Ollscoil na hÉireann, Gaillimh
National University of Ireland, Galway

GX 1487

Semester I Examinations, 2003/2004

Exam Code(s)	4BP1
Exam(s)	Fourth Year Electronic and Computer Engineering
Module Code(s)	CT414
Module(s)	Distributed Systems
Paper No.	1
External Examiner(s)	Prof. P. Nixon
Internal Examiner(s)	Prof. G. Lyons
	Dr. D. Chambers

Instructions: Answer any 4 questions.
All questions will be marked equally.

Duration	2.5 hrs
No. of Answer Books	1
No. of Pages	4
Department(s)	Information Technology

1.a: Explain, using a suitable code example (e.g. a Shopping Cart), the operation of the Session Tracking mechanism available in the Java Servlet API. How would you support session tracking for users that access a servlet with a browser that does not support cookies, or that is set up to reject cookies?

10 MARKS

b: Describe the Delegate-Model Architecture. Using a simple Bank Account as an example show how this architecture could be implemented in a Java environment. Your design should include an AccountController to modify the state of Account objects and at least one view object to display the Account state. Use the built-in Java implementation of the Observer Design Pattern in your design.

15 MARKS

2. Using Java Remote Method Invocation, outline the design for an Internet based Expense Claim Processing System. The expense server allows the client to download an expense policy object. The policy object implements an interface that provides methods to retrieve the current expense policies e.g. how much may be claimed for mileage expenses or overnight subsistence. The policy object can also be used to validate an expense claim locally (within the client application). Once validated, the expense claim can then be submitted to the expense server for processing. The following interfaces and classes are required:

- *ExpenseServer* - this (remote) interface provides methods for downloading policy objects and submission of expense claims. Obviously expense policies can change over time, so client applications can download new policy objects as often as required.
- *ExpensePolicy* - this (serializable) interface should provide methods for the retrieval of information about the current expense policies and also for checking the validity of expense claims before submission to the server.
- *ExpenseClaim* - this (serializable) class should encapsulate a completed expense claim. It should provide accessor methods to retrieve the contents of the expense claim for validation or processing.

The design of the system should make it possible for new Policy implementation classes to be easily added to the system in the future, making the system very flexible. The design uses Java RMI and Object Serialisation to download objects that implement the *ExpensePolicy* interface i.e. these objects are passed by value from the server to the client. Validated *ExpenseClaim* objects are subsequently passed back to the server for processing. Full implementation classes are not required but the answer should include source code for the Java interfaces and the *ExpenseClaim* class outlined above. Also include the mainline server code to initialise the server and show how a simple client program might use or interact with the server.

25 MARKS

3.a: Describe briefly the typical architecture for a Distributed Operating System that uses the *processor pool model*. What types of servers are normally used to support this model? 5 MARKS

b: Describe the main differences between a *two-tier* and a *three-tier* Client-Server architecture. Include in your description the main limitations of using a *two-tier* approach and how using a *three-tier* approach can overcome some of the potential problems. 10 MARKS

c: Describe the semantics of a typical synchronous *Remote Procedure Call* operation. Based on this description, show how an RPC library might implement these semantics on top of a connection-less transport layer (like UDP). In particular, show how this supports *non-idempotent* operations. 10 MARKS

4.a: Explain briefly the functionality of the following CORBA Components:

- Proxy Objects
- Object Adapter
- CORBA Name Service
- Interface Definition Language
- Dynamic Invocation Interface 10 MARKS

b: Given the following IDL interface definition, for a Hotel Reservation System, write a simple implementation class, in Java, for *interface Hotel*:

```
// File Hotel.idl
typedef string Date;
interface Hotel
{
  readonly attribute atring name;
  readonly attribute string tel_No;
  readonly attribute string fax_No;
  readonly attribute string no_Rooms;
  boolean makeBooking(in string c_name, in short r_type, in Date when);
  float getPrice(in Date when);
  boolean checkRoomFree(in short r_type, in Date when);
};
```

Assuming that an instance of Hotel is set-up on node *geminga.nuigalway.ie* with the registered server name *Ritz*, write the client code in Java to query the Hotel on availability and price. 15 MARKS

5.a: What output does the CORBA IDL compiler typically produce? For example, if an IDL file containing an interface called *Account* is compiled using a CORBA IDL to Java mapping, what output files are produced and what are their function? 10 MARKS

b: Using the example of a simple remote library service, outline the steps required to implement this as a CORBA based application. The following minimum rules should be adhered to:

- Service definition should include interfaces for *Library, Book, User* and *Administrator*. The *Library* interface allows Users and Administrators to login to the system i.e. it returns references to objects of these types.
- Users should be able to list the available books, retrieve details on books and request them for loan. Administrators should be able to set-up / modify user accounts as well as the books available.

Include in your answer the *IDL* file definitions and the steps required to complete the application development. Also show how a simple client program might use or interact with the server. Source code for the implementation classes **is not** required. 15 MARKS

6.a: Describe briefly the main steps involved in the migration of an active process from one system to another. 5 MARKS

b: Discuss the various policies that affect the design of distributed load balancing systems. Consider the example of adding load balancing capabilities to a *Unix Shell*, which algorithm do you think would work best in this case? How would varying the load exchange period affect the results? 10 MARKS

c: Web services represent an evolution and convergence of a number of important areas of technology and business. Describe briefly these technology areas and explain how Web Services builds on previous capabilities. Include in your explanation an overview of the main enabling technologies used to provide Web Services. 10 MARKS