NWEN303

Project one

Liam De Grey

300294561

Part A)

For this part I have followed my own procedure with how the program runs.   
A set number of clients and providers are first made, these are all shuffled into one big list so that the order the threads start in is unknown. From here the threads are started one by one, straight after each other. Each will thread will run a set number of times as set by ‘count’. Each cycle starts by finding a random number between 1 and 2; this is its service Id. If the thread is a client, it will first check all existing provided services. If a service provided id matches the threads service id then a match is found. Once the match is found, the thread sleeps for a random amount of time between 1ms and 500ms. Once this completes, the thread attempts to enter a synchronized block, synchronized over the service provided ids list. In here, it will check that the service id still exists. If yes, then the removal of the service will commence. Else the client will simply return from the thread and post its own service needed. Once the post is made, the thread will sleep for random intervals, checking after each interval if the service posted has been met yet. If the service does not get accepted by a provider, then the thread will timeout, remove the posted service, and run through the next cycle. This procedure is the same for providers, however, providers post the services provided, and look for services needed.

I have decided, after much consideration, to remove more of the randomness involved in this program. By altering the randomly assigned sleep times to set times of 500ms, I am able to view the behaviors of the program better. This means I can trial different amounts of threads, loops etc. and be able to compare the results and how they all run.

I will be testing at the following points:

When a service /provided has been added  
 When a service /provided has been fulfilled  
 When a service /provided has been removed

I will expect to see similar results throughout the tests, however, I will note down any major issues such as bottlenecks and deadlocks etc.

I have since changed my implementation from using simple synchronization over the arrays to creating specific locks to be used when adding, removing services provided and services needed