SEPTEMBER 2005 EXAMINATIONS

Bachelor of Arts : Year 2
Bachelor of Arts : Year 3
Bachelor of Engineering : Year 2
Bachelor of Science : Year 1
Bachelor of Science : Year 2

Advanced Object-Oriented Programming

TIME ALLOWED : Two hours and a half

INSTRUCTIONS TO CANDIDATES

Answer both questions.

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).
1. The practical exercises in this year’s running of the module concerned a chatroom; you were asked to develop both client-side and server-side code that allowed a user to connect to a chatroom server and then to exchange messages with other users connected to the chatroom.

The functionality of a chatroom server can be implemented using the following classes:

- **ChatRoomServer**: the top-level class that implements a server; this class sets up a Broadcaster to maintain a list of Chatter instances, then sets up a server socket that accepts incoming connections. When a connection comes in from a remote client, an instance of Chatter is created that will handle all interactions with the client. Each Chatter instance runs in its own Thread.

- **Broadcaster**: this maintains a list of Chatter instances, which represent logged-in users, and sends messages to all logged-in users. This class has a method `broadcast(String)`, which takes a string as argument, and sends the string to each Chatter instance in the list, using a method `getWriter()` in the Chatter class that returns a PrintWriter stream to send strings to the remote user.

- **Chatter**: an instance of this class represents a single logged-in user. Each time a client connects to the server, an instance of Chatter is created; this instance will handle all interactions with that user. The Chatter class implements the Runnable interface; its `run` method listens for input from the client, and uses the appropriate Broadcaster method to broadcast the messages.

(a) One way to implement the chatroom server would be to put all the required functionality in the main method, which would create the ServerSocket and Broadcaster instances, then enter a loop to keep listening for incoming connections; each time a connection is made, a Chatter instance is created to handle all communications with the newly connected user.

Write the main method for the ChatRoomServer class, and describe in words what happens when your code creates instances of the Broadcaster and Chatter classes. You need not write any error-handling code in try-catch blocks.

(Note that the ServerSocket class has a constructor `ServerSocket(int)` that creates a server socket on the given port number, and a useful method in the class is `accept()`, which returns a Socket instance when an incoming connection is detected on the server socket’s port.)  

(b) We have assumed that the chatroom server runs as a loop in the main thread that will end only when the java interpreter exits (for example, by typing control+D at the terminal). What changes would need to be made to the code you gave in your answer to part (a) to allow a user to explicitly shut down the server (for example, by clicking a ‘Shut Down’ button in a graphical user interface)?

(c) One possible problem that can arise when a program has multiple threads is data corruption through interference. How could data corruption arise in the Broadcaster class, and how could it be prevented? Give code for the broadcast method that illustrates your answer.
2. This question concerns the client-side implementation of a ‘chatroom application’ with a graphical user interface that allows a user to connect to the chatroom, to view all the messages that are sent from the chatroom server, and to send messages to the chatroom (which will then be sent, by the server, to all other users who are connected to the chatroom).

(a) The graphical user interface should allow a user to connect to the chatroom, and also allow the user to disconnect from (‘exit’) the chatroom. The user should be able to connect only if they are not currently connected; similarly, the user should be able to disconnect, or to send messages, only when they are currently connected to the chatroom. Describe how these requirements could be achieved in the implementation of the chatroom application. [10 marks]

(b) In Question 1, the functionality of the chatroom server was broken down into the classes ChatroomServer, Chatter and Broadcaster. What classes would you use to implement the chatroom application? Give descriptions of the most important methods in each class, including any event-handling methods. Give as much detail as you can in the time available; in particular, marks will be awarded for including Java code in the descriptions of the most important methods. [40 marks]