Sofia University
Department of Mathematics and Informatics

Course : Programming IT services
Date: January 16, 2015
Student Name:

Lab No. 14 (Networking and RMI)

Submit the all Java files developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

Problem 1a.

The Server – Client application developed in Lecture Networking (Fig 05- 08 attached as fig24_05_08.rar) allows to process a single client per server connection. Modify the given code fig24_05_08.rar to allow the Server application to handle concurrently multiple clients using a thread for each client i.e. **develop a multi threaded server. Test the application by starting multiple clients and demonstrate that the server can process their connections concurrently.**

Problem 1b.

Develop a multithreaded server client application to add addresses in a JFrame form at the client side. Declare an Address class to hold name, street, city, state and zip in an object. Allow the client to create Address objects and send them to the server using a button "Add". On receiving an Address object the server should serialize the objects to a file named address.ser.

Problem 2

Define a remote interface for computing **monthly payment** and **total payment**, where a client sends loan information(annual interest rate, number of years and loan amount) to the server. The server computes monthly payment and total payment and sends them (as a formatted string) back to the client.

Problem 3. (project)
(*Remote Phone Book Server*) Create a **remote phone book server** that maintains a **file of names** and **phone numbers.** Define **`interface PhoneBookServer`** with the following methods:

```
public PhoneBookEntry[] getPhoneBook()
public void addEntry( PhoneBookEntry entry )
public void modifyEntry( PhoneBookEntry entry )
```

`public void deleteEntry( PhoneBookEntry entry )`

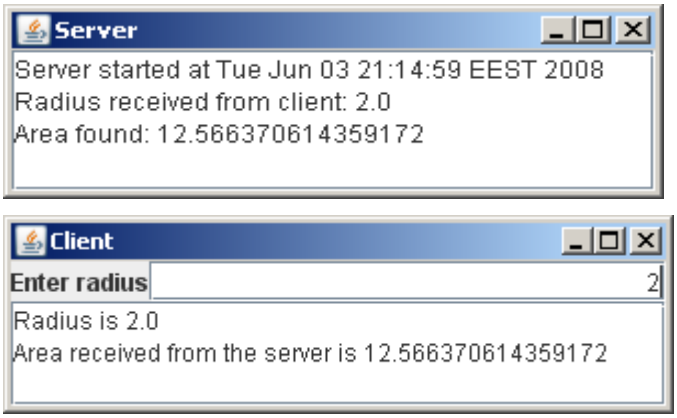
Create **`Activatable`** remote object **`class PhoneBookServerImpl`**, which implements **`interface PhoneBookServer`**. Class **`PhoneBookEntry`** should contain **`String`** **instance variables** that represent the **`first name`**, **`last name`** and **`phone number`** for one person. The class should also provide appropriate **`set/get methods`** and perform **validation** on the **phone number format**. Remember that **`class PhoneBookEntry`** also must **implement `Serializable`**, so that **`RMI`** can **serialize** objects of this class. Use a **`HashMap`** to store **`PhoneBookEntry`** objects employing their **`phone number`** as a key

Class `PhoneBookClient` should **provide a user interface** that allows the **user to scroll through entries**, add a **new entry, modify an existing entry** and **delete an existing entry**. The client and the **server should provide proper error handling** (e.g., the client cannot modify an entry that does not exist).

Problem 4a.

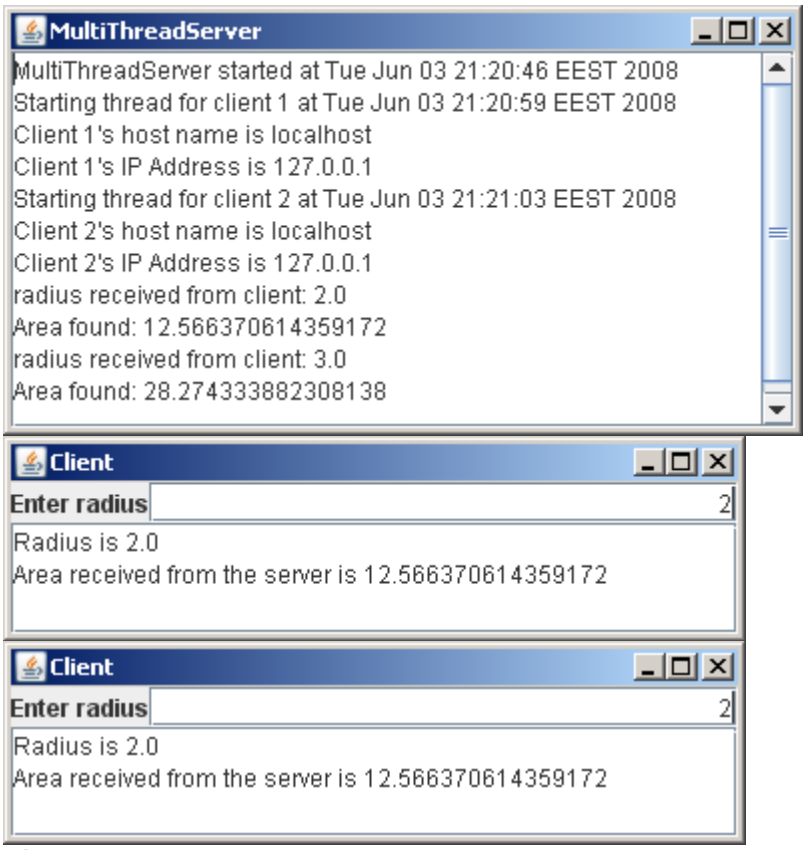
Write a server program and a client program. The client sends data to the server. The server receives the data, uses it to produce a result and then sends the result back to the client. The client displays the result on the console. In particular implement the following server client functionality:

The data sent from the client comprises the radius of a circle and the result produced by the server is the area of the circle as shown below.



Problem 4b

Write a multithread version of the server program in Problem 4a, where multiple clients can be served simultaneously by the server program as shown below:



Hint:

The server may handle simultaneously multiple connections using an endless loop as follows:

```
while(true){
      Socket socket = serverSocket.accept();
      Thread thread = new ThreadClass(socket);
      thread.start();
}
```

For each iteration of the while loop a new connection socket is created. Whenever a new connection is established, a new thread object (say, from class ThreadClass) is created to handle communication between the server and the client, which allows multiple connections to run at the same time.