PROGRAMMING ASSIGNMENT – 2

Vaishnavi Shah – 33984452
Ashlin Rodrigues – 34054291

MyDBClient.java :

```
package client;

import edu.umass.cs.nio.AbstractBytePacketDemultiplexer;
import edu.umass.cs.nio.MessageNIOTransport;
import edu.umass.cs.nio.interfaces.NodeConfig;
import edu.umass.cs.nio.nioutils.NIOHeader;
import server.SingleServer;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.InetSocketAddress;

/**
* This class should implement your DB client.
*/
public class MyDBClient extends Client {
private NodeConfig<String> nodeConfig= null;
public MyDBClient() throws IOException {
}
public MyDBClient(NodeConfig<String> nodeConfig) throws IOException {
super();
this.nodeConfig = nodeConfig;
}
}
```

MyDbSingleServer.java:

```
package server;
```

```
import edu.umass.cs.nio.AbstractBytePacketDemultiplexer;
import edu.umass.cs.nio.MessageNIOTransport;
import edu.umass.cs.nio.nioutils.NIOHeader;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
* This class should implement the logic necessary to perform the requested
* operation on the database and return the response back to the client.
*/
public class MyDBSingleServer extends SingleServer {
public MyDBSingleServer(InetSocketAddress isa, InetSocketAddress isaDB,
String keyspace) throws IOException {
super(isa, isaDB, keyspace);
}
}
```

SingleServer.java:

```
package server;

import edu.umass.cs.nio.AbstractBytePacketDemultiplexer;
import edu.umass.cs.nio.MessageNIOTransport;
import edu.umass.cs.nio.nioutils.NIOHeader;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
* @author arun
```

```
*
* This class implements a simple echo server using non-blocking IO.
*/
public class SingleServer {
public static final int DEFAULT_PORT = 2000;
public static final String DEFAULT_ENCODING = "ISO-8859-1";

protected static final Logger log = Logger.getLogger(SingleServer
.class.getName());
protected final MessageNIOTransport<String,String> clientMessenger;

public SingleServer(InetSocketAddress isa, InetSocketAddress isaDB, String
keyspace) throws IOException {
this.clientMessenger = new
MessageNIOTransport<String, String>(isa.getAddress(), isa.getPort(),
new AbstractBytePacketDemultiplexer() {
@Override
public boolean handleMessage(byte[] bytes, NIOHeader nioHeader) {
handleMessageFromClient(bytes, nioHeader);
return true;
}
});
}

// TODO: process bytes received from clients here
protected void handleMessageFromClient(byte[] bytes, NIOHeader header) {
// simple echo server
try {
log.log(Level.INFO, "{0} received message from {1}", new Object[]
{this.clientMessenger.getListeningSocketAddress(), header.sndr});
this.clientMessenger.send(header.sndr, bytes);
} catch (IOException e) {
e.printStackTrace();
```

```
}
}

/**
* @param args The first argument must be of the form [host:]port with an
optional host name or IP.
*/
public static InetSocketAddress getSocketAddress(String[] args) {
return new InetSocketAddress(args.length>0 && args[0].contains(":")?args
[0].replaceAll(":.*",""):"localhost",

args.length>0?Integer.parseInt(args[0]
.replaceAll(".*:",")):DEFAULT_PORT);
}

public void close() {
this.clientMessenger.stop();
}

public static void main(String[] args) throws IOException {
new SingleServer(getSocketAddress(args), new InetSocketAddress
("localhost", 9042), "demo");
};
}
```

client.java:

```
package client;

import edu.umass.cs.nio.interfaces.NodeConfig;
import server.SingleServer;
import edu.umass.cs.nio.AbstractBytePacketDemultiplexer;
import edu.umass.cs.nio.MessageNIOTransport;
import edu.umass.cs.nio.nioutils.NIOHeader;
```

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.InetSocketAddress;

/**
* @author arun
*
* This class implements a simple client to send requests and receive
* responses using non-blocking IO.
*/
public class Client {
private final MessageNIOTransport<String, byte[]> nio;

public Client() throws IOException {
this.nio = new
MessageNIOTransport<String, byte[]>(
new AbstractBytePacketDemultiplexer() {
@Override
public boolean handleMessage(byte[] bytes, NIOHeader nioHeader) {
handleResponse(bytes, nioHeader);
return true;
}
});
}

// TODO: process responses received from server
protected void handleResponse(byte[] bytes, NIOHeader header) {
// expect echo reply by default here
try {
System.out.println(new String(bytes, SingleServer.DEFAULT_ENCODING));
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
```

```
}
}
```

What i’m trying to do

1. The client should be able to continue to send requests as a `String` to the server and have the server execute them on the local Cassandra instance.
2. The server should send some response back to the client when the operation is complete. It is not important to parse the request or the structure of the response.
3. The server should connect to a keyspace name provided in its constructor.
4. `MyDBClient.java` should extend `Client.java` and `MyDBSingleServer.java` should extend `SingleServer.java` as indicated.
5. The classes `Client.java` and `SingleServer.java` can not be modified. As a consequence, the final method `Client.send` can also not be modified.
6. The method `Client.callbackSend` is the key method you need to override and it should, as documented, invoke the callback upon (and only upon) completing the execution of the request it previously sent to the server.

Note that the simple `Client.send` is non-blocking and is the only way to send requests to the server, and `Client.handleResponse` is the only place it receives responses. So if you call the supplied callback immediately after sending the request in callbackSend, that would be incorrect. You need to implement logic to match a received response with the corresponding previously sent request. For example, you could include a unique request identifier in each request and its corresponding response in order to match the two.