# — Übungsblatt 4 (T): Nachrichten und Java Enterprise —

## Aufgabe 1 (Nachrichtenbasierte Middleware) (T)

1. Wie Unterscheidet sich der asynchrone Nachrichtenaustausch von entfernten Aufrufen?
2. Welche Nachteile entfernter Aufrufe sollen damit umgangen werden?
3. Wie funktionert der asynchrone Nachrichtenaustausch?
4. Was ist das Point-To-Point Verfahren und wie unterscheidet es sich von Broadcasts?
5. Wie können synchrone Aufrufe mit Warteschlangen abgewickelt werden?
6. Was sind Priorität und Lebensdauer einer Nachricht?
7. Wie kann der Message Broker die Zustellung einer Nachricht sicherstellen?
8. Skizzieren Sie den Aufbau eines `MOM`-Servers.

## Aufgabe 2 (Java Naming and Directory Interface) (T)

1. Was ist `JNDI` und wofür wird es verwendet?
2. Welche Bedeutung hat der Initial Context in `JNDI`?
3. Was bewirken die beiden Operationen `Lookup` und `Bind` in einem Kontext?
4. Was ist ein Service Provider in `JNDI`?
5. Welche Konfiguration benötigt der Initial Context, damit die Verbindung zu einem Namendienst hergestellt werden kann?

## Aufgabe 3 (Namen in JNDI) (T)

`JNDI` unterscheidet die folgenden drei Arten von Namen. Erklären Sie jede Art, indem Sie mit wenigen Sätzen auf die grundlegenden Eigenschaften eingehen und diese an jeweils zwei Beispiel aufzeigen.

1. Elementare Namen
2. Zusammengesetzte Namen
3. Zusammengefasste Namen

## Aufgabe 4 (Code-Magneten) (P)

Oh nein, nicht schon wieder! Nach längerer Zeit haben Sie es endlich geschafft, den Verbindungsaufbau mit JMS richtig hinzubekommen, als ein Bösewicht die ganzen Quellcodes zerschnippelt und Papiermännchen daraus gebastelt hat. Zum Glück ist nur eine Klasse betroffen, die Sie nun rekonstruieren müssen.

Ordnen Sie die Schnippsel in der richtigen Reihenfolge an. Da Sie beim Staubsaugen aus Versehen einige der geschweiften Klammern eingesaugt haben, müssen Sie diese an den passenden Stellen ergänzen.

Schnippsel 1

```
jndi = new InitialContext();
```

Schnippsel 2

```
// Nachrichten verschicken
Message m1 = session.createTextMessage("Hallo du da!");
queueProducer.send(m1);

Message m2 = session.createTextMessage("Hallo Leute!");
topicProducer.send(m2);
```

Schnippsel 3

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

Schnippsel 4

```
try {
```

Schnippsel 5

```
ConnectionFactory connectionFactory = (ConnectionFactory)
                  jndi.lookup("/ConnectionFactory");
```

Schnippsel 6

```
import javax.jms*;
import javax.naming.*;
```

Schnippsel 7

```
Session session = connection.createSession(
    false, Session.AUTO_ACKNOWLEDGE);
```

Schnippsel 8

```
InitialContext jndi = null;
Connection connection = null;
```

Schnippsel 9

```
// Zugriff auf eine Point-To-Point Warteschlange
Queue exampleQueue = (Queue) jndi.lookup("/queue/ExampleQueue");
MessageProducer exampleProducer = session.createProducer(exampleQueue);
```

Schnippsel 10

```
} finally {
    // Ressourcen freigeben
    if (jndi != null) {
        try { jndi.close(); }
        catch (NamingException ex) { ex.printStackTrace(); }
    }

    if (connection != null) {
        try { connection.close(); }
        catch (JMSException ex) { ex.printStackTrace(); }
    }
}
```

Schnippsel 11

```
connection = connectionFactory.createConnection();
```

Schnippsel 12

```
} catch (JMSException ex) {
    ex.printStackTrace();
    System.err.println("JMS-FEHLER!");
```

Schnippsel 13

```
// Zugriff auf eine themenorientierte Warteschlange
Topic topic = (Topic) jndi.lookup("/topic/ExampleTopic");
MessageProducer topicProducer = session.createProducer(topic);
```

Schnippsel 14

```
} catch (NamingException ex) {
    ex.printStackTrace();
    System.err.println("JNDI-FEHLER!");
```

## Aufgabe 5 (EJB-Typen) (T)

Beschreiben Sie die verschiedenen Beantypen von EJBs. Stellen Sie dabei die wichtigsten Eigenschaften der Beans heraus und nennen Sie die Annotationen, mit welchen ihre Klassen annotiert werden müssen.

- Stateless Session Bean
- Stateful Session Bean
- Message Driven Bean
- Persistent Entities

## Aufgabe 6 (Enterprise Java Beans) (P)

Oh, je! Hier hat sich der Fehlerteufel eingeschlichen und unsere schönen Klassen kaputt gemacht. So war das nicht geplant. Eigentlich sollte aus der Klasse `ConcertSearchBean` eine Stateless Session Bean und aus der Klasse `TicketOrderBean` eine Stateful Session Bean werden. Außerdem sollte die Methode `sendConfirmation()` der `TicketOrderBean` asynchron aufgerufen werden, weil die Anwendung nicht darauf warten muss, bis die Bestätigungsemail verschickt wurde.
Können Sie aushelfen? Streichen Sie alle falschen Quellcodeteile durch und ergänzen Sie was fehlt.

### Interface ConcertSearch:

```
import java.rmi.Remote;
import java.util.List;
import javax.ejb.*;


public interface ConcertSearch extends Remote {
    public List<Concert> searchByMusician(String musician);
    public List<Concert> searchByMusicianAndPlace(String musician, String place);
}
```

### Klasse ConcertSearchBean:

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;
import javax.ejb.*;


public interface ConcertSearch extends UnicastRemoteObject implements ConcertSearch {
    public List<Concert> searchByMusician(String musician) throws RemoteException {
        // ...
    }

    public List<Concert> searchByMusicianAndPlace(String musician, String place) {
        // ...
    }
}
```

### Interface TicketOrder:

```
import javax.ejb.*;


public interface TicketOrder {
    public void setAddress(Address address);
    public void order(Concert concert, int amount) throws FunkyEjbException;
    public void sendConfirmation(String email);
}
```

**Klasse TicketOrderBean:**

```
@Local
public Class TicketOrderBean {
    private Address address;
    private Order order;
    @EntityManager PersistenceContext em;

    public void setAddress() {
        // ...
    }

    public void order(Concert concert, int amount) throws RemoteError {
        // ...
    }


    public void sendConfirmation(String email, String message) {
        // ...
    }
}
```

**Hinweis:** Es befinden sich mindestens 14 Fehler in den Quellcodes.

## Aufgabe 7 (Java Persistence API) (P)

Ergänzen Sie die folgenden Klassen so mit den Annotationen der Java Persistence API, dass sie alle in unterschiedlichen Tabellen einer Datenbank abgelegt werden. Außerdem sollen alle Dantesätze einen eindeutigen Schlüssel besitzen und zwischen Kunden und Proukten eine bidirektionale n:m-Beziehung bestehen.

```
public class Person {
    private int id;
    private String name;
    private String addresse;

    public int getId() { ... }
    public void setId(int id) { ... }
    ...
}

public class Kunde extends Person {
    private List<Produkt> gekaufteProdukte;
}
```

```
public class Mitarbeiter extends Person {
    ...
}

public class Produkt {
    private int id;
    private List<Kunde> kunden;
    ...
}
```

**Herzlichen Glückwunsch!**

Sie haben das letzte Aufgabenblatt geschafft.
Und nun viel Erfolg bei der Klausur!