Distributed Systems And Networks

Anish Katariya
ak7n14
Student ID:27561879

Index

1. Introduction:

The following project has been done as a part of the computer 2207 module coursework in which we were asked to implement a RMI framework which contain the Classes NotificationSink, NotificationSource and Notification.We were also asked to make an application that made use of the notification model made using the framework.

The Application I built is a simple banking system in which a user registers with an account name and a starting account balance. The server takes in that information and gives the customer a 10% interest on his current balance every minute and send a notification back to the client server sending it the new account balance as a notification. The client server then updates this amount on the customers GUI.

The classes that are being used to do the above tasks are BankServer, NotificationSink, NotificationSource, Notification and BankServer on the server side and ClientServer, Notification, Customer and NotificationSink on the customer side. The code to be changed/used if the program is to be run on separate machines has been indicated in the program's source code its self.

2. Framework:

According to the Specification our framework should have contain the classes Notification, NotificationSink and NotificationSource. Details of my implementation of the framework have been given below.

2.1 NotificationSink:

In my implementation I created Notification sink to be a remote interface which would be implemented by NotificationSource. The interface extends RMI Remote and also defines the basis for the methods which is used to transfer messages from the source to the client. The idea behind Implementing NotificationSink this way was to make the NotificationSource inform the NotificationSink aware when the notification had to be sent and have the NotificationSink to send it in this way.

2.2 Notification:

In my implementation I created the class Notification to be implementation specific to my application.

The RMI system allows us to send an object of any class as a distributed object. To take advantage of that face I created the Notification class that would carry the necessary information that has to be transported from the Server side to the client side. My implementation of the notification class is very basic, It contains variables for the name of the account holder and his balance and getter and setter methods for the same. Each time a interest is added the clients account the server creates a new notification indicating his new balance and sends it to the client server.

2.3 NotificationSource:

The notification source creates the heart of the whole framework. It implements the NotificationSink interface. It has methods to establish the connection with a ClientServer. It also creates methods to deal with lost connections in which it removes the client from the list of the sinked clients.It also tries to re-establish a connection and has a method which tries to resend the pending messages to the network once the connection has been reestablished.Every client that is connected is connected as a separate thread. In the getUpdate() method which is used to send the notification, the thread is put to sleep for 3600 ms or 1 minute so that the action is only performed once every minute or the balance is only updated once every minute. It then creates a new Notification Object containing the account holders name and new balance and sends it to the client Server.

3. Application:

Apart from the framework defined above the application also uses some other classes to make the banking application around the framework. The details of the classes are given below.

3.1 BankServer:

The bank server class contains the main method to create and run the server used by the RMI application.It uses rebind to create a connection between the client and the server.It also creates an object of BankServerGUI to create a simple GUI for the banks server. It contains a label indicating “connecting…” when the connection is being establish and changes the label to “bank ready…” once it has been done.

3.2 BankServerGUI:

This class lays down instructions for the compiler for the GUI of the BankServer. It has an init() method which when calls creates a GUI. It also has a setLabel() method which is used to change the value of the label from connecting.. to bank ready…

3.3 ClientServer:

This class creates a new instance of the customer. It asks the customer to register with details of his name and starting balance. It then forms a connection with the Bank Server send it its name and balance. It gets updates from the bank server every minute which the new balance. It uses ClientServerGUI to create a simple GUI to show these details to the user. It then updates this GUI every minute to show the new account balance.

3.4 ClientServerGUI:

This class lays down instructions for the compiler to create a GUI for the customer.It shows he user his account holder name and the current balance. It has an init() method that creates the GUI. It also has a setLabel() method which is used to update the balance every minute.

3.5 Customer:

This class creates an instance of the customer which has the name and the account information of every customer. An object of this class is created every time the client server is called.

4. Multiple Sources And Sinks:

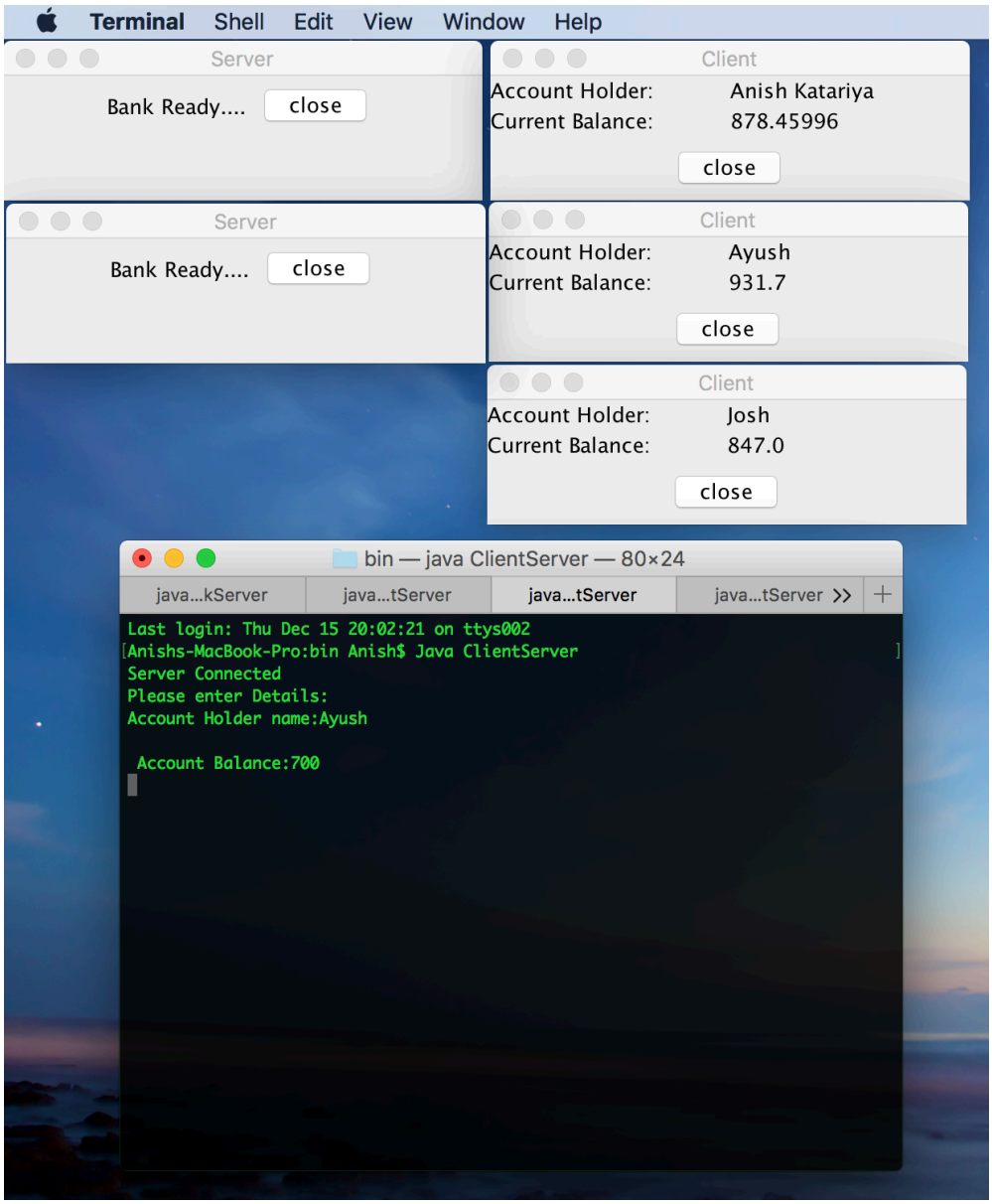

Fig: Multiple Servers And Clients Running on single Machine

As each client is connected on a separate thread in the system there can be several different sources and sinks. A source can also provide data to multiple sinks and a sink can also retrieve data from separate sources. This is ensured by the number of sinks that have been subscribed to each source and hence does not cause an error

5. Lost Connection:

The server takes each client as a thread and keeps a list of each subscriber using the synchronised method registerSink() . Incase of a lost connection the server catches the error and uses the deRegisterSink() method to remove it from the list of registered sinks and puts its pending notifications to the pending notifications list. When the sink tries to reconnect the registerSink() method also checks if the connection had previously lost the connection.If so the method reconnects and sends all the pending notifications using the sentPendingNotifications() method in the NotificationSource class.

6. Future Expansions:

As a part of the future expansions I could

Improve the GUI's for both server and client and make the server print out the number of active and lost connections. Make the Server keep a log of all notifications sent and received and the pending notifications.

I could improve the Notification class so it carries lesser amount of information and encrypt the information as it could be a security issue.

I could also add more features to the bank so as to allow it to have a feature to give people loans and have multiple accounts in the same instance of the ClientServer.