|  |
| --- |
| Concordia university |
| Comp6231 |
| PM1-RMI |
| **Team 3** |
| **Gay Hazan**  **Patrick Cristofaro**  **Ross Smith**  **Sai Sun** |
| **2015/6/5** |

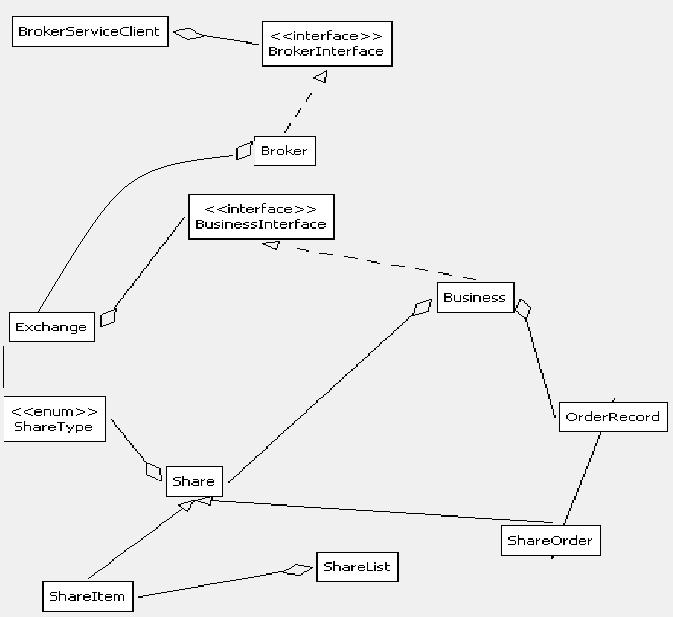
# Stock broker based on java rmi

This documentation is intended to provide a systematic view of the stock broker project which features component communications based on CORBA technology.

## Class Hierarchy

### class diagrams

The following class diagrams shows the overall relationships among the most important logic classes.



There are three CORBA interfaces used in the project, BusinessInterface which is used for internal communication between Exchange service to Business components which in this case stands for business services. Four Business remote objects will be binded to CORBA naming service, and accessible functions are defined in BusinessInterface.

BrokerInterface is used by Exchange service to communicate with client side. It offers operations accessible by clients.

ExchangeServerIF is the interface to provide advanced operations defined in requirements of PM2, namely, registerBusiness, unregisterBusiness and etc.

And in PM2 we also feature the utilization of CORBA naming service.

### Execution instructions

To execute the project, first run the projectLauncher.java, which will run java corba naming service automatically, and start default exchange service and broker service, and put all the service object references to naming service.

There are two clients available for different utilizations, BrokerServiceClient.java is the client which should be used for normal operations, such as display listings, sell/but shares.

ExchangeServiceClient\_CORBA.java provides the operations to add/remove a business from exchange server, update share price of a given business and get business info.

### Class details

* Buiness.java

The concrete logic class for share processing and transaction management. Upon initialization, the correspondent data file will be read and loaded into memory. Once there is a incoming transaction, it will process the transaction and write xml record file to file system. It also create and list the remote objects to RMI registry upon execution.

* BusinessInterface

Java RMI interface definition, internal use for stock broker project.

* OrderRecord

Class used to represent stock order record, used by Business logic to write stock order to record XML file.

* BrokerServiceClient

Front end interface of stock broker project, provides a command line fashion UI.

* Customer

Helper class to represent customer info.

* LoggerServer

Start logger thread on a given port. And communicate with LoggerClient via UDP messages.

* LoggerThread

Contains implementation to write a message to log file.

* Share

Used by Business class to represent share info.

* Broker.java

Implementation of the Broker RMI interface, and also a wrapper for the Exchange class, it redirect user operations to proper function logics inside Exchange class.

* Exchange.java

Concrete class contains logics for exchange processing, has access to three business components, and redirect share processing to proper business component.

## File hierarchy

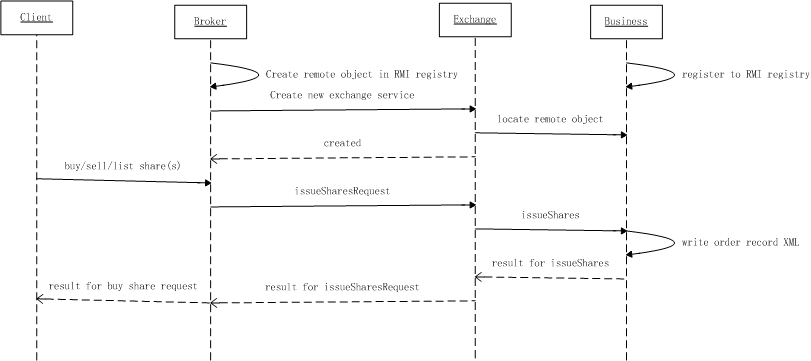
Stock information are kept in csv files inside files package, for example the contents of the csv files looks like the following sample:

GOOG,COMMON,540.11  
GOOG.B,CONVERTIBLE,532.23  
GOOG.C,PREFERRED,541.28

Log files are created under the same package as well.

Resource directory contains project related resources, e.g. project dependent library jar files, documentations, logs, settings for running the project.

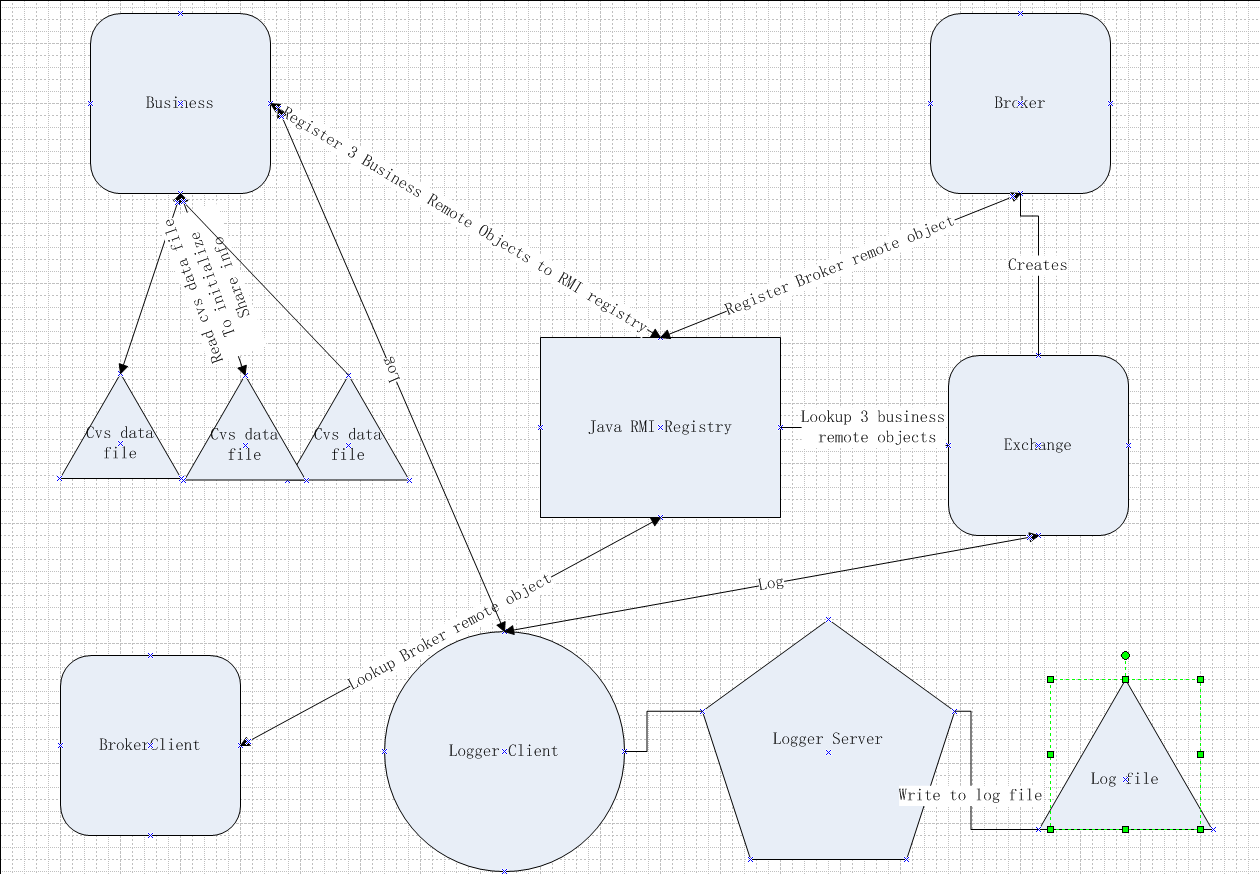
## workflow diagram



As shown in the work flow diagram, Broker will register its remote object instance to Java RMI registry, and the same will happen for the Business class, it will register three different remote objects which represents different business shares into RMI registry.

Exchange service will use the information from configuration file to locate the RMI registry, and then acquire the remote object for the three business instances. Broker instance will be responsible of creating Exchange service, and it will direct client RPC calls to the proper logics inside Exchange class, which in turn will do some processing to the three business remote objects.

## system interaction diagram



The project is a distributed system for stock broker. We assume the system has Business , StockExchange&Broker and BrokerClient, three parts. Business class creates three objects stand for three different companies, and register them into RMI registry as three remote objects, the same as Broker, it also register its object in the RMI registry. There is one BusinessInterface and one BrokerInterface. The interfaces defined which functions in the remote object can be called by StockExchange and BrokerClient.   
How does RMI work? For example, when the BrokerClient want to communicate with Broker, it uses the port number and host name to find the RMI registry and then lookup the remote object with its name ”Broker”. Then the BrokerClient perform some actions such as buy or sell shares, which has already defined in the BrokerInterface.