PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Final Examination: Semester 2 **Academic Year**: 2004-2005

Date: 4th March, 2005 **Time**: 9:00 – 12:00 **(3 hours)**

Subject Number: 240-527 **Room**: R 300

Subject Title: Java Technology and Applications

Lecturer: Aj. Andrew Davison

Exam Duration: 3 hours

This paper has 3 pages.

Authorized Materials:

- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

Instructions to Students:

- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, and short comments; marks will not be deducted for minor syntax errors.
- The marks for each part of a question are included in brackets (...).

Question 1 (45 marks; 45 minutes)

a) Write down a DTD for a language describing exams. The top-level element is `exams`, which is made up of 1 or more `exam` elements. An `exam` element is composed of:

- an `exam_name` element, which must be present;
- an `exam_type` element, which must have the value "open book" or "closed book";
- the `numStuds` element, the number of students taking the exam. (7)

Give an XML example which uses the `exams` DTD. (3)

b) Briefly explain the *SAX approach to parsing XML documents*. Do **not** include program fragments. (10)

c) Implement a SAX parser called `ExamsSummary`. It should read in an XML file containing exams information (using the DTD of part (a)), and print out a list of all the exam names, and the total number of students in all the exams. (25)

Your code should use the JAXP classes for creating a SAX parser. Your code does not need to include any error-handling methods.

Question 2 (30 marks; 30 minutes)

a) What is a *JSP*? Your explanation should include a diagram showing how a JSP is compiled, loaded, and executed. Do **not** include code examples. (10)

b) Explain **four** main implicit objects available in JSP code. Include brief code fragments and diagrams in your answer. (20; 5 marks for each one)

Question 3 (35 marks; 35 minutes)

a) Explain *custom JSP tags*. You should explain the advantages of custom tags, and explain the various files/classes needed to support a custom tag. Do **not** include code examples. (10)

b) Implement a custom tag called `lowerN`. The tag contains body text which is converted to lowercase when the tag is processed.
The tag may have an optional `num` attribute (e.g. `num="5"`). If the attribute is present, then the lowercase text is inserted into the Web page `num` times.
Implement the following:

b.1) An example JSP that uses the `lowerN` tag. Also, show the way the page appears in the browser when loaded; (6)
b.2) The handler class for the tag; (14)
b.3) The `<tag>...</tag>` elements that must be added to the TLD. (5)

Question 4 (70 marks; 70 minutes)

Implement a BMP bean called `LibBook` which stores information about a book in the CoE library. The data inside the bean includes the book's title, it's ISBN number (a unique integer), and the availability of the book (whether it can be borrowed from the library or not).

The bean methods should allow the book's current availability to be retrieved and changed by a client. There should be a finder method for searching using a primary key.

Implement the following:

a) The `LibBook` Home Interface; (5)

b) The `LibBook` Remote Interface; (5)

c) The `LibBook` Bean; (45).

Code which is the same in many methods, such as catch and finally blocks, does not need to be repeatedly written down in every method. You can explain it with brief comments.
Do **not** implement the methods `ejbLoad()` and `ejbStore()`. Instead give brief comments explaining how they are used.

d) The primary key class used by `LibBook`; (10)

e) A schema for the database used by `LibBook`; (5)

Do **not** implement the client, a deployment descriptor, or `deploy.properties`.

--- End of Examination ---