Dr. Wolfgang Emmerich
Dept. of Computer Science
University College London
Pearson Building Room 402

13-Dec-99

C340 Concurrency
Compulsory Coursework (3 of 3)

To be handed in to G14
Deadline: Monday January 10, 2000, 12:00am

Object-oriented middleware, such as CORBA, COM and Java/RMI, support the distributed invocation of operations. The primitive form is a synchronous operation invocation between one client and one server object. CORBA also supports multiple requests where a client can call multiple operations from different server objects at once and obtain the results in the order in which they become available. This primitive supports the parallel execution of several requests and minimizes the time clients have to wait for results. Multiple requests, however, are not available as a primitive for Java/RMI but can be implemented using multi-threading.

Consider the following application scenario for multiple requests: An applet is needed that can compute the revenue that is generated from a portfolio of securities, each of which is managed by a different bank. The portfolio contains different share, bond and option objects, each of which has an operation to return the revenue that the object generated in the last period. The revenue of the portfolio is the sum of the revenue of the objects contained in the portfolio.

Design a UML class diagram and implement a Java Applet that uses a concurrent architecture primitive that we have discussed in the lectures to implement a multiple request for the portfolio revenue computation that could be used with Java/RMI. Assume that the different share, bond and option objects are executed in different threads, but do not implement the server objects as RMI servers (as we only want to study concurrency aspects of distributed systems).