vector removable = new Vector();

    public BookServantLocator(ORB orb, BookDatabase db) {
        myorb = orb;
        mydb = db;
    }
```


C.2 Lösung der Aufgabe 7

- ibrary.servant.BookServant (2/2)

```
    public synchronized void setBorrowed( Card c )
        throws BookBorrowed {

        if( my_card != null ) {
            throw new BookBorrowed( my_card );
        }
        my_card = c;
    }

    public synchronized void unsetBorrowed()
        throws BookNotBorrowed {

        if( my_card == null ) {
            throw new BookNotBorrowed();
        }
        my_card = null;
    }
}
```


C.2 Lösung der Aufgabe 7

- library.server.BookServantLocator (2/5)

```
    public static int oid2int(byte[] oid) {
        int val = 0;
        for(int i=0; i<oid.length; i++) {
            val=val*256+oid[i];
        }
        return val;
    }
    public static byte[] int2oid(int val) {
        byte oid[] = new byte[4];
        oid[3] = (byte)(val % 256); val = val / 256;
        oid[2] = (byte)(val % 256); val = val / 256;
        oid[1] = (byte)(val % 256); val = val / 256;
        oid[0] = (byte)(val % 256); val = val / 256;
        return oid;
    }
    // funktioniert im Prinzip auch: (Effizient?)
    // o=new DataOutputStream(new ByteArrayOutputStream())
    // o.writeInt(val); return o.toByteArray();
```

C.2 Lösung der Aufgabe 7

- library.server.BookServantLocator (3/5)

```
public org.omg.PortableServer.Servant preinvoke(
     byte[] oid, POA adap, String op, CookieHolder cookie)
    throws ForwardRequest
{
    Integer i = new Integer(oid2int(oid));
    BookServant book;
    synchronized(hash) {
        book = (BookServant)hash.get(i);
        if(book!=null) {  // Im Cache vorhanden
            removable.removeElement(i);
            return book;
        }
        Record r = mydb.getBook(index);
        String borrowedBy = r.getData("BORROWED_BY");
        Card card = null;
        if(borrowedBy!=null && !borrowedBy.equals("")) {
            card = CardHelper.narrow(
                        myorb.string_to_object(borrowedBy));
        }
        book = new BookServant(
                r.getData("REGISTRATION"), r.getData("AUTHORS"),
                r.getData("TITLE"), r.getData("PUBLISHER"),
                Short.parseShort(r.getData("YEAR")), card);
```


C.2 Lösung der Aufgabe 7

- library.server.BookServantLocator (4/5)

```
        while(hash.size()>MAXSIZE) {
            if(removable.size()==0) break;
            Integer ri = (Integer)removable.remove(0);
            remove(ri);
        }
        hash.put(i, book);
    }
    return book;
}

private void remove(Integer id) {
    Record r = mydb.getBook(id.intValue());
    String borrower="";
    try {
        BookServant bs = (BookServant)hash.get(id);
        Card c = bs.borrowedBy();
        borrower = myorb.object_to_string(c);
    } catch(BookNotBorrowed ex) { }
    r.setData("BORROWED_BY", borrower);
    hash.remove(id);
}
```


C.2 Lösung der Aufgabe 7

- library.server.BookServantLocator (5/5)

```
public void postinvoke(byte[] oid,
             POA adapter,
             String operation,
             java.lang.Object the_cookie,
             org.omg.PortableServer.Servant the_servant)
{
    if(the_servant instanceof BookServant) {
        Integer i = new Integer(oid2int(oid));
        synchronized(hash) {
            removable.addElement(i);
        }
    }
}
```


D .NET Anwendungen bauen und binden

D.1 Assemblies

- Anwendung oder Bibliothek
  z.B. unter Windows: ausführbare Datei!?
- entspricht einer Sammlung von Klassen, Strukturen und Typen
- Assembly enthält platformunabhängigen Code
  - *Common Intermediate Language* (CIL)
  - wird von JIT-Compiler übersetzt
  - → *portable execution file* (PE)

D.1 Assemblies (2)

- private und gemeinsam benutzte Assemblies
- Metadaten werden im *Manifest* dem Assembly beigefügt
  ➔ Assembly ist selbstbeschreibend
- Manifest enthält Metadaten der enthaltenen Typen und folgenden Informationen:
  - ◆ Version
  - ◆ Umgebungsinformationen (Sprache)
  - ◆ referenzierte Typen
  - ◆ Abhängigkeiten
- *multifile Assembly*: besteht aus mehreren Dateien

D.2 Beispiel: .NET im CIP-Pool

- SharedSource CLI von Microsoft
  offiziell nur für Windows XP, FreeBSD, und Mac OS X 10.2.
- Linuxversion im CIP-Pool unter: **`/local/sscli`**
  - **`csc`**: C#-Compiler
  - **`clix`**: CLI Ausführungsumgebung
- Umgebungsvariablen setzen: **`env.csh`** oder **`env.sh`**:

```
> cd /local/sscli
> source env.csh
Fastchecked Environment
> cd /proj/i4oovs/felser/net
> csc Hello.cs
Microsoft (R) Visual C# Shared Source CLI Compiler version 1.0.0002
for Microsoft (R) Shared Source CLI version 1.0.0
Copyright (C) Microsoft Corporation 2002. All rights reserved.

> clix Hello.exe
Hello my name is Zaphod Beeblebrox
```


D.2 Beispiel: BankLibrary

- Die Bibliothek: Bank.cs

```
namespace BankLibrary {

   public class Bank {
     public static int deposit (Account account, int amount){
       return account.deposit(amount);
     }
   }

   public class Account {
     private int value = 0;
     public int deposit( int amount ) {
       value += amount;
       return value;
     }
   }

}
```

- Bibliothek erstellen

```
> csc /target:library Bank.cs
```


D.2 Beispiel: BankClient

- Anwendung: Customer.cs

```
using System;
namespace Customer {

    class Customer {
        public static void Main(String[] args) {
            BankLibrary.Account account =
                                    new BankLibrary.Account();
            Console.WriteLine("deposit 3 -> account = {0}",
                    BankLibrary.Bank.deposit(account, 3));
            Console.WriteLine("deposit 5 -> account = {0}",
                    BankLibrary.Bank.deposit(account, 5));
        }
    }

}
```

- Anwendung erstellen

```
> csc /reference:Bank.dll Customer.cs
```

- Anwendung starten

```
> clix Customer.exe
```

D.3 Manifest

- enthält Metainformationen und Referenzen auf andere Assemblies
- Manifest eines Assemblies betrachten:

```
> ildasm Customer.exe
```

- Ausgabe:

```
//  Microsoft (R) .NET Framework IL Disassembler. ...
//  Copyright (C) Microsoft Corporation 1998-2002. ...

.assembly extern mscorlib
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 1:0:3300:0
}
.assembly extern Bank
{
    .ver 1:0:0:1
}

...
```


D.4 Auflösen der Abhängigkeiten

- Suchen der referenzierten Assemblies (*probing*)
- Teil des Bindungsprozesses
- beeinflussbar durch Anwendungskonfigurationsdatei

D.4 Die Anwendungskonfigurationsdatei

- Ort: Verzeichnis der Anwendung
- Name: <Name der Anwendung>.config
  - Beispiel.: **`Customer.exe.config`**
- Format: XML
- Aufgabe: enthält z.B. Informationen für den Bindevorgang
- Beispiel: **`Customer.exe.config`**

```
<configuration>
   <runtime>
      <assemblyBinding
                  xmlns="urn:schemas-microsoft-com:asm.v1">
        <probing privatePath="libs;libs/morelibs" />
      </assemblyBinding>
   </runtime>
</configuration>
```


D.4 Der Probing Vorgang

- in Pseudocode:

```
function ProbeForAssembly
                  ( AsmName, AppBase, Culture, PrivatePath)

   for each EXT in {"dll", "exe"}
      search (AppBase\AsmName.EXT);

      if (Culture == "neutral")
         search(AppBase\AsmName\AsmName.EXT);
      else
         search(AppBase\Culture\AsmName.EXT);
         search(AppBase\Culture\AsmName\AsmName.EXT);
      end if

      for each PATH in PrivatePath
         if (Culture == "neutral")
            search(AppBase\PATH\AsmName\AsmName.EXT);
         else
            search(AppBase\PATH\Culture\AsmName.EXT);
            search(AppBase\PATH\Culture\AsmName\AsmName.EXT);
         end if
      next PATH
   next EXT
end function
```

D.5 gemeinsam genutzte Assemblies

- Identifikation: *starker Name* bestehend aus:
  - einfacher Textname (*friendly name*)
  - Umgebungsinformationen (*culture*)
  - Versionsnummer
  - öffentlicher Schlüssel
  - digitale Signatur
- Umgebungsinformation ist meist "neutral" nur bei *satellite assemblies* ist die Sprache relevant.
- Vorteile eines starken Namens:
  - eindeutiger Name
  - mehrere Versionen möglich
  - Integritätsprüfung möglich
  - Herkunft überprüfbar

D.5 gemeinsam genutzte Assemblies (2)

- Position: *Global Assembly Cache* (GAC) (z.B. c:/windows/assembly)
- Assembly in GAC installieren:
  - mit dem Windows Explorer
  - mittels gacutil:

```
> gacutil /i Bank.dll
```


D.5 Signieren

- Schlüsselpaar erzeugen mittels *Strong Name Utility* (sn.exe)

```
> sn -k MyKey.snk
```

- in einem beliebigen Source-File (z. B. in AssemblyInfo.cs) die Schlüssel-Datei mittels **`AssemblyKeyFile`**-Attribut angeben:

```
using System.Reflection;
[assembly: AssemblyKeyFile(@"MyKey.snk")]
```

```
> csc /target:library Bank.cs
```

- oder mittels Assemblylinker hinzufügen:

```
> csc /target:module Bank.cs
[...]
> al /out:Bank.dll /keyfile:MyKey.snk Bank.netmodule
```


D.5 Signieren (2)

- Manifest der Bibliothek vor dem Signieren:

```
.assembly Bank
{
    .hash algorithm 0x00008004
    .ver 1:0:0:1
}
```

- ... und danach:

```
.assembly Bank
{
  .publickey = (00 24 00 00 04 80 00 00 94 00 00 00 06 02 00 00
                00 24 00 00 52 53 41 31 00 04 00 00 01 00 01 00
                6F 9A D3 D0 71 02 DA C8 AB B1 56 30 E7 30 2D 72
                3F 89 01 86 9A BB 7A 14 9D D6 C2 E4 17 D4 73 7F
                DD D4 0F C1 7A 0D 5F 3A 7E 5A 08 B3 B5 7F BD A6
                BA AD BC 80 8F 5F 73 E7 B0 4F 12 84 D5 CD 72 3B
                E2 93 6F 02 FE 23 C1 31 2E FE 40 DA 77 95 23 A9
                3B 37 1E F8 2B 0F 45 A8 1C 87 CF B5 84 24 12 5E
                60 6B 97 9D C1 73 FC 8F 3A 41 12 C0 89 87 CA E3
                EF F6 94 D6 E0 37 64 21 BA 06 E5 3B 05 6D 82 C2)
  .hash algorithm 0x00008004
  .ver 1:0:0:1
}
```

D.5 Signieren (3)

- Manifest einer Anwendung...
  - ...mit einer Referenz auf eine private Bibliothek:

```
.assembly extern Bank
{
  .ver 1:0:0:1
}
```

  - ... mit einer Referenz auf eine signierte Bibliothek:

```
.assembly extern Bank
{
  .publickeytoken = (B4 26 2B 47 22 84 93 46 )
  .ver 1:0:0:1
}
```


D.6 Versionierung

- Versionsnummer besteht aus 4 Teilnummern durch Punkt (.) oder Doppelpunkt (:) getrennt:
  <major version>.<minor version>.<buildnumber>.<revision>
- wird durch ein Attribut festgelegt

```
[assembly: AssemblyVersion("1.0.0.1")]
```

- kann auch nur teilweise festgelegt werden:

```
[assembly: AssemblyVersion("1.0.*")]
```

- Keine Versionskontrolle bei privaten Assemblies
- Versionskontrolle beim Binden von existierenden Anwendungen beeinflussen durch:
  - die Anwendungskonfigurationsdatei
  - die systemweite Konfiguration
  - eine Richtliniendatei des Herstellers (*publisher policy*)

D.5 Verzögertes Signieren

- öffentlichen Schlüssel aus Schlüsselpaar extrahieren:

```
> sn -p MyKey.snk MyPublicKey.snk
```

- in Quellcode:

```
using System.Reflection;
[assembly: AssemblyDelaySigning(true)]
[assembly: AssemblyKeyFile(@"MyPublicKey.snk")]
```

- Schlüsselverifikation abschalten

```
> sn -Vr Bank.dll
```

- ... und wieder aktivieren

```
> sn -Vu Bank.dll
```

- Bibliothek nachträglich signieren:

```
> sn -R Bank.dll MyKey.snk
```


D.6 Umleiten von Assemblyversionen durch die Anwendungskonfigurationsdatei

- Customer.exe.config

```
<configuration>
   <runtime>
      <assemblyBinding
                  xmlns="urn:schemas-microsoft-com:asm.v1">
         <probing privatePath="libs' />
         <dependentAssembly>
            <assemblyIdentity name="Bank"
                        publicKeyToken="b4262b4722849346" />
            <bindingRedirect oldVersion="1.0.0.1-1.0.0.2"
                             newVersion="1.0.0.3" />
         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>
```

D.6 Umleiten von Assemblyversionen durch die systemweite Konfiguration

- Systemweite Konfigurationsdatei:
  <*.NET install path*>\Config\Machine.config
- Beispiel:
  `c:\windows\Microsoft.NET\Framework\v1.0.3705\CONFIG\Machine.config`
  oder
  `/local/sscli/build/v1.x86fstchk.rotor/config/machine.config`
- Unter Windows mittels mscorcfg.msc editieren

```
> mmc mscorcfg.msc
```


D.6 Umleiten von Assemblyversionen

- Versionsumleitung ist nicht schachtelbar:
  - Beispiel
    - Anwendungskonfiguration
      leitet Version **1.0.0.0 auf** Version **1.0.0.1** um
    - Systemkonfiguration
      leitet Version **1.0.0.1 auf** Version **1.0.0.2** um
  - trotzdem wird die Version 1.0.0.1 verwendet
- Konfigurationsvorrang:
  (1) Anwendungskonfiguration
  (2) Richtliniendatei
  (3) Systemkonfiguration

D.6 Umleiten von Assemblyversionen durch eine Richtliniendatei (publisher policy)

- Assembly mit speziellem Namen:
  policy.<*major*>.<*minor*>.<*assemblyname*>
- Name entspricht der Version für welche die Konfiguration angewendet werden soll
- wird aus einer XML-Datei durch den Assembly Linker (al.exe) gebaut

```
> al /link:PublisherPolicy.xml
     /out:policy.1.0.Bank.dll
     /keyfile:MyKey.snk
     /v:1.0.0.0
```

- Format: XML, wie Anwendungskonfigurationsdatei

D.6 Anwendung der Richtliniendatei unterdrücken

- Verarbeitung der Richtliniendatei kann durch Anwendungskonfiguration unterdrückt werden.
  - für die gesamte Anwendung: (Eintrag unter <*assemblyBinding*>)

```
<configuration>
    <runtime>
        <assemblyBinding
                    xmlns="urn:schemas-microsoft-com:asm.v1">
            <publisherPolicy apply="no" />
        </assemblyBinding>
   </runtime>
</configuration>
```

  - für einzelne Bibliotheken (Eintrag unter <*dependentAssembly*>):

```
<configuration>
    <runtime>
        <assemblyBinding
                    xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="Bank"
                                  publicKeyToken="b4262b4722849346" />
                <publisherPolicy apply="no" />
            </dependentAssembly>
        </assemblyBinding>
   </runtime>
</configuration>
```

D.7 Position der Assemblies

- GAC
- Anwendungsverzeichnis oder ein Unterverzeichnis
  - `<probing privatePath="libs' />`
- beliebiges Verzeichnis oder Web-Seite:

```
[...]
       <dependentAssembly>
            <assemblyIdentity name="Bank"
                        publicKeyToken="b4262b4722849346" />
            <bindingRedirect oldVersion="1.0.0.1-1.0.0.2"
                             newVersion="1.0.0.3" />
            <codeBase version="1.0.0.3"
                      href="file://c:/MyLib.dll"/>
       </dependentAssembly>
[...]
```

  - privaten Assemblies müssen im Anwendungsverzeichnis oder in einem Unterverzeichnis liegen.

D.7 Binden - Übersicht