Distributed Systems

Exam Preparation

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

Exercise 2: Explain the following terms:

a) Access Transparency

b) Location Transparency

c) Migration Transparency

d) Replication Transparency

e) Concurrency Transparency

f) Failure Transparency

g) Scaling Transparency

Exercise: Name problems in distributed systems.

Exercise: Name and explain the goals of distributed systems.

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?

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

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

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!

```
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`

b) return value of the method `getNext`

c) return value of the method `getDate`

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

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

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

a) `<%= 23 %>`

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

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

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

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

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>
```

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

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

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

Exercise 12: What is required for serialization.

Exercise 13: Is serializability inheritable?

Exercise 14: Do the standard classes implement `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!"`

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

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

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

Exercise 17: Which informaiton does a packet contain?

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

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

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

Exercise 21: Given are the following two code blocks:

`myServer.java`

```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`

```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`.

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

c) Unbind `myObject`.

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

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]`.

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

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

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

Exercise 27: Explain the roles in JMS.

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

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

Exercise 30:

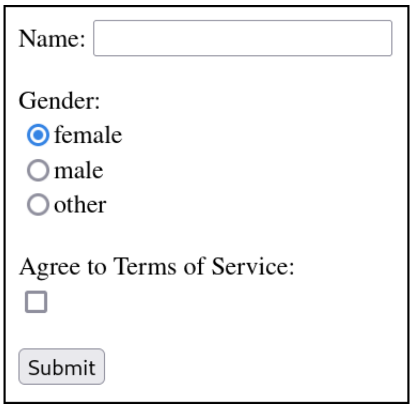
```java
import javax.jms.*;
import java.naming.*;
import java.util.*;

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.

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>
```


Name:

Gender:
- female
- male
- other

Agree to Terms of Service:

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`.

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

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

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

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

Exercise 38: Name a disadvantage of cookies?

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`.

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

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

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

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

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

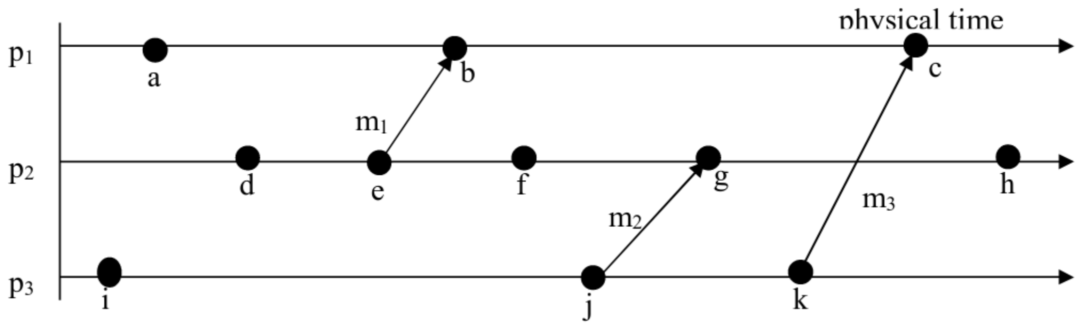
Exercise 43: Create a DTD file, `solution.dtd` from the following XML document:

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

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

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

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!

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!

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!

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?

Client/TestServiceClient.java

```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

```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

```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() );
  }
}
```

```
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();
```

```
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);
}
```