**Unit nameCOIT20257: Distributed Systems: Principles and Development**

INFORMATION SECURITY MANAGEMENT OF FUTUREPLUS

#### Tutor: Ling, Li-Der David

Nathanand Pinyochon (12072058)

**Question.**

Ideally, the SPMS should be a cloud-based application. This should allow the parents/students to access the SPMS from their home and teachers to set grades. The added convenience of having ubiquitous availability of access to the SPMS creates the necessity for additional security requirements. Write the principles of RMI and how this is used in client/server applications. Explore the possibility of using this in the impelmentation of SPMS. Are there any advantages?

**JavaRMI**

the Java Remote Method Invocation (Java RMI) is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage-collection.

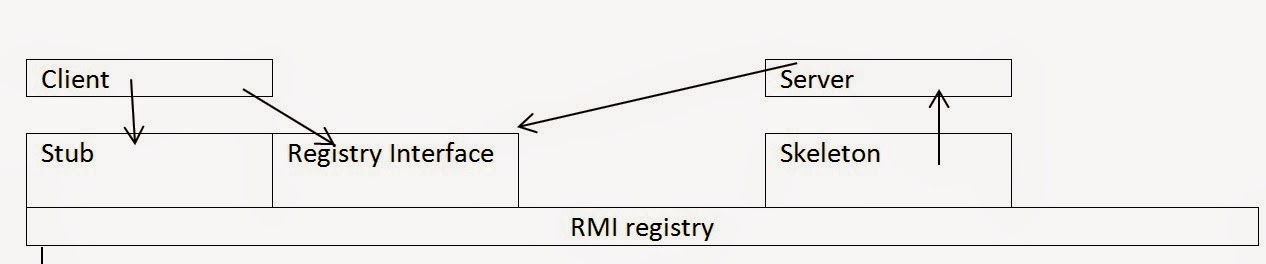
RMI is a technology developed for us to write programs to call methods from objects that are on different devices from our program. RMI will allow to call that object in the same manner as calling a method from an object that is on the same machine as our program, RMI prevents us from having to interfere with the basics of networking. For example, sockets do not need to interfere with data conversion in order to be in a form that can be sent over the network. No need to bother with having to write programs to find and run methods from ourselves on network objects. 

Figure 1.

From the Figure 1, let's explain roughly as follows: Client is a program that calls an object on another machine. Server (Server) is the object that wants to call Stub (Stub) is the part that represents the object that client will call, e.g. The client calls the object on the server side via the stub. Skeleton is the server's client. Skeleton will be the object that calls the object on the server side. To be summarized simply, the client calls the desired object via the Stub. Stub will contact Skeleton to call the object.

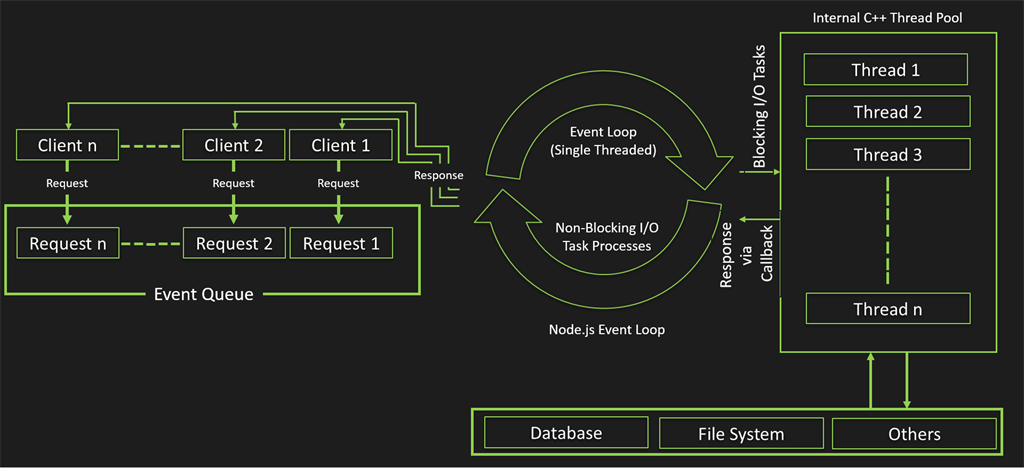
From the picture, let's explain roughly as follows: Client is a program that calls an object on another machine. Server (Server) is the object that wants to call Stub (Stub) is the part that represents the object that we will call That is, the client calls the object on the server side via the stub. Skeleton is the server's client. Skeleton will be the object that calls the object on the server side. To be summarized simply, the client calls the desired object via the Stub. Stub will contact Skeleton to call the object.

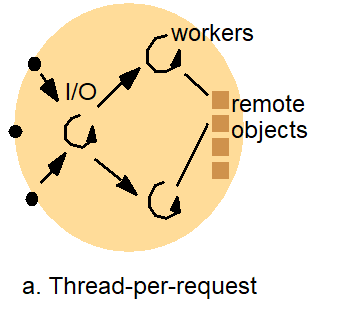
**Advantage Disadvantage**

The advantages and disadvantages are similar to those of any RPC-like (Remote Procedure Call) System. There is a superficial appearance of simplicity, because it objects which are in fact remote can be treated as though they were local.

This would seem like a great benefit to simplicity of programming, but there are hidden costs. Distributed systems have issues of latency and potential for partial failure which the programmer has to be aware of. An invocation of a remote method is subject to potential failure from security, latency problems, network failure, etc. Papering over these sorts of problems can be a disaster for reliability.

**Diagrammatic representation of the SPMS architecture.**

****



**USER INSTRUCTION**

PACKAGE CLASS NOTICE

-------------------------------------------------------------------------------------------------------------------------------

NOTICE\* 1. This project used JDK 13 to compile and Xampp server to run a localhost

2. I have use Log code to check status of server if you want to enable please go to Log class in model package and change DEBUG from “false” to “true” to show error

3. MySQL port number 3306

4. Server port 8080

5. mysql-connector-java-5.1.48.jar is a connector

Class and packages, I created 5 PACKAGES and additional class to make it easier to fix the code

Client Package

Client class: to compile client-side request to server and receiver from request

Common Package have 3 more packages

Command package for handling user input command

Command class: abstract class

GetAllStudentCommand class: extend Command class use to handle list of all students

Display Package

Converter

Model Package Java class from specification

JsonHelper class purpose of this class is convert from information to type of information for send and receive between server and client

Protocol Package

Request class: handle host port message

Response class: handle response code

Server Package

DatabaseUtility class: code to make a connection to database, create table, read from file method

FileImporter: use to import data and file

Server class: Main server class

Worker class: Use to handle thread

STEP TO RUN

1. Run SERVER main to start a sever
2. Run Client to make a request to server and respond from server.

Output:

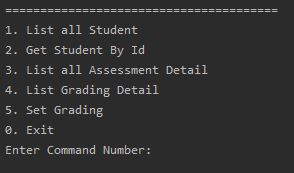


Figure. 1

List Of Assessment For Chosen Subject type 1.

Output:

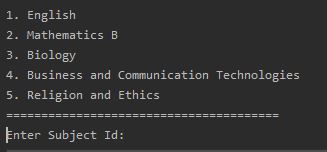


Figure. 2

Output: table (not complete)

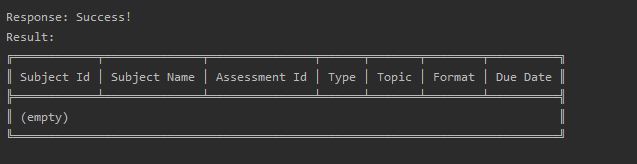
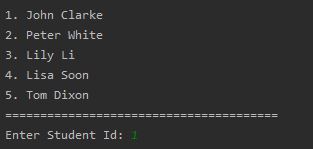


Figure. 3

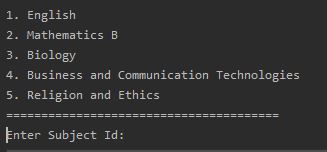
Grade Of Assessment For Student type 2

Output:

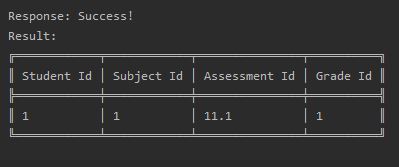


Enter subject

Output:



Output: Result

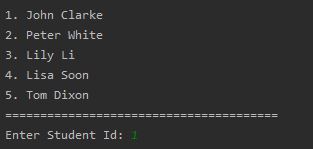


Localhost database table

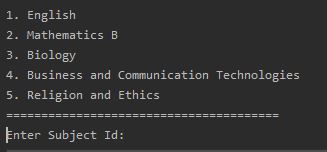


Set Grade For A Chosen Student And Subject (ADMIN) Type 3

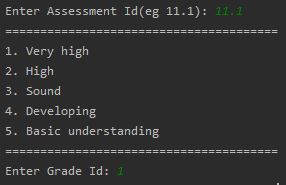
Output: enter student name



Enter Subject



Enter Assessment Id, Enter grade Id



Data will save to grade\_assessment Table in database localhost



To List grade assessment detail type 4

Output:



Figure. 4

ERD Diagram

Grade Table -> gradId = primary key

Subjects Table -> subjectId = primary key

Students Table -> studentId = primarykey

Assessment Table -> assessmentId, subject = composite primary key

Grade Assessment Table -> studentId(from Student), subjectId(from Subject), assessmentId(from Assessment), gradId(from Grade) = composite primary key