# CSN10101 Distributed Systems

# Coursework

# 04011725

## Section 1 – Description of the System

The system consists of 2 main components, 1 server – StockServer, and 1 or more clients – StockClient. The server is started up with default stock levels and then listens on the hardcoded TCP port – 1001 – for client connections. The clients connects using a TCP socket with a hardcoded host and port – set to the local machine, and again port 1001, and then display a simple text menu to the end user. When the user selects a menu option, the client sends a text string appropriate to that option to the server, and then optionally requests more details from the client, also sending the data to the server. When the client has sent all the data necessary for the menu option (serial number of a stock item to view, for example), it listens for a return message from the server, prints this to the screen, and returns to the menu.

Each connection in the server creates a new thread to handle the connection, and invokes a separate handler class which processes incoming messages. The first message in a group determines which function of the system is needed, and the handler passes control to a method for dealing with the specific request. This method listens for any additional data required, performs the relevant option, and returns a sucess or failure message, with any other appropriate information, to the client.

To store stock details, the server features a centralised repository which can be accessed from each instance of the handler class. This stores specialised stockItem objects which contain the data which is necessary for recording each item. As well as methods for accessing and updating this repository and the objects in it, the server also features a way for the handler to lock individual objects to prevent race conditions where 2 clients might simultaneously try to alter the stock levels on a single object.

## Section 4 - Implementation of “Add Stock” feature

This involved adding to 2 classes, the StockClient class and the StockHandler class.

StockClient:

(in the main method)

if (choice.equals("n")){

send("newitem");

System.out.print("Enter new item serial number\t");

String serial = getString();

System.out.print("Enter new item description\t");

String description = getString();

System.out.print("Enter new item location\t");

String location = getString();

System.out.print("Enter initial stock level\t");

//None of the others check for exceptions either... :P

int amount = Integer.parseInt(getString());

StockItem newItem = new StockItem(serial,description, amount, location);

send(newItem);

System.out.println(receive());

}

public static void showMenu(){

System.out.println("Stock Control Menu\n");

System.out.println(getDateTime()+"\n\n");

System.out.println("\t e - stock level enquiry");

System.out.println("\t h - item history");

System.out.println("\t a - add new stock");

System.out.println("\t n - new stock line");

System.out.println("\t s - sale");

System.out.println("\t l - stocklist\n");

System.out.println("\t q - disconnect");

}

StockHandler:

(in the run method)

if (command.equals("newitem"))

send(newItem());

public String newItem(){

StockItem newItem = (StockItem) receive();

// Check if the object already exists before adding it.

// Note that this isn't perfect - another client could add just after we check and before we try

// The failure in that case would be silent - the last action to succeed would win and the old

// value would be dropped.

// http://java.sun.com/j2se/1.5.0/docs/api/java/util/TreeMap.html#put(K, V)

if (server.exists(newItem.getserialNo()))

{

return "Serial number already exists";

} else {

server.put(newItem.getserialNo(),newItem);

return newItem.toString();

}

}