PAPER CODE NO.	EXAMINER : Alexei P Lisitsa
COMP514	DEPARTMENT : Computer Science Tel. No. 0151 794 67876

THE UNIVERSITY
of LIVERPOOL

MAY 2004 EXAMINATIONS

Master of Science : Year 1

DISTRIBUTED INFORMATION SYSTEMS

TIME ALLOWED : Two hours

INSTRUCTIONS TO CANDIDATES

Answer THREE out of four questions listed.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

THE UNIVERSITY
of LIVERPOOL

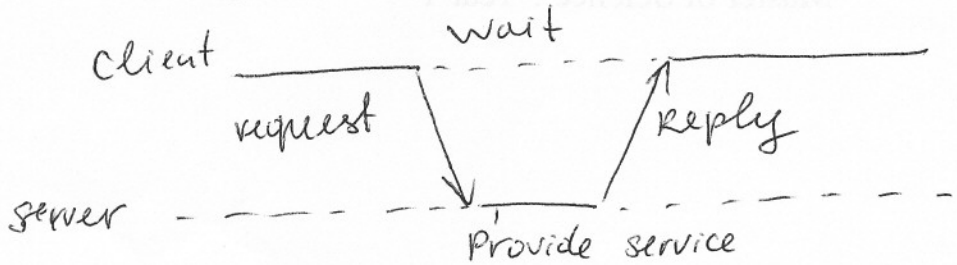
Model Solution

Answer **THREE** out of four questions listed.

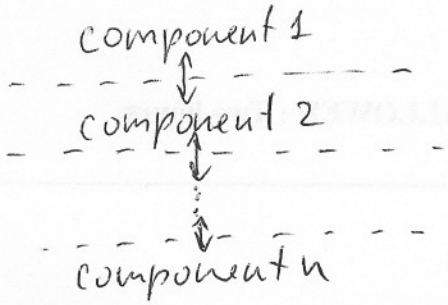
1. (a) A distributed system is a collection of autonomous hosts that are connected through a computer network. Each host executes components and operates a distribution middleware, which enables the components to coordinate their activities in such a way that users perceive the system as a single computing facility.

(b) In client-server model processes are divided into two groups:

- a server is a process implementing a specific service, e.g. a database service;
- a client is a process that requests a service from the server by sending requests and waiting for server's reply.



In a distributed system a server component can be the client component of another server component, leading to a n-tier architecture.



(c) **User-Interface level:** it is usually implemented by the client. It consists of the programs that allow the end user to interact with applications.

Processing level: middle part that contains the core functionality of an application. It might be different depending on the type of distributed system. For example, in an Internet search engine, processing level is responsible for the generation of queries to the database and ranking the links obtained from the database.

Data level: contains the programs that maintain the actual data on which the operations operate. This level is responsible of maintaining data consistent across different applications.

In two-tiered architecture these three levels can be distributed between two machines in several different ways. Most common ways are:

1. • a client machine implementing interface level;

- a server machine implementing processing and data levels.

2. • a client machine implementing interface level and part of the processing level;
   • a server machine implementing part of the processing level and data level.

(d) Essential steps in writing RMI client are as follows:

- create and install the security manager;
- construct the name of the remote object;
- use the naming.lookup method to look up the remote object with name defined in the previous step;
- invoke the remote method on the remote obect.

2. (a) Transport layers implementations are available to us in the form of sockets. A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

(b) The Socket class provides a number of constructors which enable the programmer to create a socket to a remote computer. It is usually for programming clients. The simplest constructor has two arguments:

- string (name of the computer using the DNS convention);
- port number.

Example:

```
Socket cSocket = new Socket("alexei.staff.csc.liv.ac.uk", 3333);
```

The ServerSocket class Provides a number of constructors which enable the programmer to create a socket on a server. The simplest is `ServerSoket(int port)` This creates a socket on the given port. Example: `ServerSocket sSocket = new ServerSocket(3333);`

(c) Traditional object-oriented programming focuses on the relationship between classes that are combined into one large binary executable. Component-oriented programming instead focuses on interchangeable code modules that work independently and don't require one to be familiar with their inner working to use them.

White box reuse, common in traditional object-oriented programming, assumes inheritance and specialization as main mechanisms of reuse. Existing code is reused by inheriting from an existing class and specialising its behaviour. When deriving a subclass from a base class, one must be aware of the implementation details of the base class.

Component-oriented programming promotes black box reuse, which allows one to use an existing component without caring about its internals, as long as the component complies with a predefined set of operations and interfaces.

(d) The remote interface layer is concerned with the protocol that is to be used for invoking remote methods. For example this layer might support a protocol which only results in a single object being sent messages (a unicast protocol) or it might support a protocol whereby a number of objects are sent messages (a multicast protocol).

3. (a) Distributed systems should be perceived by users and application programmers as a single system rather than as a collection of cooperating components. Transparency allows to achieve this. Transparency has different dimensions.

Access transparency: enables local and remote information objects to be accessed using identical operations.

Location transparency: enables information objects to be accessed without knowledge of their location.

Concurrency transparency: enables several processes to operate concurrently using shared information objects without interference between them.

Replication transparency: enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs.

Failure transparency: enables the concealment of faults. Allows users and applications to complete their tasks despite the failure of other components.

Migration transparency: Allows the movement of information objects within a system without affecting the operations of users or application programs.

Performance transparency: Allows the system to be reconfigured to improve performance as loads vary.

Scaling transparency: Allows the system and applications to expand in scale without change to the system structure or the application algorithms.

These represent various properties that distributed systems should have.

(b)

```
interface ComputerScienceStudent{
  readonly attribute string name;
  readonly attribute string course;
  void setStudentAverageMark(in float f);
  };
```

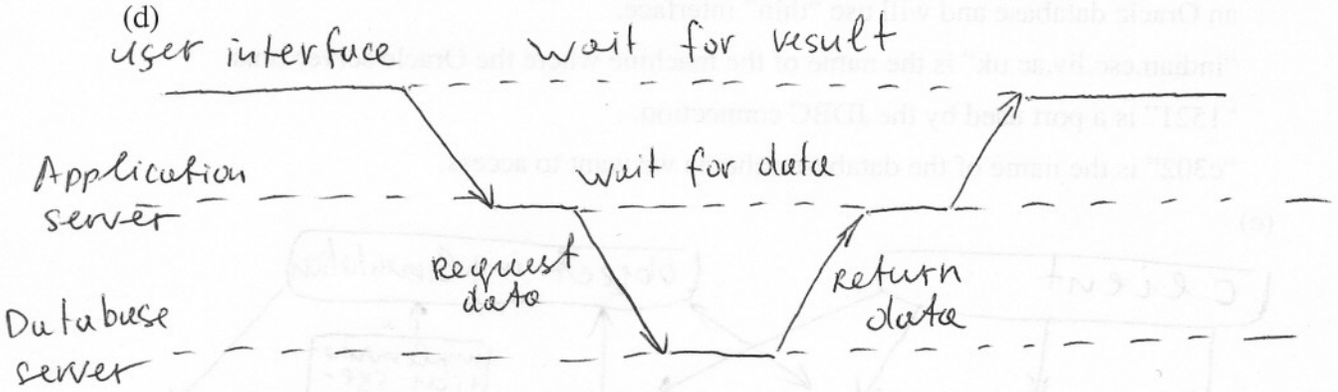
(c) The Dynamic Invocation Interface allows clients to dynamically:

- discover objects;
- discover objects' interfaces;
- create requests;
- invoke requests;
- receive responses.

Dynamic Invocation Interface dynamically issues requests to objects without requiring IDL stubs to be linked in. Clients discover interfaces at run-time and learn how to call them performing the following steps:

1. Obtain interface name
2. Obtain method description (from interface repository)
3. Create argument list
4. Create request
5. Invoke request

(d)



4. (a) **With traditional CGI scripts:**

- The programmers can choose the preferred/most appropriate programming language;

- They do not require any special extra program, only a cgi-bin directory authorized by the web administrator.

- They are very reliable methods.

But, on the other hand:

- The entire dynamic web page is created every time a form is submitted, even if the real dynamic part is usually only a (small) subset of the entire web page;

- A new process is created on the web server every time a form is submitted. This generates a big load on the web server, that is able to satisfy only few requests at a time;

- Not very flexible in terms of design and layout. All happens with print statements;

- CGI scripts usually contain both the presentation level (user interface) and the processing level in the same code. This is a disadvantage when big applications have to be developed by teams of professionals where separate team members are responsible for different parts of the Web site.

With JSP:

- Mixing of HTML and Java code;

- Instead of running a separate program that generates the entire page it uses a built-in interpreter (the JSP engine) on the HTTP server that can make the small modifications that a page needs. That is the JSP engine allows conventional HTML to pass through unchanged, and replaces the scripting information with the results of interpreting the java code.

- Easy way of separating the design and implementation of the presentation level from the design and programming of the processing level

(b) `jdbc:oracle:thin:@indian.csc.liv.ac.uk:1521:c302`

The fragment "jdbc" reflects the fact that this URL is going to be used for the connection of a Java application with a database using JDBC.

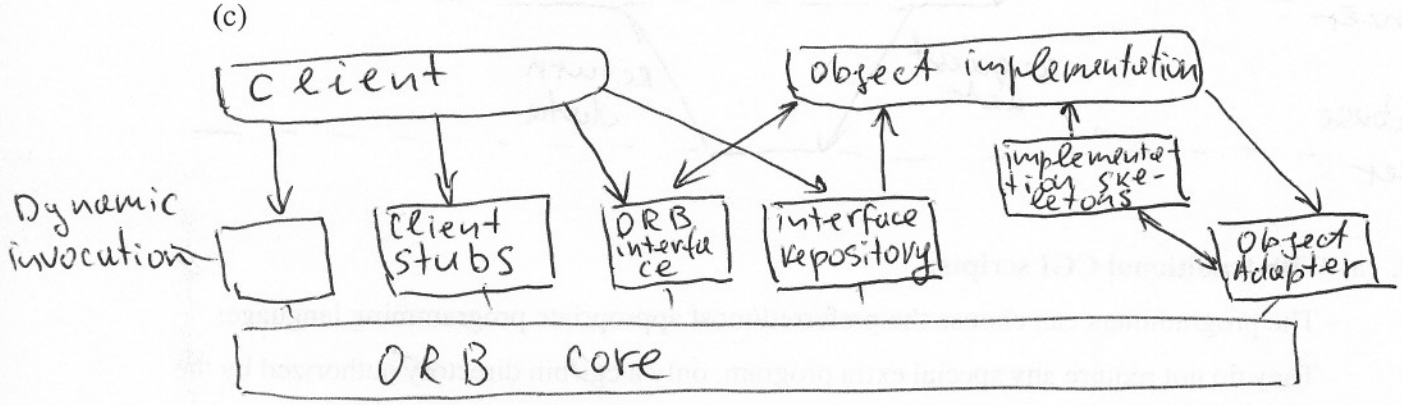
"oracle:thin" is a subprotocol part of the URL indicating that the connection will be with an Oracle database and will use "thin" interface.

"indian.csc.liv.ac.uk" is the name of the machine where the Oracle server runs.

"1521" is a port used by the JDBC connection.

"c302" is the name of the database schema we want to access.

(c)



(d) In the fragment of code the interface should be defined as public not as a private. The correct code is

```
    Import java.rmi.*;
public interface Second extends Remote
{
    long getMilliSeconds() throws RemoteException;
}
```

The interface must be declared public in order that clients can access objects which have been developed by implementing it.

All the remote interfaces inherit from the class Remote (java.rmi). Objects created using this interface will have facilities that enable it to have remote messages sent to it.

All methods that are called remotely must throw the exception RemoteException.

GetMilliseconds() is a method returning a value of the type long.