Distributed Systems

Exam Preparation

Exercise 1: Explain what properties a system has to have to fulfil the definition of a distributed system.

1. Consists of differnt autonomous systems

2. They communiate by exchanging messages over the network

3. They appear as a single large system to the user

Exercise 2: Explain the following terms:

a) Access Transparency

Access transparency makes it possible to use the same operations to access local and remote resources. File distribution must be hidden from the clients.

b) Location Transparency

Location transparency enables access to resources regardless of their physical or network location.

c) Migration Transparency

Migration transparency makes it possible to move a system or resources without disrupting user or software processes.

d) Replication Transparency

Replication transparency ensures the presence of numerous instances of resources to improve reliability and performance without requiring the user to be aware of replication.

e) Concurrency Transparency

Concurrency transparency allows many processes to run in parallel and share resources without interfering with each other.

f) Failure Transparency

Fault transparency enables background fault abstraction so that users and application programs can perform tasks even when hardware and software components fail.

g) Scaling Transparency

Scaling Transparency enables systems and applications to scale without requiring changes to system architecture or application techniques.

Exercise: Name problems in distributed systems.

- Network transmission can induce errors
1. Messages can get lost or damaged
2. Messages can change their order
3. Network latency

- Subsystems can break

- No common state

- No common time

Exercise: Name and explain the goals of distributed systems.

- Resource Sharing: Ability to use any hardware, software and data anywhere in the system. Resource manager used to manage/control the resource sharing.

- Openess: How Distributed Systems can be extended and proved.

- Concurrency: Components are executed in concurrent processes. Components access and update the same resources.

- Scalability: Adaptions of Distributed System to *add more users*, *respond faster*. (Usually) done by adding more and/or faster processor

- Fault Tolerance: Hardware, Software and Networks can fail. Objective of fault tolerenace is trying avoic those failures. Can be avoided by *recovery*, *redundancy*

- Transparency

Exercise 3: Consider the following code.

```
 1 import java.net.*;
 2 import java.io.*;
 3
 4 public class Server {
 5     public static void main (String args[]) {
 6         try{
 7             System.out.println("Der Server ist gestartet");
 8             int serverPort = 7896;
 9             ServerSocket So = new ServerSocket(serverPort);
10             while(true) {
11                 Socket clientSocket = So.accept();
12                 DataOutputStream out = new DataOutputStream(
13                     clientSocket.getOutputStream()
14                 );
15                 DataInputStream in = new DataInputStream(
16                     clientSocket.getInputStream()
17                 );
18                 out.writeUTF(args[0]);
19                 String data = in.readUTF();
20                 clientSocket.close();
21             }
22         } catch(IOException e) {
23             System.out.println("Fehler :"+ e.getMessage());
24         }
25     }
26 }
```

a) What transport protocol is used by the sockets in the example code?

In the sample code TCP is used. This can use as server the socket of type `ServerSocket` which uses TCP.

b) In which line of the code a connection is established?

The connection is established in line `11`. `So.accept()` accepts incoming connections.

c) In which line of the code does this program send data?

in line `18` the server sends data by writing data in the output stream.

Exercise 4: We have a distributed system using Java RMI. The server offers to the client objects with the following interface. When invoking methods, data is exchanged between client and server. This data can contain copies as well as references to remote objects. What kind of data is exchanged in which of the following cases? Explain your answers!

```java
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Date;

public interface myServer extends Remote {
    public StringBuffer reverse(StringBuffer text) throws RemoteException;
    public myServer getNext() throws RemoteException;
    public Date getDate() throws RemoteException;
}
```

a) return value of the method `reverse`

The method returns a copy, because `StringBuffer` is a standard Java class and no standard classes implements `Remote`.

b) return value of the method `getNext`

The method returns a reference because the return is `myServer` where we implemented `Remote`.

c) return value of the method `getDate`

The method returns a copy, because `Date` is a standard Java class and no standard classes implements `Remote`.

Exercise 5: Name and describe the communication models supported by JMS. For what kind of scenarios would you use which.

JMS supports two types of communication models, *Point-To-Point* and *Publish/Subscribe.*

Point-To-Point uses queues. Exactly one process receives a message.

(The sender (producer) creates a message and sends it to a queue. Each message has only one recipient (consumer). The queue stores the message permanently until it is read/acknowledged by the recipient. The queues work according to the first-in-first-out principle (FiFo), i.e. the messages are taken out in the order in which they were added. A scenario Point-To-Point is suitable for is fax.)

Publish/Subscribe uses topics. An arbitrary number of processes can subscribe to a topic and receive it's message.

(The sender publishes a message for a particular topic. Multiple receivers can subscribe to a topic and will receive the messages. The receiver is only getting the Message while it's process is active. The receiver is only interest in message on point in time when it's send but not later. A scenario Publish/Subscribe is suitable for is a newpaper.)

Exercise 6: Describe the live-cycle of a servlet.

A servlet is created by invoking the `init()` method is called. Afterwards for each request the `service()` method is invoked. After each request the servlet is not destroyed, so the following requests work with the same object. When shutting down the servlet-container the servlet is destroyed by calling the `destroy()` method.

Exercise 7: Decide which of the following lines are correct JSP-Expressions. Explain your answer!

a) `<%= 23 %>`

It's a JSP expression as it can be converted to a string and does not contain a semicolon.

b) `<%= 2*5; %>`

It's not a JSP expression as it ends with an semicolon.

c) `<%= new String("Hallo"); %>`

It's not a JSP expression as it ends with an semicolon.

d) `<%= new String[4] %>`

It's a JSP expression as it can be converted to a string and does not contain a semicolon.

e) `<%= String s = "Hallo" %>`

It's not a JSP expression because it's a declaration.

Exercise 8: Consider the following two XML-Documents. Provide an appropriate DTD `address.dtd` so that both documents are valid.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE address SYSTEM "adresse.dtd">
<address>
<name>
<firstName>Hans</firstName>
<surname>Schmidt</surname>
</name>
<email>schmidt@world.org</email>
<tel>234-123-222</tel>
</address>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE address SYSTEM "adresse.dtd">
<address>
<name>
<firstName>Adelheit</firstName>
<surname>Braun</surname>
</name>
<tel type = "home">22-33-444</tel>
<tel type = "work">11-43-222</tel>
</address>
```

`address.dtd`

```
<!ELEMENT address (name, email, tel)>
<!ELEMENT name (firstName, surname)>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ATTLIST tel type (home|work) #IMPLIED>
```

Exercise 9: Describe the functionality of SOAP, WSDL and UDDI in the context of Web Services.

- SOAP (Simple Object Access Protocol) is a XML based message format. SOAP messages can be transmitted using HTTP, HTTP/S and SMTP.

- WSDL (Web Services Description Language) is a protocol used to describe web services.

- UDDI (Universal Description Discovery and Integration) is an XML-based standard for describing, publishing and searching web services.

Exercise 10: There exist different categories of JDBC-drivers. Name and describe two of them.

- JDBC-ODBC-Bridge: The JDBC-ODBC bridge driver uses the ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls to ODBC function calls. This is now discouraged because of the thin driver.

- Pure Java-Driver: The JDBC-calls are directly implemented in the DBMS.

Exercise 11: Which data is and which isn't written into the output stream when using serialization.

- Information about the class of the object and all-non static and non-transient attributes of the object, including attributes of the super-class are written into the output stream.

- All static attributes and attributes declared with the keyword `transient` are not written into the output stream.

Exercise 12: What is required for serialization.

The class to be serialized needs to implement the interface `Serializable`.

Exercise 13: Is serializability inheritable?

Yes, serializability is inherited so only the subclass needs to implement `Serializable`.

Exercise 14: Do the standard classes implement `Serializable`?

Yes, almost evey subclass implements `Serializable`.

Exercise 15: The following code is given.

```java
import java.io.ObjectOutputStream;
import java.io.FileOutputStream;

public class WriteObjects {
    public static void main(String[] args) {
            try {
            FileOutputStream fos = new FileOutputStream("file.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            // 3141
            // true
            // "hello, world!"
            oos.close();
        } catch(Exception e) { System.err.println(e.toString()); }
    }

}
```

a) Write the following data to the output stream in the order given: `3141`, `true`, `"hello, world!"`

`oos.writeInt(3141);`

`oos.writeBoolean(true);`

`oos.writeObject("hello, world!");`

b) In which order is data read from the output stream

The data is read in the order in which it was inserted into the output stream, so `3141`, `true`, `"hello, world!"`.

c) What are the steps involved in deserializing objects and what is their state afterwards?

First the objects get their default values assigned and afterwards the values from the datastream are set. After deserialization created object has the same state as the serialized original object (except for static and transient attributes).

Exercise 16: Which port is used when no port is specified when working with sockets?

if no port is defined, a default port is used.

Exercise 17: Which informaiton does a packet contain?

It contains a byte array (data), the length of the byte array, the address and the port of the receiver.

Exercise 18: What happens if `.close()` is used on a socket object.

When using `.close()` the used port is released

Exercise 19: What is needed by the interface when working with JVM?

All interface need to implement the interface `Remote` and all methods need to throw `RemoteException`.

Exercise 20: Name and explain the two parts of the server. What does they consist of?

A Server consists of two parts, the *servant* and the *real server*.

The servant is the implementation of the class for remote objects which implements the remote interface. The class must extend `UnicastRemoteObject`. The servant consists of a constructor without parameters for remote objects, the implementation of the methods declared in the interface and possibly additional methods.

The real server consists of maybe creation of security manager, the creation of one or more instances of remote objects (servants), the registration of at least one remote obejct with nameserver

Exercise 21: Given are the following two code blocks:

`myServer.java`

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Date;

public interface myServer extends Remote {
    public StringBuffer reverse(StringBuffer text) throws RemoteException;
    public myServer getNext() throws RemoteException;
    public Date getDate() throws RemoteException;
}
```

`myServerImpl.java`

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Date;

public class myServerImpl extends myServer
                          implements DateServer {

    public myServerImpl() throws RemoteException { }

    public StringBuffer reverse(StringBuffer text) throws RemoteException {
        return  text.reverse();
    }

    public myServer getNext() throws RemoteException {

    }

    public Date getDate() throws RemoteException {
        return new Date();
    }

    public static void main(String[] args) {
        try {
            // ...
        } catch (Exception e) { System.err.println(e.getMessage()); }
    }
}
```

a) Create an object of the myServerImpl and bind it to the nameserver with the name `myObject`.

```
myServerImpl obj = new myServerImpl();

Naming.bind("myObject", obj);
```

b) A new object `obj2` of `myServerImpl` is created, override the reference of `myObject` with the referene to `obj2`.

```
Naming.rebind("myObject", obj2);
```

c) Unbind `myObject`.

```
Naming.unbind("myObject");
```

Exercise 22: What information does the RMI client need to know about the server?

- Has to know the location of the servant

- Has to know the name of the servant

- Has to know the methods of the interface

Exercise 23: Given is the following code for the RMI client.

```
import java.rmi.Naming;
import java.util.Date;

public class myClient {
    public static void main (String[] args) throws Exception {
        try {
            // ...
        } catch (Exception e) { System.err.println(e.getMessage()); }
    }

}
```

a) Get an object reference from the nameserver for an object of `myServer`. The reference is stored under the name `myObject`. The address of the server is stored in `args[0]`.

```
myServer obj = (myServer)Naming.lookup("rmi://" + args[0] + "/myObject");
```

Exercise 24: Explain the difference between synchronous and asychronous message calls.

- Synchronous: Code execution blocks (waits) till return is received

- Asyncrhonous: Code execution is not blocked and continuous program execution. When server returns a "callback" function is executed.

Exercise 25: What exchange type is Message-Oriented-Middleware based on?

- Asychronous

Exercise 26: What is the purpose of the JMS API?

The purpose of the JMS API is to access the quering system offered by a vendor.

Exercise 27: Explain the roles in JMS.

JMS distinguishes between two roles the *JMS provider* and the *JMS client*

The JMS provider is the querying system. The JMS client is the Java application that is sending and receiving messages.

Exercise 28: Which protocol is used to establish a connection between client and provider?

- TCP

Exercise 29: Name and explain the two entry points of of JMS.

The two entry points are the *connection factory* and *destination*.

The connection factory is needed by the JMS client to establish a connection to the JMS provider. It contains methods for initiliztation of connection.

The destination represents the message storage facility of the jms provider (queue or topic). It's used by the JMS client to insert and read messages.

Exercise 30:

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

public class PTPSender {
    public static void main(String[] main) {
        Hashtable properties = new Hashtable<String, String>();
        properties.put(
            Context.INITIAL_CONTEXT_FACTORY,
            "org.apache.activemq.jndi.ActiveMQInitialContextFactory"
        );
        properties.put(
            Context.PROVIDER_URL,
            "tcp://localhost:61616"
        );

        Context context = new InitialContext(properties);
        QueueConnectionFactory connFactory =
            (QueueConnectionFactory)context.lookup("ConnectionFactory");

        QueueConnection conn = connFactory.createQueueConnection();
        QueueSession session = conn.createQueueSession(
            false, Session.AUTO_ACKNOWLEDGE
        );

        Context context = new InitialContext(properties);
        Queue q = (Queue) context.lookup("dynamicQueues/queue1");
        QueueSender sender = session.createSender(q);

        TextMessage msg = session.createTextMessage();
        msg.setText("Hello World!");
        sender.send(msg);

        session.close();
        conn.close();

    }

}
```

Exercise 31: Explain the disadvantage if the client uses a specific program instead of a web browser.

Updating system results in modifications of server and client program

Exercise 32: Explain the HTTP-methods `GET`, `POST`, `PUT`, `DELETE`.

Exercise 33: Given is the following HTML file and form.

```
<html>
    <head>
        <title>my webpage</title>
    </head>
    <body>
        <!----> ADD FORM HERE <!---->
    </body>
</html>
```

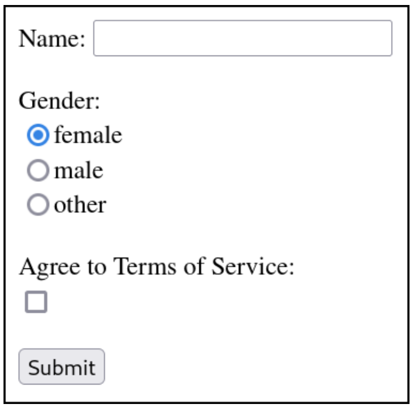


Figure 1: Form

a) Insert the code for the form shown above between the body tags. The form shows the state when the page is first accessed. The form forwards to a servlet `my-servlet` using the HTTP method `POST`.

```
<form action="my-servlet" method="POST">
    Name: <input type="text" name="name"><br/><br/>
    Gender:<br/>
    <input type="radio" name="choice_gender" CHECKED>female<br/>
    <input type="radio" name="choice_gender">male<br/>
    <input type="radio" name="choice_gender">other<br/><br/>
    Agree to Terms of Service:<br/>
    <input type="checkbox" name="tos"><br/><br/>
    <input type="submit" value="Submit">
</form>
```

Exercise 34: Explain what a servlet and a servlet-engines is.

Servlets are Java Programs that are executed within a server. It cannot run alone, instead it always needs a serverlet-engine/servlet-container to run. Servlet-engines are related to webservers and are used to invoke servlets and handover provided data by client.

Exercise 35: Explain how the interfaction between servlet and servlet-engine works.

Each servlet has signature, request and response parameter. Each servlet implements `Servlet` interface. The servlet-engine knows how to interact with the servlet.

Exercise 36: What are the three ways of creating a servlet?

1. Implement `Servlet` interface

2. Create Subclass of `GenericServlet`

3. Create Subclass of `HttpServlet`

Exercise 37: Explain what cookies are and where they are stored?

Cookies are used to store information about client. The information are stored on the client side. The purpose is to reuse information when the client sends additional requests. Cookies are stored as key-value pairs which can only a strings can be in a cookie.

Exercise 38: Name a disadvantage of cookies?

They allow storage of arbitrary information on disk of user.

Exercise 39: The following code is given:

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse res) throws Execption {
    PrintWriter out = res.getWriter();
    // ...
}
```

a) Write code for creating a cookie `c`, with the key `myCookie` and the value `0`.

```
Cookie c = new Cookie("myCookie", "0");
```

b) Write code for adding the cookie to the response.

```
res.addCookie(c);
```

c) Write code for reading and storing the cookies of a client.

```
Cookies[] reqc = req.getCookie();
```

Exercise 40: Write a XML prolog for a document of the version `3.1`, with the encoding `ISO-8859-1` and is standalone.

```
<?xml verison="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

Exercise 41: What does it mean if a XML document is well-formed?

A XML document is in line with the syntax rules of XML.

Exercise 42: What does it mean if a xml document is valid?

A XML document is valid if it's well-formed and in line with a specific schema, e.g document type definition (DTD).

Exercise 43: Create a DTD file, `solution.dtd` from the following XML document:

```
<Actor>
<aname>
<fname>Dwight</fname>
<lname>Schrute</lname>
</aname>
</Actor>
```

`solution.dtd`

```
<!ELEMENT Actor (aname)>
<!ELEMENT aname (fname, lname)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT lname (#PCDATA)>
```

Exercise 44: Which properties must be specified when realizing a service class for REST.

it must be specified what resource is addressed, what HTTP operation is realized, what representation of the resource are we using. This is done using the annotations like: `@Path("/extra")`, `@GET`, `@Produces(MediaType.TEXT_PLAIN)`

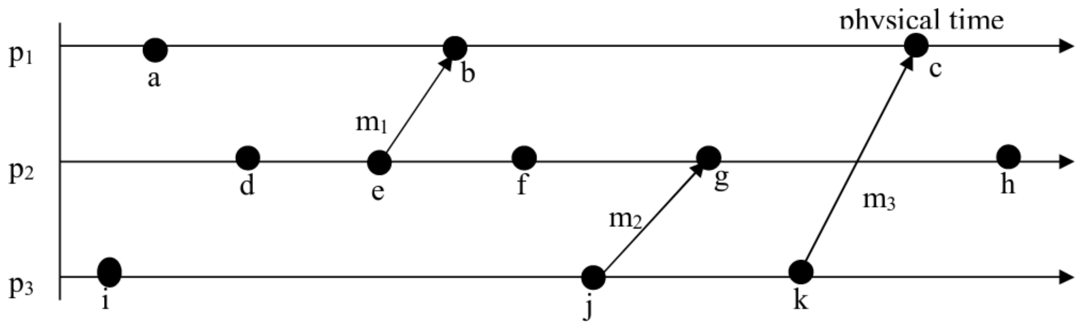
Exercise 45: Explain the three strategies to transfer classes to a database:

- Table per class hierarchy: a class and its subclasses share the same table

- Table per subclass: each class has a separate table but the table contains only the attributes that are not inherited each tuple has a reference to the base class in form of a foreign key

- Table per concrete class: each class has a separate table with all its attributes (inherited and newly defined)

Exercise 46: Assign Lamport-timestamps to the events in the following diagram.



Exercise 46: Are the two events d und k in the Happened-Before-Relation, i.e., does $d \rightarrow k$ hold? Justify your answer!

Yes $d \rightarrow k$ holds since $d \rightarrow e, e \rightarrow b, b \rightarrow c$ and $i \rightarrow j, j \rightarrow k, k \rightarrow c$ implies that d happend before c.

Exercise 47: Independent from the above diagram consider an event k that happens in physical time before another event m. Can it be it possible that the Lamport-timestamp assigned to k is larger than the Lamport-timestamp assigned to m? Explain your answer!

Yes this can happen if more events happend before k than happend before m. Since the time is defined as $t = C_i$ and C_i is a value incremented by 1 for each new event that occurs in a process. For example if 5 events occured before the occurend of k the time of k would be 6 and if 4 events happend before the occurance of event m, the time of m is only 5, regardless of the physical time.

Exercise 48: You want to synchronize your local clock with a time-server using the algorithm of Christian. Your local clock is 3 when you send a request to the time-server and 9 when you receive a response. The time value that the time-server sends you is 10. You have no further information. To what value will you set your clock when you receive the response from the time-server? Explain your answer!

$$\frac{(9-3)}{2} = 3$$

Exercise 49: The current value of your clock is 9. A synchronization algorithm provides you with the information that the correct value of your clock should be 7. What do you do to synchronize your clock?

The clock is gradually adjusted until the desired time value is reached, i.e the clock is slowed down till it's value is 7.

Client/TestServiceClient.java

```
import testws1.*;

public class TestService1Client {
    public static void main(String args[]) {
        TestService1Service service = new TestService1Service();
        TestService1 stub = service.getTestService1Port();
        System.out.println("Date: " + stub.getDate());
        System.out.println("Hello reversed: " + stub.reverse("Hello"));
    }
}
```

Server/testws1/TestServer.java

```
package testws1;
import javax.xml.ws.Endpoint;

public class TestServer {
  public static void main (String args[]) {
    TestService1 server = new TestService1();
    Endpoint endpoint =
      Endpoint.publish("http://localhost:8765/DateReverse", server);
    System.out.println("The Server is running");
  }
}
```

Server/testws1/TestService1.java

```
package testws1;

import java.util.Date;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style=Style.RPC)
public class TestService1 {
  public String getDate() {
    return new Date().toString();
  }

  public String reverse(String input){
      return new String ( new StringBuffer(input).reverse() );
  }
}
```

```java
Connection con = DriverManager.getConnection(
    "jdbc:hsqldb:file:myDBs/userDB;shutdown=true",
    "sa",""
);
PreparedStatement prepStmt =
    con.prepareStatement("INSERT INTO users VALUES (?,?)");
prepStmt.setString(1,"Meier");
prepStmt.setInt(2,27);
prepStmt.executeUpdate();
```

```java
Connection con = DriverManager.getConnection(
    "jdbc:hsqldb:file:myDBs/userDB;shutdown=true",
    "sa",""
);
PreparedStatement prepStmt = con.prepareStatement("SELECT * FROM users");
ResultSet rs = prepStmt.executeQuery(sqlQuery);

while(rs.next()) {
    String name = rs.getString("name");
    System.out.println(name);
    int zahl = rs.getInt("number");
    System.out.println(zahl);
}
```