Test application - absence of type compatibility verification in RMI protocol implementation:

Overview:

This RMI application consists of a server application, a client application and two remote interfaces. The server-side program generates a stub with locally instantiated remote object and binds it to the RMI registry. The client program looks up in the RMI registry to obtain the stub. However, as the client-program attempts to cast the stub using a different remote-interface, a ClassCastException would occur. This would demonstrate that although a type verification is carried out when the stub is attempted to be cast into an incompatible type, the remote stub was allowed to be looked up successfully due to absence of type verification in RMI protocol level.

Application components:

**Remote Interfaces:**

For this application, we would use the two previously use remote-interfaces, namely RemoteJobInterface and RemoteEmployeeInterface. Details of these interfaces have been given in sections 4.2.2 and 5.2.1.

**ServerSideProgram:**

A server-side program instantiates a remote Employee object, generates stub (of type RemoteEmployeeInterface) and binds it to the RMI registry as described in section 5.2.2.3. It looks like this:

Employee transportingEmployee = **new** Employee("Michael");

RemoteEmployeeInterface employeeStub = (RemoteEmployeeInterface)

UnicastRemoteObject.*exportObject*(transportingEmployee, 0);

registry.rebind("employee", employeeStub);

**ClientSideProgram:**

As described in section 5.2.3, a client-side program looks up in the RMI registry in search of the remote object reference (of type RemoteEmployeeInterface) by its name represented in String (i.e. “employee”) . Once the program obtains the stub, it then deliberately attempts to cast the stub into an incompatible remote-interface type, more precisely RemoteJobInterface type. This should result in an Exception being thrown, which is a throwable object.

Registry registry = LocateRegistry.*getRegistry*();

Throwable thrownException = (Throwable) ((RemoteJobInterface)

registry.lookup("employee"));

If this throwable object proves to be a ClassCastException, then the purpose of this demonstration is achieved.

**if** (thrownException **instanceof** ClassCastException) {

System.*out*.println("RMI call successful.");

} **else** {

System.*out*.println("RMI call unsuccessful.");

}

Application in action:

For this application, both Server and Client are run on the same machine. As the server program is run, it outputs the following on the console indicating that the service has started successfully:

Exported an employee object in RemoteEmployeeInterface wrapper...

Once the service is up and running, the client program is run and the following message is printed on the console:

RMI call successful.

This demonstrates that the casting is only in effect once the stub is obtained and attempted to be cast to an incompatible type. Therefore, this has been proved that there is no type verification in place in the RMI protocol implementation.