

IS2103 Enterprise Systems Development Concepts

**Part 1: RMI and RMI-IIOP
Textbook Search**

Total Marks: 10

Part 1a (RMI): 4 marks

Implement the `TextbookSearch` client and server applications as described in pages 2 to 4 of this document.

Part 1b (RMI-IIOP): 6 marks

Implement the `TextbookSearch` client and server applications as described in pages 5 to 7 of this document.

Submission of Part 1a and 1b

The folders for Part 1a and 1b should include all “.java” and “.class” files and any data files that you may have. Instructions on *how* to submit the zip file and *how* the assignment will be evaluated are available at the IS2103 IVLE page under the Assessment heading.

Evaluation of Part 1a and 1b

You will be evaluated on the implementation of the clients and servers of the two applications. You should therefore test them thoroughly to ensure they work in accordance with the specification of the two applications documented below.

Part 1a: Remote Method Invocation (RMI)

Textbook Search

A TextbookSearchServer exposes the following interface to its clients:

```
package textbooksearch;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface TextbookSearchServer extends Remote {

    //input : course code (int)
    //output: boolean (true for success, false for failure)
    public boolean search(int courseCode)
        throws RemoteException;

    //input : course code (int)
    //output: Textbook for the course (String)
    public String getTextbook(int courseCode)
        throws RemoteException;

    //input : textbook (String)
    //output: Number of Available copies (int)
    public int checkCopies(String textbook)
        throws RemoteException;

}
```

You are required to develop an application that can be used for getting textbook information for a course offered by the University.

The interface defines three methods:

1. The `search()` method searches whether the course is offered by the University. If a course code is not found in the server, a false value is returned else a true value is returned. The course code of the courses that are offered is stored in “course.txt” file (See Appendix A). If a course code is found in the server, a true value is returned indicating that the course is offered by the school.
2. The `getTextbook()` method returns the title of the textbook that is used for a course with a course code. If the course code is not found in “course.txt”, a string “Course not offered” is returned. The list of textbook is stored in “textbook.txt” file (See Appendix A). If a textbook is not found for a course code, a string “Textbook not available” is returned.
3. The `checkCopies()` method returns the number of copies that are available in the library.

You are required to implement the TextbookSearchServer interface. You may implement the methods in whatever way you deem fit subject to the following conditions:

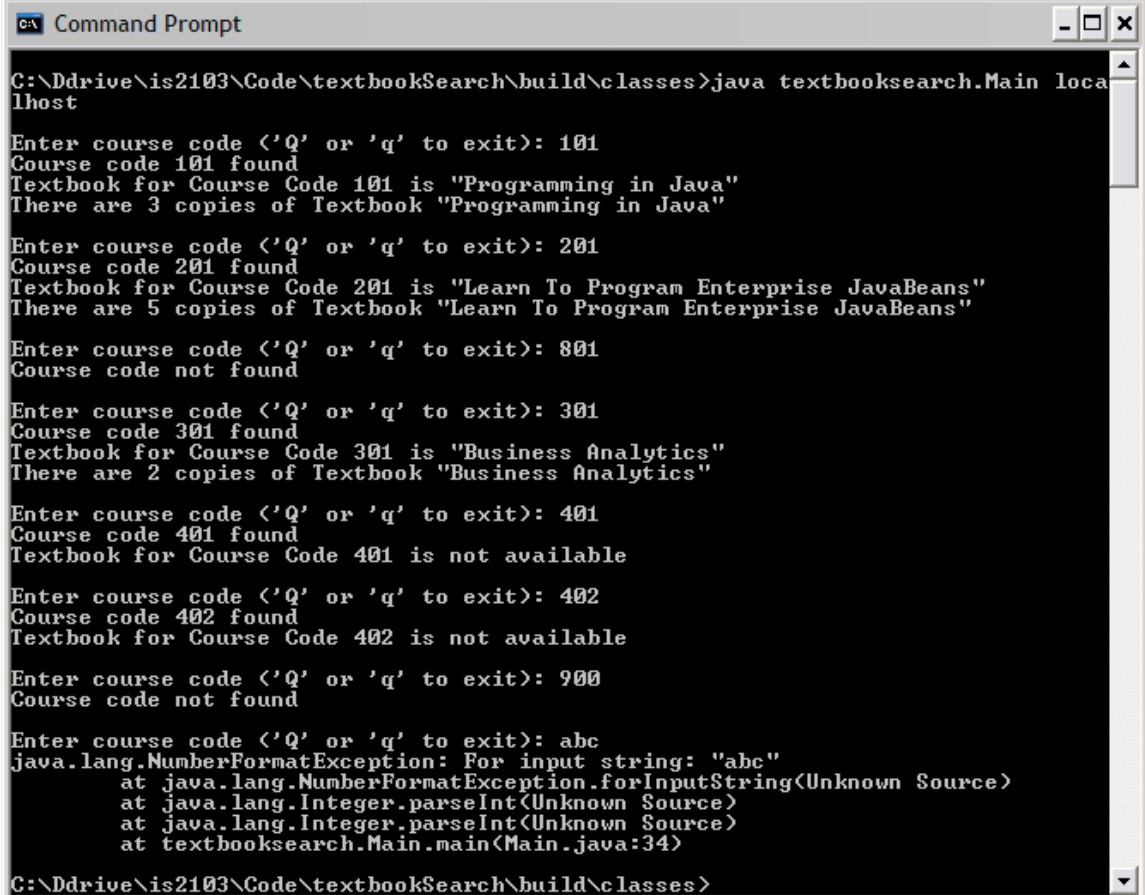
1. There are 10 courses that are been offered. You are required to use “course.txt” in the server and the order of the courses is fixed.

2. The nth textbook entry in the “textbook.txt” is used for the nth course entry in “course.txt” file. For example, “Programming in Java” is the textbook for course “101” and so on.
3. Since there are more course entries than textbook entries, the textbook for some courses (at the bottom of the list) are not available.
4. For simplicity, we assume that the number of copies for a textbook is determined by the number of words in the title of the book. For example, the number of copies for “Programming in Java” is 3.
5. The server is case-insensitive. i.e. “Programming in Java” is the same as “PrOgramMing in jAvA”.
6. Name the implementation class of the TextbookSearchServer interface as TextbookSearchServerImpl. Produce an accompanying server application class to start the server. Name the server application as TextbookSearchServerApp.

Create a client application and name it as Main. Your client should be able to demonstrate the following functionalities:

- ☐ Able to connect to the server via the RMI protocol using the default port 1099.
- ☐ It has one argument “localhost” entered at the DOS prompt. This argument denotes the server’s IP address.
- ☐ Able to repeatedly call the server’s search() method to search a course, getTextbook() method to get the textbook for a course, and checkCopies() method to check the number of available copies.
- ☐ Able to terminate the repeated call when the user is done with your client application.

The following is a series of outputs at the client side:



```
G:\Ddrive\is2103\Code\textbookSearch\build\classes>java textbooksearch.Main localhost

Enter course code (<'Q' or 'q' to exit>): 101
Course code 101 found
Textbook for Course Code 101 is "Programming in Java"
There are 3 copies of Textbook "Programming in Java"

Enter course code (<'Q' or 'q' to exit>): 201
Course code 201 found
Textbook for Course Code 201 is "Learn To Program Enterprise JavaBeans"
There are 5 copies of Textbook "Learn To Program Enterprise JavaBeans"

Enter course code (<'Q' or 'q' to exit>): 801
Course code not found

Enter course code (<'Q' or 'q' to exit>): 301
Course code 301 found
Textbook for Course Code 301 is "Business Analytics"
There are 2 copies of Textbook "Business Analytics"

Enter course code (<'Q' or 'q' to exit>): 401
Course code 401 found
Textbook for Course Code 401 is not available

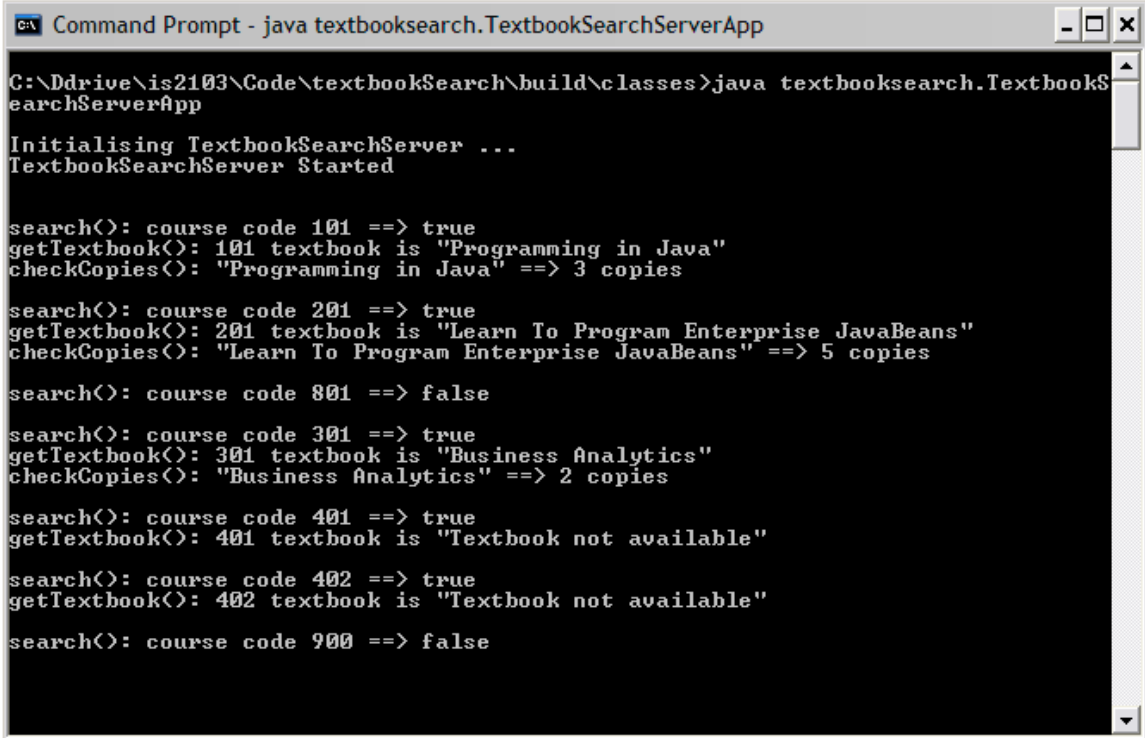
Enter course code (<'Q' or 'q' to exit>): 402
Course code 402 found
Textbook for Course Code 402 is not available

Enter course code (<'Q' or 'q' to exit>): 900
Course code not found

Enter course code (<'Q' or 'q' to exit>): abc
java.lang.NumberFormatException: For input string: "abc"
    at java.lang.NumberFormatException.forInputString(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at textbooksearch.Main.main(Main.java:34)

G:\Ddrive\is2103\Code\textbookSearch\build\classes>
```

The following is the corresponding outputs at the server side:



```
C:\ Command Prompt - java textbooksearch.TextbookSearchServerApp

C:\Ddrive\is2103\Code\textbookSearch\build\classes>java textbooksearch.TextbookSearchServerApp

Initialising TextbookSearchServer ...
TextbookSearchServer Started

search(): course code 101 ==> true
getTextbook(): 101 textbook is "Programming in Java"
checkCopies(): "Programming in Java" ==> 3 copies

search(): course code 201 ==> true
getTextbook(): 201 textbook is "Learn To Program Enterprise JavaBeans"
checkCopies(): "Learn To Program Enterprise JavaBeans" ==> 5 copies

search(): course code 801 ==> false

search(): course code 301 ==> true
getTextbook(): 301 textbook is "Business Analytics"
checkCopies(): "Business Analytics" ==> 2 copies

search(): course code 401 ==> true
getTextbook(): 401 textbook is "Textbook not available"

search(): course code 402 ==> true
getTextbook(): 402 textbook is "Textbook not available"

search(): course code 900 ==> false
```

Part 1b: RMI-IIOP

Textbook Search

A TextbookSearchServer exposes the following interface to its clients:

```
package textbooksearch;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface TextbookSearchServer extends Remote {

    //input : course code (int)
    //output: boolean (true for success, false for failure)
    public boolean search(int courseCode)
        throws RemoteException;

    //input : course code (int)
    //output: Textbook for the course (String)
    public String getTextbook(int courseCode)
        throws RemoteException;

    //input : textbook (String)
    //output: Number of Available copies (int)
    public int checkCopies(String textbook)
        throws RemoteException;

}
```

You are required to develop an application that can be used for getting textbook information for a course offered by the University.

The interface defines three methods:

1. The `search()` method searches whether the course is offered by the University. If a course code is not found in the server, a false value is returned else a true value is returned. The course code of the courses that are offered is stored in “course.txt” file (See Appendix A). If a course code is found in the server, a true value is returned indicating that the course is offered by the school.
2. The `getTextbook()` method returns the title of the textbook that is used for a course with a course code. If the course code is not found in “course.txt”, a string “Course not offered” is returned. The list of textbook is stored in “textbook.txt” file (See Appendix A). If a textbook is not found for a course code, a string “Textbook not available” is returned.
3. The `checkCopies()` method returns the number of copies that are available in the library.

You are required to implement the TextbookSearchServer interface. You may implement the methods in whatever way you deem fit subject to the following conditions:

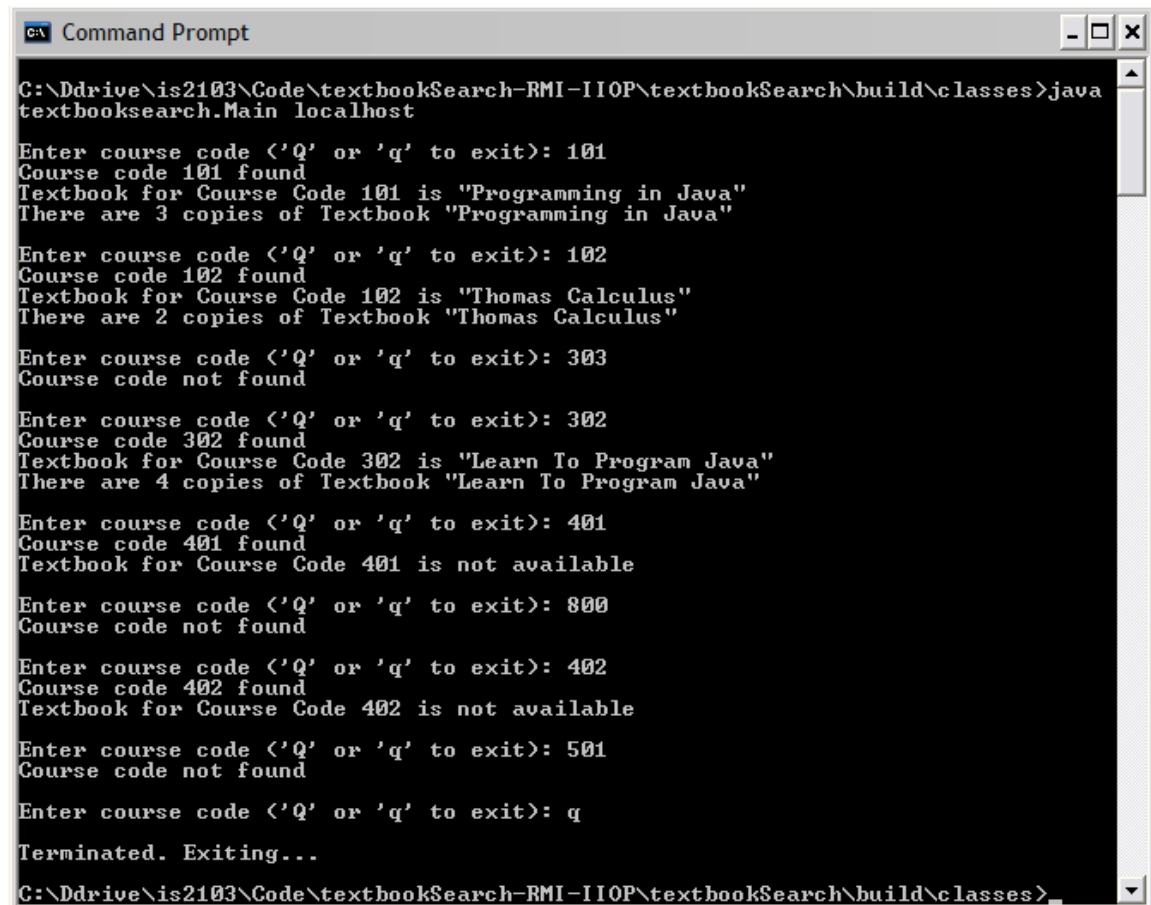
1. There are 8 courses that are been offered. You are required to use “course.txt” in the server and the order of the courses is fixed.

2. The *nth* textbook entry in the “textbook.txt” is used for the *nth* course entry in “course.txt” file. For example, “Programming in Java” is the textbook for course “101” and so on.
3. Since there are more course entries than textbook entries, the textbook for some courses (at the bottom of the list) are not available.
4. For simplicity, we assume that the number of copies for a textbook is determined by the number of words in the title of the book. For example, the number of copies for “Programming in Java” is 3.
5. The server is case-insensitive. i.e. “Programming in Java” is the same as “PrOgramMing in jAvA”.
6. Name the implementation class of the TextbookSearchServer interface as TextbookSearchServerImpl. Produce an accompanying server application class to start the server. Name the server application as TextbookSearchServerApp.

Create a client application and name it as Main. Your client should be able to demonstrate the following functionalities:

- ☐ Able to connect to the server via the RMI-IIOP protocol using the default port 900.
- ☐ It has one argument “localhost” entered at the DOS prompt. This argument denotes the server’s IP address.
- ☐ Able to repeatedly call the server’s search() method to search a course, getTextbook() method to get the textbook for a course, and checkCopies() method to check the number of available copies.
- ☐ Able to terminate the repeated call when the user is done with your client application.

The following is a series of outputs at the client side:



```
C:\Ddrive\is2103\Code\textbookSearch-RMI-IIOP\textbookSearch\build\classes>java
textbooksearch.Main localhost

Enter course code (<'Q' or 'q' to exit>): 101
Course code 101 found
Textbook for Course Code 101 is "Programming in Java"
There are 3 copies of Textbook "Programming in Java"

Enter course code (<'Q' or 'q' to exit>): 102
Course code 102 found
Textbook for Course Code 102 is "Thomas Calculus"
There are 2 copies of Textbook "Thomas Calculus"

Enter course code (<'Q' or 'q' to exit>): 303
Course code not found

Enter course code (<'Q' or 'q' to exit>): 302
Course code 302 found
Textbook for Course Code 302 is "Learn To Program Java"
There are 4 copies of Textbook "Learn To Program Java"

Enter course code (<'Q' or 'q' to exit>): 401
Course code 401 found
Textbook for Course Code 401 is not available

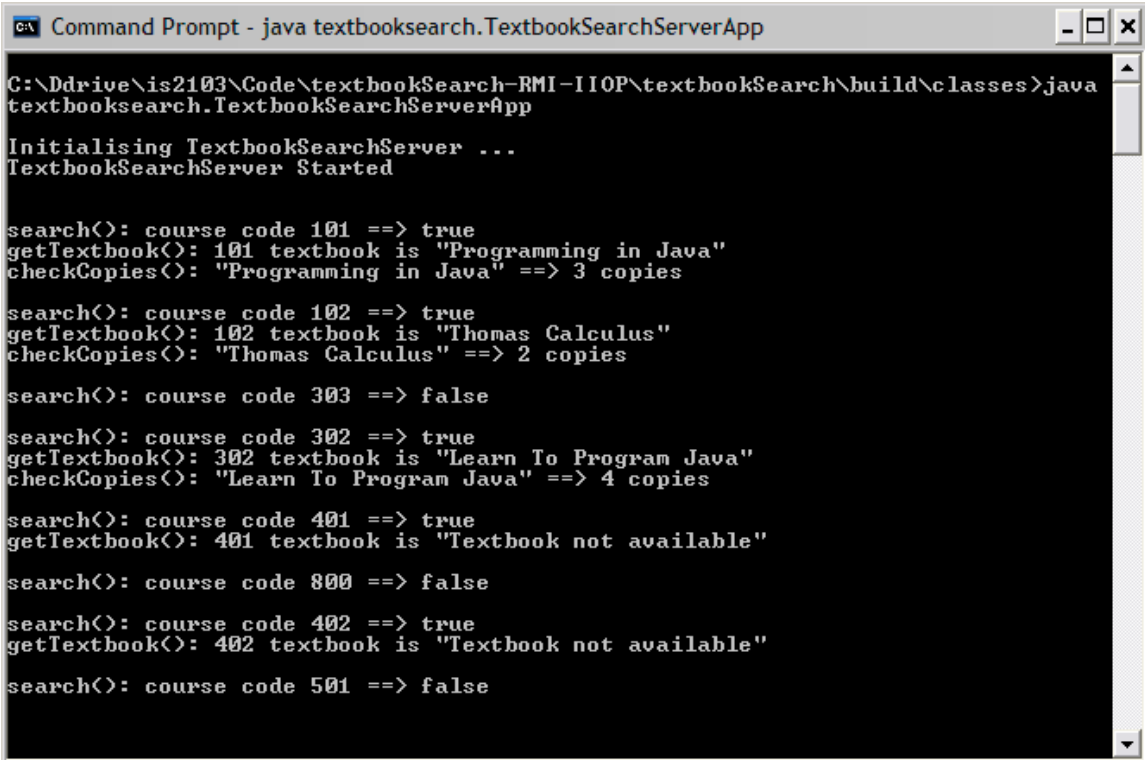
Enter course code (<'Q' or 'q' to exit>): 800
Course code not found

Enter course code (<'Q' or 'q' to exit>): 402
Course code 402 found
Textbook for Course Code 402 is not available

Enter course code (<'Q' or 'q' to exit>): 501
Course code not found

Enter course code (<'Q' or 'q' to exit>): q
Terminated. Exiting...
C:\Ddrive\is2103\Code\textbookSearch-RMI-IIOP\textbookSearch\build\classes>
```

The following is the corresponding outputs at the server side:



```
Command Prompt - java textbooksearch.TextbookSearchServerApp

C:\Ddrive\is2103\Code\textbookSearch-RMI-IIOP\textbookSearch\build\classes>java
textbooksearch.TextbookSearchServerApp

Initialising TextbookSearchServer ...
TextbookSearchServer Started

search(): course code 101 ==> true
getTextbook(): 101 textbook is "Programming in Java"
checkCopies(): "Programming in Java" ==> 3 copies

search(): course code 102 ==> true
getTextbook(): 102 textbook is "Thomas Calculus"
checkCopies(): "Thomas Calculus" ==> 2 copies

search(): course code 303 ==> false

search(): course code 302 ==> true
getTextbook(): 302 textbook is "Learn To Program Java"
checkCopies(): "Learn To Program Java" ==> 4 copies

search(): course code 401 ==> true
getTextbook(): 401 textbook is "Textbook not available"

search(): course code 800 ==> false

search(): course code 402 ==> true
getTextbook(): 402 textbook is "Textbook not available"

search(): course code 501 ==> false
```

Part 2: Enterprise JavaBeans TLS: The Library System

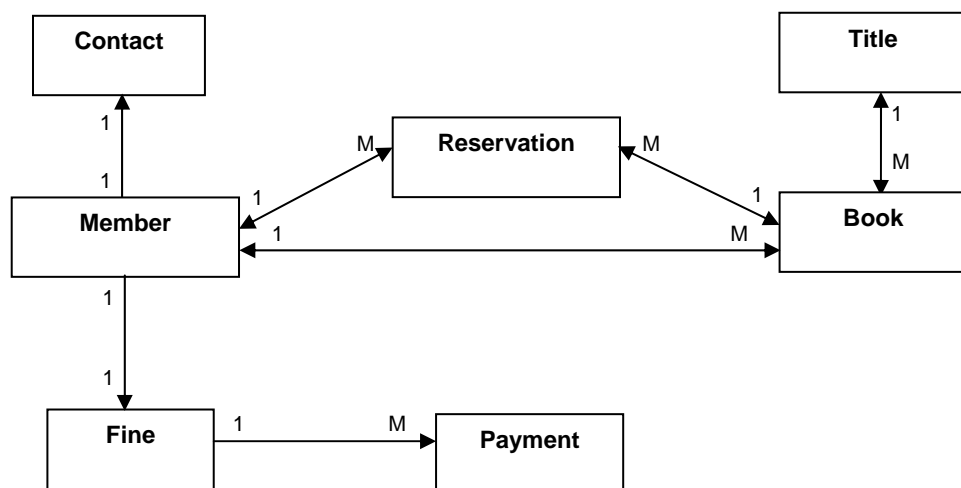
Total Marks: 30

DOMAIN INFORMATION

The Library is a general library in a university which has a student population of 3,000 students and staff. It has a collection of 1,000 printed books which students and staff can borrow and return. It has no other form of materials available for use.

Every student or staff of the university is a registered member of the library. Membership to the library is terminated (or cancelled) when a student completes his/her studies or when a staff leaves the employment of the university. No other form of membership is exercised in the Library.

From a detailed study of the business processes at the Library, an entity relationship diagram is provided:



Each library member provides to the library a contact which the library uses to get in touch with the member.

A library member is associated to a book via a loan. Each loan is for a period of 14 days. The loan period can be extended (renewed) for another period of 14 days provided it is not reserved by another member. The additional 14 days is counted from the day the book is renewed. For example, supposing a book is due for return on 3 March, renewing the book on 1 March extends the loan to 15 March and not 17 March. A library member can only renew a book once. The book must be returned when the new due date is up; otherwise, a fine will be imposed on a rate of \$1 a day for each day the book is overdue. The fine and payment is recorded in a Fine entity.

A library member can borrow up to a maximum of 6 books.

Fines are cumulative and are cleared by making one or more payments. Each payment reduces the total amount owing in the fine. The amount of fine paid is recorded in the Payment entity.

No loan is permissible if a library member has outstanding fine.

A library member can make multiple reservations for books that are currently not available and each reservation is for one book. However, a book can be reserved by more than one member. Multiple reservations on a book are queued on a first-come-first-served basis. You are not required to handle the situation when the reserved book becomes available for the members.

Each entity in the entity relationship diagram has a set of attributes (i.e. properties or data fields) that describes its static information. The list below defines the attributes. *Note that the list does not include data fields for managing entity relationships. The set of attributes is sufficient for this assignment.*

Member

- Id (unique identifier)
- Name
- Type (i.e. student or staff)
- Password¹

Contact

- Id (unique identifier)
- Department
- Faculty
- Phone number
- Email address

Book

- Id (unique identifier)
- Copy Number (e.g. 1 for the first copy, 2 for second copy, etc.)
- Loan Date (i.e. date the loan is made)
- Due Date (i.e. date the book is due for return. Renewal extends this date)

Title

- ISBN (unique identifier)
- Title of Book
- Author
- Publisher

Fine

- Id (unique identifier)
- Total Amount Owing

Payment

- Id (unique identifier)
- Date Paid
- Amount Paid

Reservation

- Id (unique identifier)
- Reservation Date (i.e. date the reservation is made)

¹ There is no requirement for password to be encrypted in this assignment

SYSTEM REQUIREMENTS

Part 2a: The Library System (for the Administrator)

1. Given the above domain information, produce an EJB 3.0 application for the Library System with a ***Windows DOS-based client administration application***.
2. The application will be called “TLS: The Library System”.
3. The entities given above are sufficient for your application. You DO NOT HAVE TO CREATE ADDITIONAL ENTITIES FOR THIS ASSIGNMENT.
4. The Windows DOS-based client application that you build will be used to test your EJB 3.0 application. For evaluation purpose, you are required to produce the following functionalities² in the DOS-based client application³:
 - a. Register a library member and his/her associated contact.
 - b. Terminate a library membership and the library member’s associated contact. A membership can only be terminated if the library member does not have any outstanding loan.
 - c. Add Titles. Allows a library administrator to add titles into the system.
 - d. Add books. Allows a library administrator to add books into the system. Each book must be associated with a title.
 - e. Delete books. Allows a library administrator to remove books from the system. The deleted book is disassociated from the title.
 - f. Facilitate loans (Borrow Books). A book on loan to a library member has a relationship link with the member.
 - g. Facilitate book returns (Return Books). A book loan will be disassociated from the library member. If the date of return is after the due date, a fine is imposed. A Fine instance is then associated with a library member.

Part 2b: The Library System (for the Member)

1. Produce a ***Web-based client application using JSP and Servlet*** for the same Library System in Part 2a.
2. The entities given above are sufficient for your application. You DO NOT HAVE TO CREATE ADDITIONAL ENTITIES FOR THIS ASSIGNMENT.
3. The Web-based client application will be used concurrently with the Windows DOS-based client application to test your EJB 3.0 application. For evaluation purpose, you are required to produce the following functionalities in the Web-based client application⁴ specially designed for the library member’s use. A library member has to log in before he can work on the functionalities:
 - a. Facilitate the viewing of current loans. A list of books on loan to the library member will be displayed. The library member may renew any loans that are not reserved by other library members. A renewal will extend the due date by 14 days. The library member can renew one or more loans on this screen.
 - b. Provide a simple search for books in the library. The library member can search for books by title or author. The books that match the search query will

² For part 2a and 2b, we will not be evaluating validation checks. That is, if an integer value is required for a data entry, we will enter the appropriate integer value.

³ How the client application execution flows is left to your own design. Bonus will be given for good user interface design.

⁴ How the web-based client application execution flows is left to your own design.

be displayed⁵ together with information of their loan status (i.e. if the books are on loan to library members, the due dates of the books). The library member can reserve one or more books on this screen.

- c. Facilitate viewing and deletion of reservations. A reservation previously made can be deleted. Display a list of reservations made by a library member. The library member can delete one or more reservations on this screen.

Implementation

1. In your implementation of the TLS and prior to the evaluation of your assignment, your system should have:
 - a. At least 5 members in your implementation.
 - b. At least 10 titles in your implementation.
 - c. At least 30 books of varying titles in your implementation.

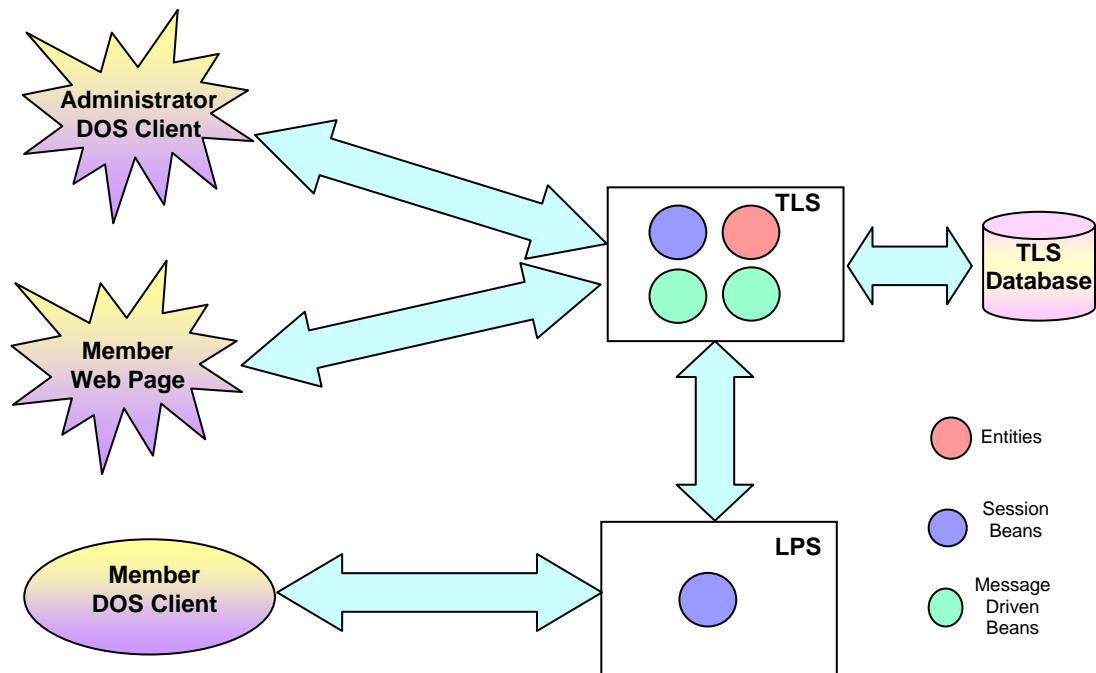
Part 2c: LPS: The Library Payment System

1. Produce another EJB 3.0 application on a **separate Netbeans project** to facilitate payment by a library member.
2. The application will be called “LPS: Library Payment System”.
3. This application will have a ***Windows DOS-based client*** to facilitate payment. No login is required.
4. The application serves as a kiosk for members to make payment for outstanding fines. It displays the amount of outstanding fine a library member has when his membership id is provided. The library member then makes partial or full payment on the fine e.g. if a member has a fine of \$10, if he pays \$6, then his outstanding fine is reduced to \$4; however, if he pays \$10 instead, the outstanding fine will be reduced to \$0. If the member pays \$12, his outstanding fine becomes -\$2 (suggesting the Library owes the member \$2). A Payment instance is created and associated with the fine at TLS.
5. LPS has a session bean that sends a message to TLS to request for information on any outstanding fine a library member may have. TLS returns a reply via a queue arranged by LPS. If there is no outstanding fine, a message will be displayed on the DOS prompt to inform the member that there is no fine to pay. Otherwise, LPS displays the amount of outstanding fine the library member has on the DOS prompt. The library member enters the payment amount and a message is sent to the TLS to update the payment and outstanding fine for the library member in TLS. TLS acknowledges the receipt of the message with a text message⁶ to LPS. Once LPS receives the acknowledgement text message, it displays on the DOS prompt “Payment made successfully” or “Error in payment process”. If the payment is successful, the latest fine amount is also displayed to inform the library member of the latest outstanding fine amount.

Pictorially, TLS and LPS communication looks like this:

⁵ Display book id, copy number, title, author, publisher, due date (if it is on loan, otherwise, blank)

⁶ The content of the message is up to you.



EVALUATION

1. There will be an evaluator assigned to assess your work during the assignment evaluation week⁷.
2. You will be required to deploy the EJB application on a computer in the laboratory or on your own laptop if you desire.
3. Tests on the functionalities of the Windows DOS-based and Web-based client applications will be carried out by the evaluator simultaneously. The evaluator will be examining the source code of the client and server logs of the EJB 3.0 application to ascertain the proper use of Enterprise JavaBeans 3.0 concepts.
4. Note that all database queries within your EJB application should be done using EJB-QL. **Marks will be deducted for the use of native SQL in database queries.**
5. Marks⁸ will be allocated in the following manner for all *executable* applications:
 - a. TLS Functionalities (up to a maximum of 22 marks)
 - TLS for Administrator (up to 12 marks)
 - TLS for Member (up to 10 marks)
 - b. LPS Functionalities (up to a maximum of 8 marks)
6. Bonus marks: based on additional features such as good and nice web page design, advanced search function, encrypted passwords, or any function that has not been specified above (up to a maximum of 3 bonus marks can be awarded).
7. If your EJB 3.0 application is unable to deploy and therefore not executable, up to 3 marks will be awarded for your Part 2 submission.
8. If your EJB 3.0 application is able to deploy and is executable, but none of the functionalities required in Part 2 works properly, up to a maximum of 5 marks will be awarded for your Part 2 submission.
9. Please be reminded that plagiarism is a serious offence. Disciplinary actions will be taken against those caught cheating.

⁷ see IVLE page for latest information.

⁸ A student can score up to a MAXIMUM of 30 marks for Part 2 (including bonus marks).

Appendix A: Course List and Textbook List

Content of “course.txt”:

101
102
103
201
202
203
301
302
401
402

Content of “textbook.txt”:

Programming in Java
Thomas Calculus
Principles of Economics
Learn To Program Enterprise JavaBeans
Essentials of Statistics
Management Information Systems
Business Analytics
Learn To Program Java