Advanced Object-Oriented Programming

TIME ALLOWED : Two hours

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 message board; you were asked to develop both client-side and server-side code that allowed a user to connect to a message-board server and then to receive messages from, and send messages to the message board.

The functionality of the Message Board server is:

- to maintain a list of current topics (e.g., the 100 most-recently updated topics),
- to accept requests for the list of current topics by sending, for each topic in the list, strings representing the title of the topic, the nickname of the initiator of the topic, etc.
- to accept a new topic: it is assumed the client-side application will send the title of the topic, the nickname of the user who is initiating the topic, and a message to start off the topic; this topic will then be given a unique identifier and added to the list of current topics;
- to accept requests for a block of messages on a given topic: it is assumed the client-side application will send: the identifier of the topic, and the number of the block being requested (assuming that messages on a topic are sent in blocks of up to twenty messages); upon receiving this request, the server will send the desired block of messages, or an error message if there is no such block;
- to accept a request to add a message to a given topic: it is assumed the client-side application will send: the identifier of the topic to which the message is to be added, and the text of the message to be added; upon receiving this request, the server will store the message, and update the information on the topic: the topic will become the most-recently updated topic.

The functionality of a message-board server could be implemented using the following classes:

- MessageBoardServer: the top-level class that implements a server; this class sets up a TopicManager to maintain the list of current topics, then sets up a Server Socket that accepts incoming connections. When a connection comes in from a remote client, an instance of SessionHandler is created which will handle all interaction with the client.

- TopicManager: this maintains a list of current topics, and provides methods for sending the list of topics, starting a new topic, etc. For example, the signature of the method that sends the list of current topics, given an output stream to the remote client, would be:

  ```java
  // send the list of topics to the PrintWriter out
  public void sendTopicList(Printwriter out);
  ```

  There would also be methods for adding a new topic, adding a message to a given topic, and sending a block of messages from a given topic.

- SessionHandler: this handles interactions with a remote client. Its constructor would take as arguments the input and output streams from the client’s connection. Its
main functionality is to read from the client’s input stream to determine, according to the application protocol, whether the client wishes to receive the current list of topics, start a new topic, etc., and call the appropriate method from the TopicManager class.

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

Write the main method for the MessageBoardServer class, and describe in words what happens when your code creates instances of the TopicManager and SessionHandler classes; in particular, you should say what arguments need to be passed to the constructors, and why. 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 message-board 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 TopicManager class, and how could it be prevented?
2. This question concerns the client-side implementation of a ‘message-board application’ with a graphical user interface that allows a user to connect to the message board, to view all the topics in the current list, start a new topic, and to view messages on, or send a new message to a given topic.

(a) The graphical user interface should allow a user to connect to the message board, and to view the current list of topics. In the practical assignments, we used a class of Topic Icons to display each topic, and the user could view the latest messages on a given topic by clicking on the icon for that topic. What other controls would be needed for the graphical user interface? Give a sketch of a reasonable layout for these controls, and briefly describe what components from the AWT or swing packages could be used to achieve your layout. [10 marks]

(b) In Question 1, the functionality of the message-board server was broken down into the classes MessageBoardServer, TopicManager and SessionHandler. What classes would you use to implement the message-board client 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 or pseudocode in the descriptions of the most important methods in each class. [40 marks]