

Full name: _____

Part I: Theory Exam

Duration: 2 hours, 30 minutes
Marks 9 points (+ 1 point for the Web technologies presentation)

The use of books or notes is not permitted.

Reply to each question on a separate sheet of paper.

Exercise 1. (2 points)

(a) **(1 point)** Provide a conceptual data model of the database in Exercise 2 in the form of a UML class diagram, where your model must take into account the following considerations:

- A book is considered to be composed of copies.
- Each book has a single classification.
- The classifications form a tree structure which can be modelled via an `is_child_of` association (with appropriate roles and multiplicities). The additional constraint required to restrict the model to a tree - that there is exactly one path between any two classifications via instances of the `is_child_of` association (*you do not need to understand why this restriction is sufficient*) - can simply be added as a natural language comment.
- There are three operations corresponding to taking out, renewing and returning a copy.
- Some attributes used in the database should not appear in the conceptual model; whether or not the two count attributes are among these is debatable, here we want you to include the count attributes.
- One of the classes is best modelled as an association class.

You do not need to include any typing information.

(b) **(1 point)** Provide a UML state diagram to describe the behaviour of the `COPY` class, where your model must take into account the following considerations:

- There are three transitions, the trigger event of each one being the invocation of one of the three methods mentioned in the previous section.
- The actions to be performed concern managing the `Loan_count` of the corresponding user and managing the `Out_date` and `Return_date` of the corresponding loan.
- Your model must include the information that a user cannot take out more than 6 books and cannot renew a loan if it has already expired.

You should use the correct syntax for UML state transition labels but you are free to use any syntax for the different elements of these labels.

Exercise 2. (1,5 points)

A library database contains the following tables:

BOOKS

ISBN	Title	Author	Dewey_ref	Publisher_info	Publication_year	Copy_count

Full name: _____

CLASSIFICATIONS

Dewey_ref	Name

COPIES

Copy_ID	ISBN	Loan_type	Date_acquired

USERS

User_ID	Name	Address	Date_registered	Loan_count

LOANS

Loan_ID	Copy_ID	User_ID	Out_date	Return_date

where:

- `Copy_count` and `Loan_count` are integer attributes, recording the number of copies of a book in the library and the number of copies that a user has on loan, respectively,
- `Loan_type` is an integer attribute with value 1 or 2, indicating 3-day and 28-day loans respectively,
- `Out_date` is the date a copy is borrowed from the library and `Return_date` is the date it is returned to the library (note: NOT the day its loan expires); when a new loan is created, the value of the `Return_date` field is set to NULL.

You may assume that the function `CURRENT_DATE()` returns the current date in the same format as that used in the date fields of the `COPIES`, `USERS` and `LOANS` tables and that dates in this format can be converted to integer format using the function `TO_DAYS()` (e.g. for comparison purposes).

Now provide SQL sentences that implement the following instructions:

- (0,5 points)** List the `Title` and `Copy_ID` of all book copies that are currently overdue (that is, copies whose loan has expired but that haven't been returned to the library) together with the `Name` of the user that currently has the book on loan.
- (0,6 points)** Delete the oldest copy of the book entitled "UML2 for Dummies" from the library and then, with another SQL command, modify the `Copy_count` for this book accordingly. You may assume that the oldest copy is the one with the minimum `Copy_ID`, that this is not the only copy of the book in the library and that this copy is not currently on loan. Moreover, you do not need to worry about transactional aspects of the two commands. **Hint:** Use a nested `SELECT` containing the set function `MIN`.
- (0,4 points)** The user with `User_ID` "007" returns the copy of a book with `Copy_ID` "T42.4Me.4U". Modify the database to take this into account, using two SQL commands. You do not need to worry about transactional aspects of the two commands.

Full name: _____

Exercise 3 (2 points)

- (a) **(1 point)** Study the following code for a JSP and then answer the questions appearing below it. [Hint: RFC 2396 entitled *Uniform Resource Identifiers (URI): Generic Syntax* states that "a URI reference that does not contain a URI is a reference to the current document"].

```
1. <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
2. <%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>
3.
4. <h1>Video Rental Service Registration</h1>
5.
6. <c:if test="${param.submitted}">
7.
8.     <c:if test="${empty param.name}" var="noName" />
9.     <c:if test="${empty param.email}" var="noEmail" />
10.    <c:if test="${empty param.age}" var="noAge" />
11.
12.    <c:catch var="ageError">
13.        <fmt:parseNumber var="parsedAge" value="${param.age}" />
14.        <c:if test="${parsedAge < 16}" var="youngAge" />
15.    </c:catch>
16.    <c:if test="${not empty ageError}" var="badAge" />
17.
18.    <c:if
19.        test="${not (noName or noEmail or noAge or badAge or youngAge)}">
20.        <c:set value="${param.name}" var="name" scope="request"/>
21.        <c:set value="${param.email}" var="email" scope="request"/>
22.        <c:set value="${param.age}" var="age" scope="request"/>
23.        <jsp:forward page="registrationFormHandler.jsp" />
24.    </c:if>
25.
26. </c:if>
27.
28. <form method="post" action="">
29.     <p>
30.         Please sign up for our video rental service.
31.     </p>
32.
33.     <p>
34.         Enter your name:
35.         <input type="text" name="name"
36.             value="<c:out value="${param.name}"/>" />
37.     <br />
38.     <c:if test="${noName}">
39.         <small><font color="red">
40.             Note: you must enter a name
41.         </font></small>
42.     </c:if>
43. </p>
44.
45. <p>
```

Full name: _____

```
46. Enter your email address:
47. <input type="text" name="email"
48.     value="<c:out value="{param.email}"/>" />
49. <br />
50. <c:if test="{noEmail}">
51.     <small><font color="red">
52.         Note: you must enter an email address
53.     </font></small>
54. </c:if>
55. </p>
56.
57. <p>
58. Enter your age:
59. <input type="text" name="age" size="3"
60.     value="<c:out value="{param.age}"/>" />
61. <br />
62. <c:choose>
63.     <c:when test="{noAge}">
64.         <small><font color="red">
65.             You must enter your age
66.         </font></small>
67.     </c:when>
68.     <c:when test="{badAge}">
69.         <small><font color="red">
70.             The format of the age you entered was not correct
71.         </font></small>
72.     </c:when>
73.     <c:when test="{youngAge}">
74.         <small><font color="red">
75.             I'm sorry, you are too young to rent videos without
76.             your parents' approval
77.         </font></small>
78.     </c:when>
79. </c:choose>
80. </p>
81.
82. <input type="submit" value="Register" name="submitted"/>
83.
84. </form>
```

- (i) What is the purpose of the `submitted` request parameter
- (ii) What is the reason for defining three request-scoped variables before forwarding to another page (lines 20-22), instead of letting the destination page obtain the values directly from the corresponding parameters of the request?
- (iii) What does the code on lines 36, 48 and 60 do? What is the effect if the user did not provide a value for the parameter the previous time the form was sent?
- (iv) State briefly what the page does overall.

(b) **(1 point)** Below is a slightly edited version of the code for a servlet, contained in a file called `Controller.java`, that constitutes the central part of a Web application allowing

Full name: _____

registered users to login and then send data to be stored in the database. Study the code and then answer the questions appearing below it.

The code assumes the existence of JSP pages `login.jsp`, `dataInput.jsp` and `goodbye.jsp`. It further assumes that:

- the value of the `method` attribute of the form tag on page `login.jsp` is `POST` and that the form has two variables, called `UID` and `PWD`, and a hidden variable `referrer` with value `login`.
- the value of the `method` attribute of the form tag on page `dataInput.jsp` is `POST` and that the form has many variables, the details of which are not needed in this exercise, and a hidden variable `referrer` with value `dataInput`.

Note that the attributes `loginError` and `dataInputError` are used on the pages `login.jsp` and `dataInput.jsp` respectively, in order to know whether the user is accessing the page for the first time or whether control has been returned to the page due to an error.

```
import javax.servlet.*;
import java.sql.* ;
import javax.sql.DataSource;
import javax.naming.Context;
import javax.naming.InitialContext;
import java.io.IOException;

public class Controller extends HttpServlet {

    DataSource ds;
    HttpSession session;

    public void init() {
        /* initialize the servlet; use JNDI to look up a DataSource */
    }

    void doPost (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        session = object1.method1 ();
        session.removeAttribute("loginError");
        session.removeAttribute("dataInputError");

        String referringPage = object1.method2("referrer");

        if (referringPage != null) {

            if (referringPage.equals("login")) {
                String uid = req.getParameter("UID");
                String pwd = req.getParameter("PWD");
                if (authenticate(uid, pwd)) loginSuccess(req, res);
                else loginFailure(req, res);
            }
            else if (referringPage.equals("dataInput")) {
                String validUser = (String) object2.method1 ("uid");
                if (validUser != null) {
                    if (storeData(req)) dataInputSuccess(req, res);
                    else dataInputFailure(req, res);
                }
                else loginFailure(req, res);
            }
            /* should be an exception: unrecognised value of "referrer" variable in request */
            else loginFailure(req, res);
        }
        /* should be an exception: no "referrer" variable in the request */
        else loginFailure(req, res);
    }
}
```

Full name: _____

```
}  
  
private void gotoPage(String page, HttpServletRequest req, HttpServletResponse res) {  
    throws ServletException, IOException {  
        RequestDispatcher dispatcher = object1.method3(page);  
        if (dispatcher != null) {  
            dispatcher.forward(req, res);  
        }  
    }  
}  
  
private void loginSuccess(HttpServletRequest req, HttpServletResponse res)  
    throws ServletException, IOException {  
    object2.method2("uid", uid);  
    gotoPage("/WEB-INF/jsp/dataInput.jsp", req, res);  
}  
  
private void loginFailure(HttpServletRequest req, HttpServletResponse res)  
    throws ServletException, IOException {  
    object2.method2("loginError", "y");  
    gotoPage("/login.jsp", req, res);  
}  
  
private void dataInputSuccess(HttpServletRequest req, HttpServletResponse res)  
    throws ServletException, IOException {  
    object2.method3("uid");  
    gotoPage("/WEB-INF/jsp/goodbye.jsp", req, res);  
}  
  
private void dataInputFailure(HttpServletRequest req, HttpServletResponse res)  
    throws ServletException, IOException {  
    object2.method2("dataInputError", "y");  
    gotoPage("/WEB-INF/jsp/dataInput.jsp", req, res);  
}  
  
private boolean authenticate(String uid, String pwd) {  
    /* Connect to the database and check if the pair (uid, pwd) is registered */  
    /* - return true if this is the case (i.e the user is valid) */  
    /* - return false if it is not, or if a database connection problem occurred */  
}  
  
private boolean storeData(HttpServletRequest req) {  
    /* Connect to the database and store the input data */  
    /* - return true if data was stored OK */  
    /* - return false if a database connection problem occurred */  
}  
  
public void destroy() {}  
}
```

- (i) Explain briefly the role of the two parameters of the `doPost` and `doGet` methods in the general case, i.e. the objects of classes `HttpServletRequest` and `HttpServletResponse` (in this example they are called `req` and `res` respectively).
- (ii) Provide the two object names and six method names in the method calls that have been replaced by the character strings *objectM* for $M = 1,2$ and *methodN*, for $N = 1,2,3$.
- (iii) Judging from the information available here, does this Web application follow the MVC architectural pattern; provide a brief justification for your response.

Full name: _____

Exercise 4 (2 points)

(a) (0,5 points)

- (i) **(0,1 points)** Explain briefly what a transaction is.
- (ii) **(0,4 points)** Explain the two ways of managing transactions in EJB, identifying the differences between them.

(b) (1,5 points)

Look at the following code (conforming to the EJB2.1 specification) for the remote interfaces of a bean class and then answer the questions appearing below it. Note that certain parts of the code have been replaced with XXX.

Remote Interface

```
package examen;

public interface BookRemote extends XXX {

    public String getAuthor() throws XXX;
    public String getTitle() throws XXX;
    public String getCategory() throws XXX;
    public String getPublisher() throws XXX;
    public double getPrice() throws XXX;

}
```

Remote local interface

```
package examen;

import javax.ejb.CreateException;
import javax.ejb.DuplicateKeyException;
import javax.ejb.FinderException;

public interface BookHomeRemote extends XXX {

    public BookRemote create(String id, String title, String author,
        double price) throws XXX, DuplicateKeyException, CreateException;

    public BookRemote create(String id, String title, String author,
        String category, String publisher, double price) throws XXX,
        DuplicateKeyException, CreateException;

    public BookRemote findByPrimaryKey(String id)
        throws XXX, FinderException;

    public BookRemote findByTitle(String title)
        throws XXX, FinderException;

}
```

- (i) **(0,5 points)** Replace the parts of the code xxx by the corresponding classes.
- (ii) **(1 point)** Provide the code for the entity bean class associated with the above interfaces, taking into account that this bean uses CMP (Container-Managed Persistence).

Communication Software Replacement Exam

5º *Ingeniero de Telecomunicación*
February 4th 2008

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

Full name: _____

Note: the bean class must include both life-cycle methods (including callback methods) and business methods; there should be seven callback methods: two concerned with activation, two concerned with synchronisation with the database, 2 concerned with context management and one method for deleting the bean.

Full name: _____

Exercise 5. (1,5 puntos)

Instructions for Web technologies multiple choice questions

- There is only one correct response to each question.
- Draw a circle around the number corresponding to the correct response.
- If you wish to rectify a mistake, draw several lines through the entire column of letters next to the responses and write a new column next to this.
- 1 point will be added for a correct answer, 0.33 points will be deducted for an incorrect answer and 0 points will be given if the question is not attempted.

< Here were 15 multiple-choice questions about the content of the presentations >