CS5204 Operating Systems

A Simple Electronic Commerce System

Using

Remote Method Invocation (RMI)

By

Li, Wnag; Wensi Xi

NOVEMBER 29, 2000

Abstracts

Electronic commerce is taking a huge stride in the Internet world and becoming very popular. The aim of the projects is to demonstrate a simple E-Commerce System, which makes use of JAVA RMI Mechanism. The system consists of four different clients, two "stores", and a "bank." The system allows the client to view and purchase items from either store. As part of the purchase order, the client provides an "account number" which the store will verify by contacting the "bank". Before authorizing the expenditure, the bank will seek confirmation from the client that the amount and the store are acceptable. The interaction between the client and the store should be asynchronous.

TABLE OF CONTENTS

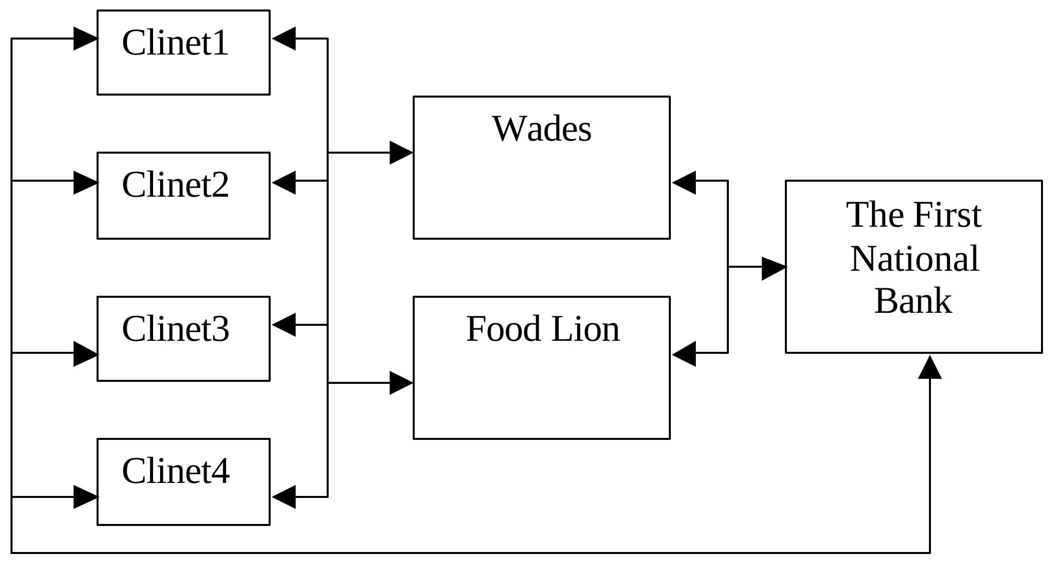
Introduction:

In this project, a well-designed simple experimental E-Commerce System had been developed using the Java **RMI** (Remote Method Invocation). Before the design and implementation of this system, an extensive reading on the JAVA **RMI** mechanism had been conducted by both of the team members. Java^{TM} Remote Method Invocation (RMI) is a distributed object model for the Java language that retains the semantics of the Java object model, making distributed objects easy to implement and to use. The Java remote method invocation system has been specifically designed to operate in the Java environment. The Java language's **RMI** system assumes the homogeneous environment of the Java Virtual Machine, and the system can therefore take advantage of the Java object model whenever possible

The simple electronic commerce system is implemented using the free Borland JAVA Builder ver. 3.5. The system consists of four different clients, two "stores", and a "bank." The system allows the client to view and purchase items from either store. As part of the purchase order, the client provides an "account number" which the store will verify by contacting the "bank". Before authorizing the expenditure, the bank will seek confirmation from the client that the amount and the store are acceptable. The interaction between the client and the store should be asynchronous. That is, the client should not block waiting for the store to reply to it requests to view or purchase items. This allows the client to conduct business with several stores simultaneously.

High Level Architecture

Below is the Marco-Scope level architecture of the project



All the communication between the classes (noted as arrows in the graphic above) will be implemented using the JAVA **RMI** mechanism. The generic architecture of the system can be described like this:

"The Client will first contact with the shop for showing items in the shop and purchasing items, on receiving purchasing requests from the Client the Shop will contact with the Bank and seek confirmation from the Bank. The bank, upon receiving a message form the Shop regarding a transaction, will first check if the purchase is valid (e.g. check Credit Card number and available credit) and contact to the client for confirmation if the purchase is proved to be valid.

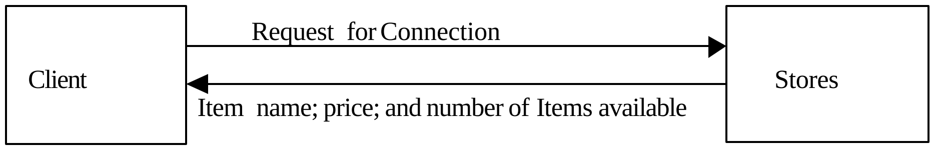
When finishing a purchasing, a Client will send a message to the shop and disconnect from that shop, and the shop will also delete necessary information it retained for that specific user."

The detailed architecture design of the Client, Store and Bank will be described in the following sections.

The Client Component

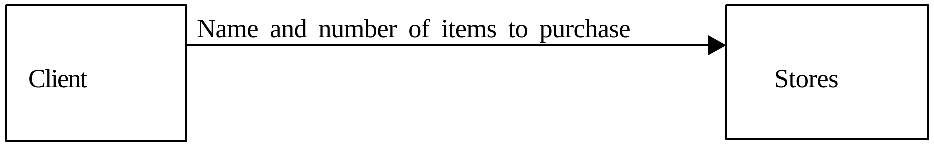
There are three cases that a Client will contact with the Shop:

1: Establish a connection with a store:



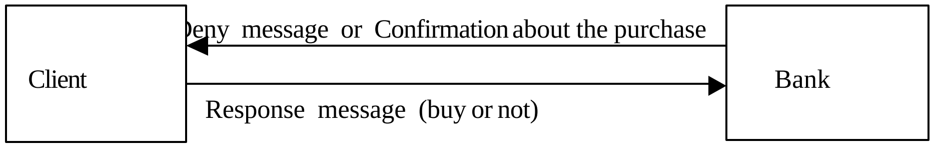
Before a Client could start a purchase, he/she must first send a request to the store for establishing a connection. Upon receiving a connection request, the shop will send the name of the items and number of each item available back to the client.

2 Made a purchase:



After the client received all the necessary information about the goods and prices of them, he/she could decide what to buy and how much to buy. The tricky point is that our interface of the client allow user to connect with the two stores simultaneously put purchased items from different stores in one single shop carter. Our Client component them different the items into two purchase lists according to the store name and send these information to the corresponding store.

3 Receiving confirmation:


Client
Deny message or Confirmation about the purchase
Response message (buy or not)
Bank

The Client will receive a message from the Bank regarding his/her purchase request, it might be a deny request, if the user name and credit card number is not valid or the total credit is not enough. It may also be a confirmation message from the Bank. The user can made final decision on whether to process the transaction or not.

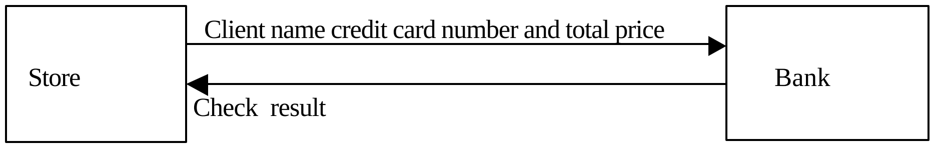
The Store Component:

Since we are using the free Borland Java Builder ver. 3.5. The Database related functions are not available. The item information is kept in the store as a hash table. The detailed functions of the Store component are described below:

1: Establish a connection with a Client:

As described earlier, upon receiving a connection request from the Client, the store will return item name and price information to the Client

2: Send confirmation message to the Bank:



After the store received the purchase request it will send the client name, total purchase price and credit card number to the Bank for confirmation. The Bank, after confirming the transaction with the Client, will send the conformation message back to the Store, and the store will perform the transaction if the transaction is finally approved by the bank.

The Bank Component:

The bank keeps the information of the Client name Credit Card number and available credit for each of the 4 clients. The detailed process of the Bank are:

1: Receiving a purchase confirmation request from the Store

As described in the previous section, when receiving a comfirmation request from the store, the bank will

1: Check if the Client's name and Credit Card number is Valid;

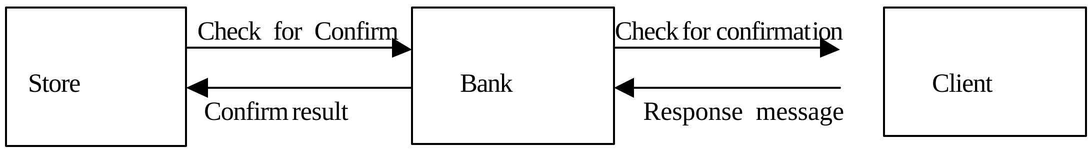
2: Check whether the total amount of the credit in the user's credit card is sufficient to perform the transaction.

3: if condition 1 and 2 are both satisfied, the bank will send confirmation message to the Client.

4: if the conditions in 1 and 2 are not both satisfied, a refuse message will send to both the Store and the Client.

2: Receiving a purchase confirmation request from the Client

When the Bank received the confirmation message from the Client, it will deduce the amount of money in the Credit Card account of the Client and send confirmation message to the Store.



The tricky of this part of the program is that the message Store invoked in the bank message will also invoke a message from the Client side , thus the Method Invocation can be nested in the **RMI** mechanism.

Summery

By finishing this project, project members will have a deep understanding of the java **RMI** communication mechanism; it is a distributed object model for the Java language that retains the semantics of the Java object model, making distributed objects easy to implement and to use. The Java language's **RMI** system assumes the homogeneous environment of the Java Virtual Machine, and the system can therefore take advantage of the Java object model whenever possible. By finishing this project, each member gained a rich experience on the Borland J-builder software development platform, which is quite convenient for building Java interfaces. After finishing this project members will also have insight knowledge of the async hronous schemes in modern E-commerce Systems, which is quite prevalent today.