Assignment 1.1

Client.java

```
import java.io.*;
import java.net.*;
public class Client {
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello Server");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Server.java

```
 import java.io.*;
    import java.net.*;
    public class Server {
    public static void main(String[] args){
    try{
    ServerSocket ss=new ServerSocket(6666);
    Socket s=ss.accept();//establishes connection
    DataInputStream dis=new DataInputStream(s.getInputStream());
    String  str=(String)dis.readUTF();
    System.out.println("message= "+str);
    ss.close();
    }catch(Exception e){System.out.println(e);}
    }
    }
```

Assignment 1.2

Clients.java

```
import java.rmi.*;
public class Clients {
public static void main(String args[]) {
try {
String Serverurl = "rmi://" + args[0] + "/Server";
ServerIntf intf = (ServerIntf) Naming.lookup(Serverurl);
String str = "Sample string";
System.out.println("output from server is :" + intf.upper(str));
} catch (Exception e) {
System.out.println("Exception");
}
}
}
```

Server1.java

```
import java.rmi.*;
import java.net.*;
public class Server1 {
public static void main(String args[]) throws Exception {
ServerImpl impl = new ServerImpl();
Naming.rebind("Server", impl);
}
}
```

ServerImpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class ServerImpl extends UnicastRemoteObject implements ServerIntf {
public ServerImpl() throws RemoteException {
}
public String upper(String s) throws RemoteException {
return s.toUpperCase();
}
}
```

Server Intf.java

```
import java.rmi.*;
public interface ServerIntf extends Remote {
String upper(String s) throws RemoteException;
}
```

Assignment 2

ScatterGather.java

```
import mpi.MPI;
public class ScatterGather {
public static void main(String args[]){
MPI.Init(args);
int rank = MPI.COMM_WORLD.Rank();
int size = MPI.COMM_WORLD.Size();
int root=0;
int sendbuf[]=null;
sendbuf= new int[size];
if(rank==root){
sendbuf[0] = 10;
sendbuf[1] = 20;
sendbuf[2] = 30;
sendbuf[3] = 40;
//print current process number
System.out.print("Processor "+rank+" has data: ");
for(int i = 0; i < size; i++){
System.out.print(sendbuf[i]+" ");
}
System.out.println();
}
//collect data in recvbuf
int recvbuf[] = new int[1];
MPI.COMM_WORLD.Scatter(sendbuf, 0, 1, MPI.INT, recvbuf, 0, 1, MPI.INT, root);
System.out.println("Processor "+ rank +" has data: "+recvbuf[0]);
System.out.println("Processor "+ rank +" is doubling the data");
recvbuf[0]=recvbuf[0]*2;
MPI.COMM_WORLD.Gather(recvbuf, 0, 1, MPI.INT, sendbuf, 0, 1, MPI.INT, root);
if(rank==root){
System.out.println("Process 0 has data: ");
for(int i=0;i<4;i++){
System.out.print(sendbuf[i]+ " ");
}
}
MPI.Finalize();
}
}
```

Assignment 3

ReverseModule.idl

```
module ReverseModule
{
interface Reverse
{
string reverse_string(in string str);
};
};
```

ReverseClient.java

```
import ReverseModule.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;
class ReverseClient
{
public static void main(String args[])
{
Reverse ReverseImpl=null;
try
{
// initialize the ORB object request broker
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
String name = "Reverse";
// narrow converts generic object into string type
ReverseImpl = ReverseHelper.narrow(ncRef.resolve_str(name));
System.out.println("Enter String=");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str= br.readLine();
String tempStr= ReverseImpl.reverse_string(str);
System.out.println(tempStr);

}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```

ReverseServer.java

```
import ReverseModule.Reverse;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
class ReverseServer
{
public static void main(String[] args)
{
try
{
// initialize the ORB
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
// initialize the BOA/POA
POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
rootPOA.the_POAManager().activate();
// creating the object
ReverseImpl rvr = new ReverseImpl();
// get the object reference from the servant class
org.omg.CORBA.Object ref = rootPOA.servant_to_reference(rvr);
System.out.println("Step1");
Reverse h_ref = ReverseModule.ReverseHelper.narrow(ref);
System.out.println("Step2");
org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
System.out.println("Step3");
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
System.out.println("Step4");
String name = "Reverse";
NameComponent path[] = ncRef.to_name(name);
ncRef.rebind(path,h_ref);
System.out.println("Reverse Server reading and waiting....");
orb.run();
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```

Assignment 5

Calculator.java

```
import javax.jws.WebService;
import javax.xml.ws.Endpoint;
@WebService
public class Calculator {
public static void main(String[] args) {
Calculator.calculate(1, 1, "ADD");
Calculator.calculate(2, 3, "MULT");
Calculator.calculate(5, 2, "asdf");
Endpoint.publish("http://localhost:12345/calc", new Calculator());
}
public int compute(int x, int y, String operation) {
return Calculator.calculate(x, y, operation);
}
public static int calculate(int x, int y, String operation) {
int result;
String op;
if ("ADD".equals(operation)) {
result = x + y;
op = "+";
} else if ("SUB".equals(operation)) {
result = x - y;
op = "-";
} else if ("MULT".equals(operation)) {
result = x * y;
op = "*";
} else if ("DIV".equals(operation)) {
result = x / y;
op = "/";
} else {
// defaults to SUB

result = x - y;
op = "-";
}
log(x, y, result, op);
return result;
}
private static void log(int x, int y, int result, String op) {
```

```
System.out.format("%d %s %d = %d%n", x, op, y, result);
}
}
```

Assignment 6

Publisher.java

```
import javax.jms.*;
import org.apache.activemq.ActiveMQConnection;
import org.apache.activemq.ActiveMQConnectionFactory;

public class Publisher {
private static String url = ActiveMQConnection.DEFAULT_BROKER_URL;
public static void main(String[] args) throws JMSException {
ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);
Connection connection = connectionFactory.createConnection();
connection.start();
Session session = connection.createSession(false,

Session.AUTO_ACKNOWLEDGE);

Topic topic = session.createTopic("CL9");
MessageProducer producer = session.createProducer(topic);
TextMessage message = session.createTextMessage();
message.setText("This is a new message from publisher");
producer.send(message);
System.out.println("Sent message '" + message.getText() + "'");
connection.close();
}
}
```

Subscriber.java

```
package pubsub;
import java.io.IOException;
import javax.jms.*;
import org.apache.activemq.ActiveMQConnection;
import org.apache.activemq.ActiveMQConnectionFactory;
public class Subscriber {
private static String url = ActiveMQConnection.DEFAULT_BROKER_URL;
public static void main(String[] args) throws JMSException {
ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);
Connection connection = connectionFactory.createConnection();
connection.start();
Session session = connection.createSession(false,

Session.AUTO_ACKNOWLEDGE);

Topic topic = session.createTopic("CL9");
MessageConsumer consumer = session.createConsumer(topic);
MessageListener listner = new MessageListener() {
public void onMessage(Message message) {
try {
if (message instanceof TextMessage) {
TextMessage textMessage = (TextMessage)

message;

System.out.println("Received message" +

textMessage.getText() + "'");

}
} catch (JMSException e) {
System.out.println("Caught:" + e);
e.printStackTrace();
}
}
};
consumer.setMessageListener(listener);
try {

System.in.read();
} catch (IOException e) {
e.printStackTrace();
}
connection.close();
}
}
```

Assignment 7

```
script.py
from flask import Flask
from flask import jsonify, json
import pandas as pd
app = Flask(__name__)
@app.route('/')
def index():
return "HELLO WORLD"
@app.route('/users')
def disp_users():
data = pd.read_json('users.json')
data = data.values.tolist()
return jsonify(data)
@app.route('/movies')
def disp_movies():

data_movies = pd.read_json('movies.json')
data_movies = data_movies.values.tolist()
return jsonify(data_movies)
```

```
@app.route('/booking')
def disp_booking():
data_booking = pd.read_json('booking.json')
data_booking = data_booking.values.tolist()
return jsonify(data_booking)
@app.route('/showtimes')
def disp_showtimes():
data_showtimes = pd.read_json('showtimes.json')
data_showtimes = data_showtimes.values.tolist()
return jsonify(data_showtimes)
if __name__ == '__main__':
app.run(debug=True)
```