Source code

For Server:

**package** server;

**import** java.io.\*;

**import** java.net.\*;

**public** **class** Server {

**public** **static** **void** main(String[] args) {

**try** {

ServerSocket serverSocket = **new** ServerSocket(8000);

Socket socket = serverSocket.accept();

ObjectOutputStream out = **new** ObjectOutputStream(socket.getOutputStream());

ObjectInputStream in = **new** ObjectInputStream(socket.getInputStream());

String state = "INIT";

String command = "";

**while** (!command.equals("CLOSE")) {

// Wait for a command from the client

command = (String)in.readObject();

// Handle the command

**if** (command.equals("START")) {

// Begin a distributed transaction

**if** (state.equals("INIT")) {

state = "STARTED";

out.writeObject("OK");

} **else** {

out.writeObject("ERROR");

}

out.flush();

} **else** **if** (command.equals("UPDATE")) {

// Perform some work as part of the transaction

**if** (state.equals("STARTED")) {

state = "UPDATED";

out.writeObject("OK");

} **else** {

out.writeObject("ERROR");

}

out.flush();

} **else** **if** (command.equals("PREPARE")) {

// Prepare to commit the transaction

**if** (state.equals("UPDATED")) {

state = "PREPARED";

out.writeObject("OK");

} **else** {

out.writeObject("ERROR");

}

out.flush();

} **else** **if** (command.equals("COMMIT")) {

// Commit the transaction

**if** (state.equals("PREPARED")) {

state = "COMMITTED";

out.writeObject("OK");

} **else** {

out.writeObject("ERROR");

}

out.flush();

} **else** **if** (command.equals("CLOSE")) {

// Close the connection

out.writeObject("OK");

out.flush();

socket.close();

}

}

// Clean up resources

in.close();

out.close();

socket.close();

serverSocket.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

For Client:

**package** client;

**import** java.io.\*;

**import** java.net.\*;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

**try** {

Socket socket = **new** Socket("localhost", 8000);

ObjectOutputStream out = **new** ObjectOutputStream(socket.getOutputStream());

ObjectInputStream in = **new** ObjectInputStream(socket.getInputStream());

// Start a distributed transaction

out.writeObject("START");

out.flush();

String result = (String)in.readObject();

System.***out***.println("Transaction started: " + result);

// Perform some work as part of the transaction

out.writeObject("UPDATE");

out.flush();

result = (String)in.readObject();

System.***out***.println("Work done: " + result);

// Prepare to commit the transaction

out.writeObject("PREPARE");

out.flush();

result = (String)in.readObject();

System.***out***.println("Transaction prepared: " + result);

// Commit the transaction

out.writeObject("COMMIT");

out.flush();

result = (String)in.readObject();

System.***out***.println("Transaction committed: " + result);

// Close the connection

out.writeObject("CLOSE");

out.flush();

socket.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}