

S225/416P

**DUBLIN INSTITUTE OF TECHNOLOGY  
KEVIN STREET, DUBLIN 8.**

---

# **BSc Applied Sciences & Computing**

**Year 4**

---

**SUMMER EXAMINATIONS 2006**

---

**COMPUTER NETWORKS AND DISTRIBUTED SYSTEMS**

MR. C. O'LEARY  
DR. B. O'SHEA  
MR. S. MALONE

TIME ALLOWED: 3 ½ HOURS

ATTEMPT  
**2 QUESTIONS FROM SECTION A**  
AND  
**2 QUESTIONS FROM SECTION B**

ALL QUESTIONS CARRY EQUAL MARKS.

## Section A

1. (a) Clearly identify the various states that a thread can be in while running inside a process in a Java Virtual Machine (JVM). Explain how a thread enters and exits each state.

(6 marks)

- (b) Consider the four Java classes shown below:

```
public class Main {
    public static void main(String args[]) {
        MyData data = new MyData();
        new Thread(new Producer(data)).start();
        new Thread(new Consumer(data)).start();
        new Thread(new Consumer(data)).start();
    }
}

public class Consumer implements Runnable {
    MyData data;
    public Consumer(MyData data) { this.data = data; }
    public void run() {
        for (;;)
            System.out.println ("Consumer: " + data.load());
    }
}

public class Producer implements Runnable {
    MyData data;
    public Producer(MyData data) { this.data = data; }
    public void run() {
        for (int i = 0; ; i++) {
            data.store(i);
            System.out.println ("Producer: " + i);
        }
    }
}

class MyData {
    private int data;
    private boolean ready;
    private boolean taken;
    public MyData() {
        ready = false;
        taken = true;
    }
    public synchronized void store(int data) {
        while (!taken);
        this.data = data;
        taken = false;
        ready = true;
    }
    public synchronized int load() {
        while (!ready);
        ready = false;
        taken = true;
        return this.data;
    }
}
```

Using diagrams, describe the most significant synchronisation problem that could occur while the program above is running. Show how to modify the code to prevent this problem from arising and explain clearly why this prevents the problem from occurring.

**(9 marks)**

(c) Consider the code shown below:

```
public class Main {
    public static void main(String args[]) {
        new Main();
    }
    public Main() {
        new MyThread(this).start();
        new MyThread(this).start();
    }
    public synchronized void iWantToRun() {
    }
    public synchronized void imFinishedRunning() {
    }
}

class MyThread extends Thread {
    private Main m;
    public MyThread(Main m) {
        this.m = m;
    }
    public void run() {
        m.iWantToRun();
        for(int i = 0; i < 10; i++) {
            System.out.print(i + " ");
            yield();
        }
        m.imFinishedRunning();
    }
}
```

When this code is compiled and run, the output is as follows:

0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9

Without modifying its constructor, show how to modify the `Main` class so that the output becomes:

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

i.e. one thread is allowed to complete running before the other one starts.

Note: You should only modify the `iWantToRun` and `imFinishedRunning` methods. You may add additional state to the `Main` class if you wish. You should not modify the `MyThread` class.

**(10 marks)**

2. (a) Consider the code below:

```
import java.io.*;
import java.net.*;

public class C {

    public static void main(String[] args) {
        try {
            DatagramSocket datagramSocket =
                new DatagramSocket();
            DatagramPacket packet =
                new DatagramPacket("hello".getBytes(),
                    5, InetAddress.getLocalHost(), 12345);
            datagramSocket.send(packet);
        } catch(Exception e) {}
    }
}
```

This code sends a packet containing the string “hello” to another process. Provide the code for the process which will receive the packet and print out the string.

**(6 marks)**

- (b) Consider the code shown below:

```
import java.io.*;
import java.net.*;
public class C {
    public static void main(String[] args) {
        try {
            Socket socket =
                new Socket(
                    InetAddress.getLocalHost(), 12345);
            ObjectInputStream reader =
                new ObjectInputStream(
                    socket.getInputStream());
            System.out.println(reader.readObject());
        } catch(Exception e) {}
    }
}

class Quote {
    private static String quotes[] = {
        "I am the greatest",
        "Cogito ergo sum",
        "Well done is better than well said",
        "A problem well stated is a problem half solved"};
    private String q;
    public Quote() {
        q = quotes[(int)(Math.random() * quotes.length)];
    }
    public String toString() {
        return q;
    }
}
```

```

import java.io.*;
import java.net.*;
public class S {
    public S() {
        try {
            ServerSocket serverSocket
                = new ServerSocket(12345);
            while(true) {
                Socket socket
                    = serverSocket.accept();
                ObjectOutputStream writer
                    = new ObjectOutputStream(
                        socket.getOutputStream());
                writer.writeObject(new Quote());
            }
        } catch(IOException ioe) { }
    }
    public static void main(String[] args) {
        new S();
    }
}

```

The intended functionality for these programs working as a pair is as follows:

- S operates as a server, issuing quotes as `Quote` objects to clients who connect.
- C operates as a client, connecting to the server, downloading a `Quote` object and printing out the quote.

The program above contains one bug that will prevent this functionality from executing correctly. Identify this bug and show how to correct it.

Show how to modify the server code so that it can handle multiple client connections concurrently.

**(9 marks)**

- (c) Consider the code shown below:

```
import java.net.*;
import java.io.*;
public class S {
    public static void main(String[] args) {
        try {
            BufferedReader input =
                new BufferedReader(
                    new InputStreamReader(System.in));
            DatagramSocket socket =
                new DatagramSocket(12345);
            InetAddress addressToSendTo = // FILL THIS IN
            while(true) {
                System.out.print("Enter message :/> ");
                String message = input.readLine();
                socket.send(new DatagramPacket(
                    message.getBytes(),
                    message.length(),
                    addressToSendTo, 12345));
            }
        } catch (Exception e) {}
    }
}
```

This is a partial implementation of a server that takes a message from the user and sends it to *many* clients. Complete the implementation of this code, and write a client that can receive the message. Note that several client processes should be able to run concurrently, with *all* of them receiving the message.

**(10 marks)**

3. Consider the code, database tables and screenshots shown below. They will be needed for all parts of this question.

```
<%@ page import='java.sql.*' %>
<%
    if(request.getParameter("customer_name") == null) {
%>
<html>
    <form method="GET">
        <input type="submit" value="Enter Name">
        <br>
        <input name="customer_name" size="14">
    </form>
</html>

<%
    } else {
        try {
            // GET CUSTOMER ID FROM DATABASE
            // PUT CUSTOMER ID INTO SESSION
            response.sendRedirect("book_list.jsp");
        } catch (Exception e) {}
    }
%>
```

get\_name.jsp

```

<%@ page import='java.sql.*' %>
<%
    if(request.getParameter("selection") == null) {
%>
<html>
    <h1>Select Book</h1>
    <form method="GET">
    <table border="1">
        <tr>
            <th>Select</th>
            <th>Name</th>
            <th>Author</th>
            <th>Published</th>
        </tr>
<%
        try {
            // GET BOOK DETAILS FROM DATABASE
            while (results.next()) {
%>
                <tr>
                    <td><input type="radio" name="selection"
                        value="<%=results.getString(1) %>"></td>
                    <td><%=results.getString(2) %></td>
                    <td><%=results.getString(3) %></td>
                    <td><%=results.getString(4) %></td>
                </tr>
<%
            }
%>
                <tr>
                    <td align="center" colspan="4">Enter quantity:
                    <input type="text" size="4" name="qty"></td>
                </tr>
                <tr>
                    <td align="center" colspan="4"><input
                    type="submit" value="Place Order"></td>
                </tr>
            </table>
        </form>
<%
        } catch(Exception e) {}
    } else {
        // PUT BOOK SELECTION INTO DATABASE
        // PUT QUANTITY INTO DATABASE
        response.sendRedirect("place_order.jsp");
    }
%>

```

book\_list.jsp

```

<%@ page import='java.sql.*' %>

<html>
    <h1>Order Placed</h1>
</html>
<%
    try {
        // GET CUSTOMER ID FROM SESSION
        // GET BOOK ISBN FROM SESSION
        // GET QUANTITY FROM SESSION
        // WRITE VALUES TO DATABASE
    } catch(Exception e) {}
%>

```

place\_order.jsp

customer_id	name	address
1	Tom Smith	Ireland
2	Mary Swan	England
3	Pete Foley	Wales
4	Mike Thomas	Scotland
5	Marie Blanco	France

isbn	title	author	publisher	price
0224062247	Rough Ride	Paul Kimmage	Yellow Jersey Press	8
0385504209	The Da Vinci Code	Dan Brown	Doubleday	15
0385510438	The Last Juror	John Grisham	Doubleday	19
0451524934	1984	George Orwell	Signet	8
0743258398	One Billion Customers	James McGregor	Free Press	19

order_id	customer_id	isbn	quantity
1	1	0224062247	2
2	4	0451524934	1

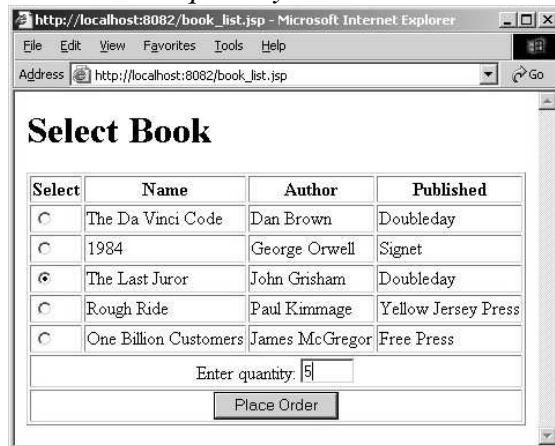
Tables in Database registered with ODBC as *BookShop*

The application works in the following way:

*Step 1: User enters their name – corresponding id looked up and saved in session*



Step 2: User selects book and quantity – isbn and quantity added to session.



Step 3: Order placed in database



- (a) Complete the code above by providing the code for the highlighted sections. You should assume that no additional JSP pages or Java Servlets are being used in this web application. You can also assume that the *customer\_id* in the *Customer* table and the *order\_id* in the *Orders* table will be automatically generated in the database.

**(15 marks)**

- (b) Using sample code and an accompanying discussion, show how you would modify *place\_order.jsp* to use an Entity Enterprise Java Bean (EJB) for entering a new *order* into the database. In your answer, you should explain clearly the role of the interfaces and also highlight the benefits of using EJB for this application.

**(10 marks)**

4. (a) Consider the code for the RMI application below.

```
import java.rmi.*;
public interface Stock extends Remote {
    public int getPrice() throws RemoteException;
    public void setPrice(int price) throws RemoteException;
}

import java.rmi.*;
import java.rmi.server.*;
public class StockImpl extends UnicastRemoteObject
    implements Stock {
    private String name;
    private int price;
    public StockImpl() throws RemoteException {}
    public void setName(String name) { this.name = name; }
    public String getName() { return name; }
    public void setPrice(int price) throws RemoteException {
        this.price = price;
    }
    public int getPrice() throws RemoteException {
        return price;
    }
}

import java.rmi.*;
public interface StockManager extends Remote {
    public void createStock(String name, int price)
        throws RemoteException;
    public Stock getStock(String name) throws RemoteException;
}

import java.rmi.*;
import java.rmi.server.*;
import java.util.Vector;
public class StockManagerImpl extends UnicastRemoteObject
    implements StockManager {
    private Vector<StockImpl> stocks = new Vector<StockImpl>();
    public StockManagerImpl() throws RemoteException {}
    public void createStock(String name, int price)
        throws RemoteException {
        StockImpl stock = new StockImpl();
        stock.setName(name);
        stock.setPrice(price);
        stocks.add(stock);
    }
    public Stock getStock(String name) throws RemoteException {
        for(int i = 0; i < stocks.size(); i++)
            if(stocks.get(i).getName().equals(name))
                return stocks.get(i);
        return null;
    }
}

import java.rmi.*;
public class S {
    public static void main(String[] args) throws Exception {
        StockManagerImpl o = new StockManagerImpl();
        String name = "rmi://localhost/StockManager";
        Naming.rebind(name, o);
    }
}
```

This application is composed of two remote classes. Instances of *StockManager* allow *Stock* objects to be created and hosted in the *S* process. Stock objects can then be queried for their *price* and can also have their price changed.

Provide the code for a client that will allow the user:

- Create a stock object by providing the *name* and *price* of the stock.
- View the *price* of a stock.
- Change the *price* of a stock for which they provide the *name*.

Note: You may use pseudo code for STDIN/STDOUT.

**(6 marks)**

- (b) Show how to modify all the code from part (a) so that
- Stock objects are not remote.
  - Stock object are created on the client side.
  - Stock objects are passed from the client to the server.
  - Stock objects are all stored by the server.
  - Stock objects can be requested by the client as before.

Briefly describe the impact this change in design has on the ability of the client to update the *price* of a stock on the server.

Note: You need only provide pseudocode once it is accompanied by a detailed explanation of the changes required.

**(9 marks)**

- (c) Design for a distributed chat application with the following functionality:
- Clients join the application by contacting a server.
  - The server selects two clients to talk to each other.
  - These clients will now chat directly with each other – i.e. their messages will *not* pass through the server.

You should provide the code for any remote interfaces in your application. You may use pseudocode to illustrate the remaining parts of your application.

**(10 marks)**

## Section B

5. (a) Explain what is meant by *garbage collection* in an object oriented system, and using a clear example, show how garbage collection is made more difficult when objects are hosted in separate processes. Describe one approach to garbage collection in distributed systems.

(6 marks)

- (b) Methods invoked on objects by other objects in the same process will not suffer from the types of problems that can occur when the *caller* and *callee* are located in separate processes on separate hosts. Middleware platforms can attempt to mask solve or mask the problems introduced by distributing objects. When doing so the platform designers define the invocation semantics for their middleware platform.

Identify the three different types of invocation semantics that can be guaranteed for a middleware platform and distinguish between each type according to the fault tolerance measures that need to be adopted to provide the guarantees.

(9 marks)

- (c) It is widely felt that *web services* represent the future for middleware and Internet-wide distributed computing. Describe clearly the core technologies of the web service approach, and provide an analysis of its strengths and weaknesses when compared with competing approaches.

(10 marks)

6. (a) DES and RSA are two well known cryptographic algorithms with diverse characteristics. Distinguish clearly between the families of algorithms to which both DES and RSA belong.

(6 marks)

- (b) The DES algorithm in its original form was famously cracked for the first time in 1997. Discuss the approach that was adopted in cracking DES on that occasion, and show how algorithm designers can protect against this type of codebreaking.

(9 marks)

- (c) Java tools such as `keytool` and `jarsigner` can be used to digitally sign documents and authenticate signatures.

Discuss the common approaches taken to perform *authentication* in distributed systems under the following headings:

- Authentication server approaches
- Authentication with public / private key pairs
- Authentication with shared secret keys

In your answer you must provide an analysis of the strengths and weaknesses of each approach as well as a clear description of the required characteristics of *secure hash algorithms*.

(10 marks)

7. (a) Describe clearly the approach adopted by Ethernet to avoid collisions between transmitted packets, and show how this is adapted for Wireless Ethernet (WiFi).  
**(6 marks)**
- (b) Describe clearly the function and operation of the Internet Protocol (IP), and show how some of its addressing and routing limitations are improved upon by the next generation of IP.  
**(9 marks)**
- (c) UDP, TCP and HTTP can all be used to transport data for higher level applications. Evaluate each protocol according to its suitability for a diverse set of applications, treating in particular each protocol's failure mechanism for handling failure.  
**(10 marks)**
8. (a) Define the term *transparency* in relation to distributed systems, and using as examples *four* different types of transparency, show how they *are* or *are not* provided for by some well known distributed applications (e.g. World-Wide-Web).  
**(6 marks)**
- (b) *Heterogeneity* and *openness* are key issues in the design and implementation of distributed systems. Provide a detailed discussion of how CORBA addresses these important issues for distributed object systems.  
**(9 marks)**
- (c) Provide a detailed analysis of the *End-to-End* argument in system design.  
**(10 marks)**