****

**Telecommunication Assignment 1 Command & Control Documentation**

**Name : Alan Abraham**

**Student Number: 18322576**

**Lab Group: A2**

**Demonstrator: Ali Talpur**

**CONTENTS**

**1 Introduction.........................................................................................................**

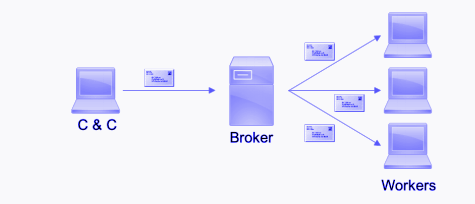
**2 Implementation...................................................................................................**

**3 Theory…..............................................................................................................**

**3 Reflection............................................................................................................**

***Introduction***

The aim of this assignment was to create a command and control protocol which had the ability to send and receive packets which contain certain data while learning about socket and threads. In my implementation the four actors that played a part in the main process are namely the C&C server , Broker, Worker1 and Worker2. My implementation of this command and control network allows the C&C server to send a package containing the work description to the broker and the broker being able to relay it to the workers. The workers can then agree or disagree to do the work and sends a package relaying said message back to the broker.



***Implementation***

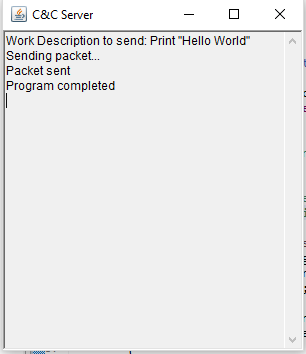
As opposed to other students doing this module I decided to take a simpler approach to this assignment. I make use of four main actors or classes in this assignment. These actors are C&C server, broker, worker1 and worker2 as mentioned above.

Each actor sends packets and receive packets from each other. I will go into detail about these actors and how I implemented them below.

***C&C Server (Client Class)***

The C&C Server stands for the Command and Control Server and in this case I consider it a client. In this assignment the Command and Control Server sends a packet to the broker. This packet contains the work description that is to be sent to the workers. The methods that this class uses are OnReceipt(), Start() and the Main.

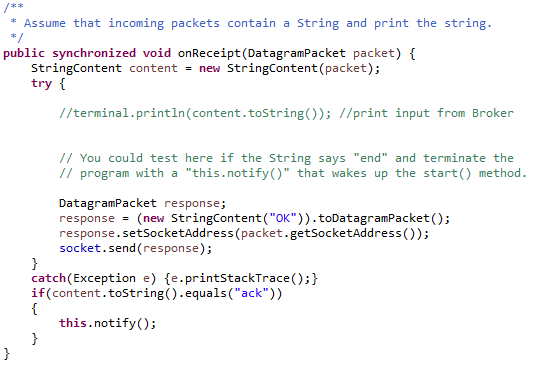
The class also contains the constructor for the C&C Server terminal and initialises this terminal in the main.



Above: Image of GUI for C&C Server

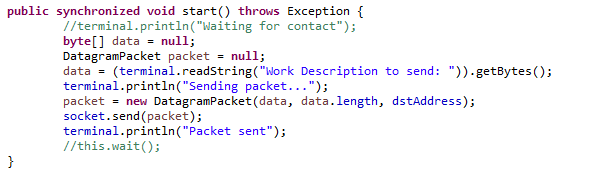
*OnReceipt(Packet):*

This function takes a packet that the C&C server receives as a parameter and sends a response to the broker. The response is sent in a packet to the broker and its “OK” technically it’s a form of acknowledgment from the server to the broker that tells the broker that the server did receive the message.



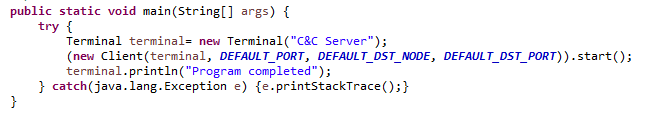
*Start():*

This function initialises the C&C server and it also throws exceptions. When the user types in the terminal it takes input and sends it to the broker as a packet and then it prints out that the packet has been sent successfully if it is received by the broker.



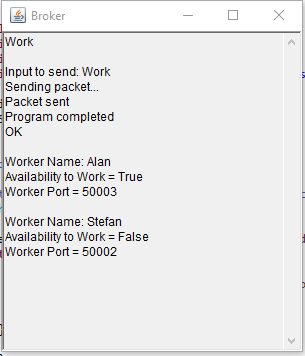
*Main()*

This is the main method of the C&C server. This method initialises the terminal for the C&C server within a try-catch statement. When the program ends the main also prints out “Program Completed”.



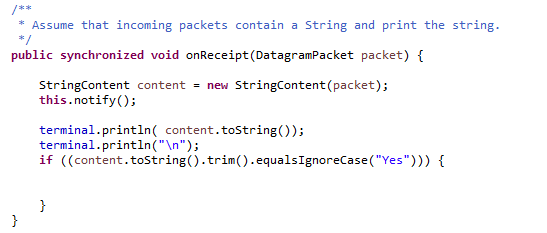
***Broker***

The broker class receives messages from the server and sends those messages to the worker classes and the workers reply with their availability, port number and worker name. The broker class contains the methods OnReceipt(),Start() and the Main method.



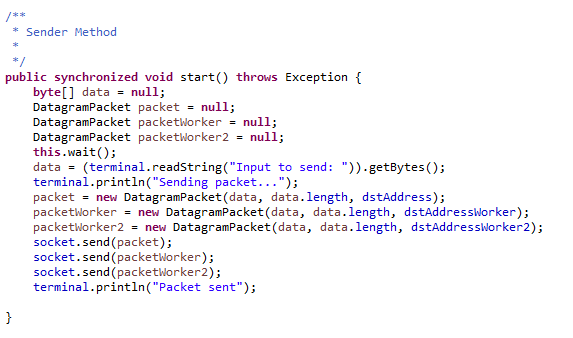
*OnReceipt(Packet)*

This function takes a packet that broker receives from the server as a parameter and sends it to the workers. The work description is sent in a packet from the broker to the workers and then the broker receives information from the workers consisting of the workers name, port and availability to work.



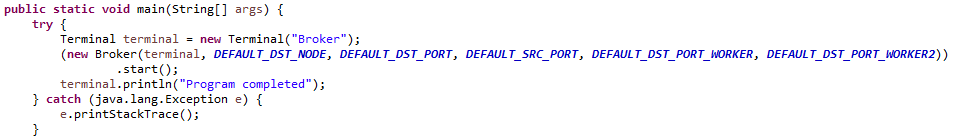
*Start()*

This function initialises the broker and it also throws exceptions. When the user types in the terminal it takes input and sends it to the workers as a packet and then it prints out that the packet has been sent successfully if it is received by the workers

.

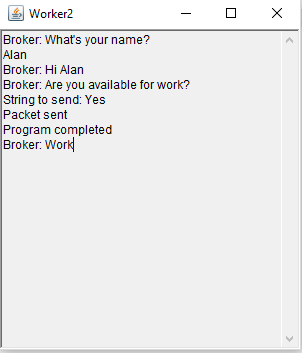
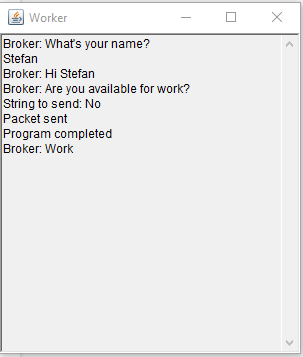
Main()

This is the main method of the broker class.This method initialises the terminal for the broker within a try-catch statement. When the program ends the main also prints out “Program Completed”.



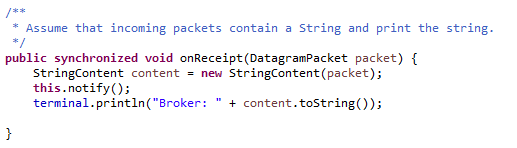
***Workers***

The worker class does work and replies to the broker by sending acknowledgements such as sending worker name,port and availability to work.This class is what I struggled the most on because I couldn’t get the workers to actually do any work which caused a lot of disappointment for myself.

*OnReceipt(Packet)*

This function takes a packet that worker receives from the broker as a parameter and sends it to the workers. The work description is sent in a packet from the broker to the workers. This function allows the workers to receive the work description and other messages from the broker.



*Start()*

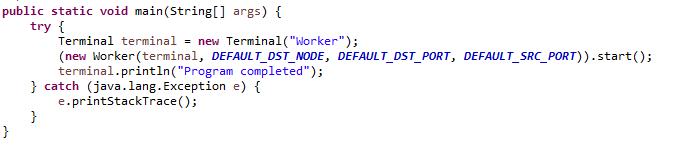
This function initialises the worker class and it also throws exceptions. When the user types in the terminal it takes input and sends it to the broker as a packet and then it prints out that the packet has been sent successfully if it is received by the broker.The packet contains the worker name, port and availability to work.



Above: Start() method of Worker Class

Main()

This is the main method of the worker class.This method initialises the terminal for the worker within a try-catch statement. When the program ends the main also prints out “Program Completed”.



***Theory***

Where a sender sends data packets to a receiver. Our requirement is that all of the packets fed into the sender must be successfully received by the receiver and arranged in the correct order. The sender can send packets sequentially knowing that they will be received. The task is to develop a system to guarantee that they are received without ﬂaw

***Acknowledgements***

A solution to the problem that we have could be to only allow the sender to send a single packet at a time. After sending a packet the sender will wait for acknowledgement that the packet was received before sending the next packet. These acknowledgements can be done with ACK packets. The Receiver will send one of these ACK’s upon the receipt of a packet. Acknowledgments were another part of my protocol that I could have improved. I spent so long on the project that by the time I went to fix this it was too difficult to change all of the code.

***Stop and Wait ARQ***

If I had not made such mistakes as I mentioned above I would have finished a fully functional system that solves our problem. This protocol is known as Stop and Wait ARQ (automatic repeat-request). A downside of stop and wait is that the sender must wait around for an ACK before it can send the next packet. The packet rate is limited directly by latency.

***Reflection***

In conclusion I believe I could have improved upon this protocol if I had not made certain mistakes or even if I had some more extra time. However, I have learned more about sockets, threads and the different types of ARQ due to this assignment and I am content that I was able to create this protocol to the best of my ability.