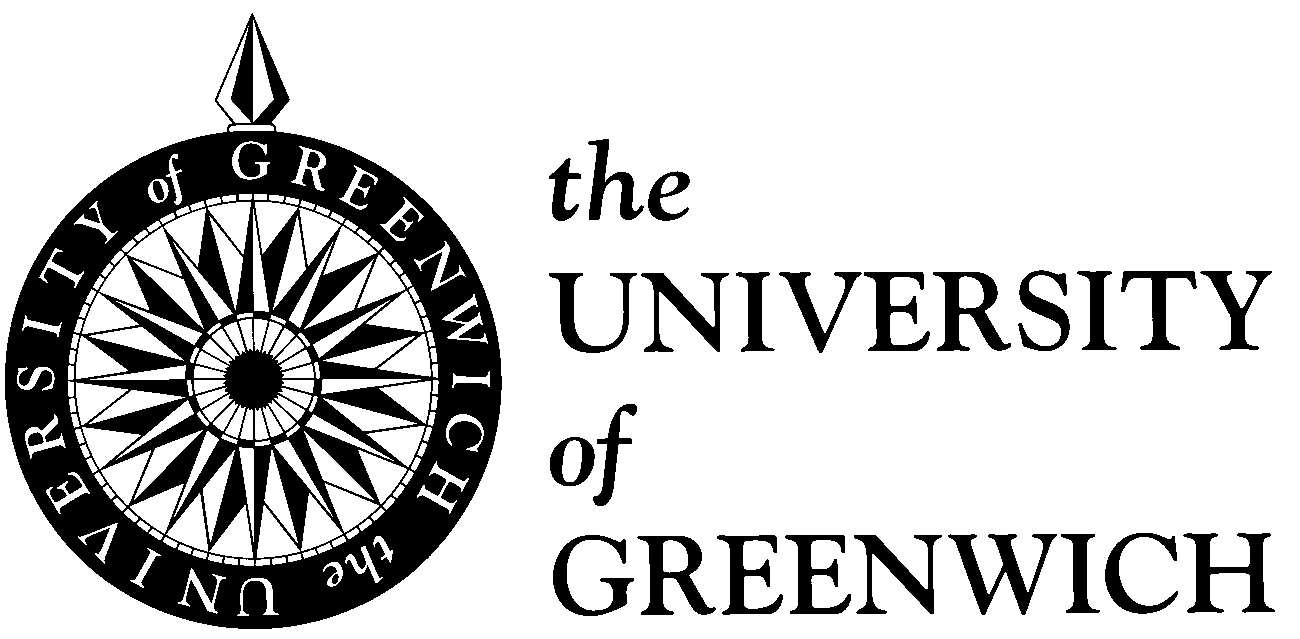
UNYT Logo 

University of New York, Tirana

**M.Sc. in Computer Science**

Distributed Systems

**Development of a distributed object based system based on Remote Method**

**Invocation.**

**Host:**

**Department of Computer Science**

**School of Computing and Mathematical Sciences**

**University of Greenwich**

**Partner:**

**University of New York in Tirana**

**Student: Ester Daci**

Contents

[1. Architecture 3](#_Toc15398106)

[2. Design of the components 4](#_Toc15398107)

[3. Implementation 7](#_Toc15398108)

[4. Test 8](#_Toc15398109)

[References 15](#_Toc15398110)

# Architecture

RMI Architecture consist in three parts:

A client displays a GUI (graphical user interface but, in this project, the result will be displayed in console) to a user, who get information about the products of an electronic store including some product’s information and sales. Clients communicate with the server using RMI. The server stores the information of products and sales in a database using JDBC, the Java relational database package.

With RMI you can have the client upload behavior from the server with a simple method invocation, providing a flexible way to offload computation from the server to the clients while providing users with faster feedback.

(https://www.oracle.com, n.d.)

Each remote object has two separate

parts

**Definition of its behavior**

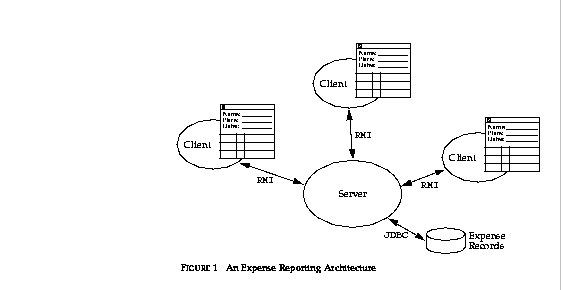
* Clients are concerned about the definition of a
* service
* Coded using a Java interface, which defines the
* behavior

**Implementation of its behavior**

* Servers are focused on providing the service
* Coded using a Java class, which defines the

implementation (University)

Figure 1 shows the general picture of this achitecture:

**Figure.1[[1]](#footnote-1)**

# Design of the components

**In this project  the remote interface that defines the methods the client can invoke on the server are:**

*import java.rmi.Remote;*

*import java.rmi.RemoteException;*

*public interface* ***Products*** *extends Remote*

*import java.rmi.Remote;*

*import java.rmi.RemoteException;*

*public interface* ***Sale*** *extends Remote {*

*// Add method to return master ElectronicStore*

*public ElectronicStore getElectronicStore()*

*throws RemoteException;*

*import java.rmi.Remote;*

*import java.rmi.RemoteException;*

*public interface* ***ElectronicStore*** *extends Remote*

The interfaces are a normal Java interface with two interesting characteristics

They extend the RMI interface named Remote, which marks the interface as one available for remote invocation.

All the methods throw RemoteException, which is used to signal network and messaging failures.

**The server looks like this:**

*import java.io.Serializable;*

*import java.util.Hashtable;*

*import java.rmi.server.UnicastRemoteObject;*

*import java.rmi.RemoteException;*

*import java.util.ArrayList;*

*import java.sql.\*;*

*//import static sun.io.Win32ErrorMode.initialize;*

*public final class ElectronicStoreImpl extends UnicastRemoteObject implements ElectronicStore{ ….. }*

The type UnicastRemoteObject defines the kind of remote object this server will be, in this case a single server. The Java class *ElectronicStoreImpl* implements the methods of the remote interface *ElectronicStore*. Clients on remote hosts can use RMI to send messages to *ElectronicStoreImpl* objects. (Java Remote Method Invocation - Distributed Computing for Java, n.d.)

**The remote class *ElectronicStoreServer* is:**

public class ElectronicStoreServer{..}

And one of its methods makes a connection with server :

*private void connectServer() {*

*try {*

*ES = new ElectronicStoreImpl();*

***Naming.rebind("rmi://localhost/ElectronicStore", ES);***

*connected = true;*

*} catch (MalformedURLException err) {*

*System.out.println(err.getMessage());*

*err.printStackTrace();*

*} catch (RemoteException err) {*

*System.out.println(err.getMessage());*

*err.printStackTrace();*

*}*

*if (connected) {*

*buttonDisconnect.setEnabled(true);*

*buttonConnect.setEnabled(false);*

*labelState.setText("RMI Server started");*

*System.out.println("Server Connected.");*

*} else {*

*buttonDisconnect.setEnabled(false);*

*buttonConnect.setEnabled(true);*

*labelState.setText("Pending to start RMi Server...");*

*System.out.println("Server Disconnected.");*

*}*

The Naming class provides methods for storing and obtaining references to remote objects in a remote object registry. (affiliates, 2018)

**JDBC-Direct to the Database**

Here is ElectronicStoreImpl that uses JDBC to implement all the methods that include the queries:

*public boolean initializeConnection(String SERVER, String DATABASE, String USER\_ID,*

*String PASSWORD) throws ClassNotFoundException, SQLException {*

*try {*

*Class.forName("com.mysql.jdbc.Driver").newInstance();*

*Conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/electronicsproducts","root","Ester123");*

*s = conn.createStatement();* (Biba, 2019)

TheElectronicStoreImpl class continues with the other methods as shown:

*getProductNumberFromDatabase();*

*getListOfProductsInfoFromDatabase();*

*getListOfProductWithAvUnitLess10();*

*getTotalSale();*

*getNrOfSaleLastMonth();*

*getTotAmOfSale();*

*getTotAmOfSaleLastMonth();*

# Implementation

**Software versions:**

Netbeans IDE 8.2

MySQL80

mysql-connector-java-8.0.17 (Corporation, n.d.)

Programming Language: Java

**Hardware Specifications:**

Windows Edition –Windows 10

RAM-8GB

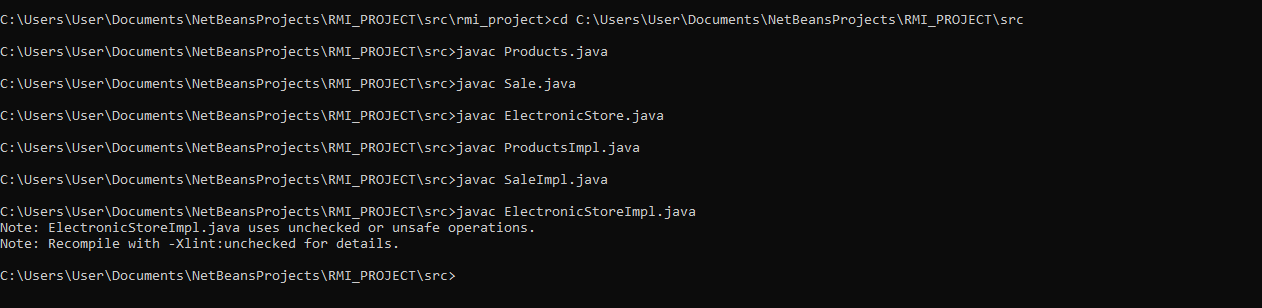
System Type -> 64-bit Operating System

# Test

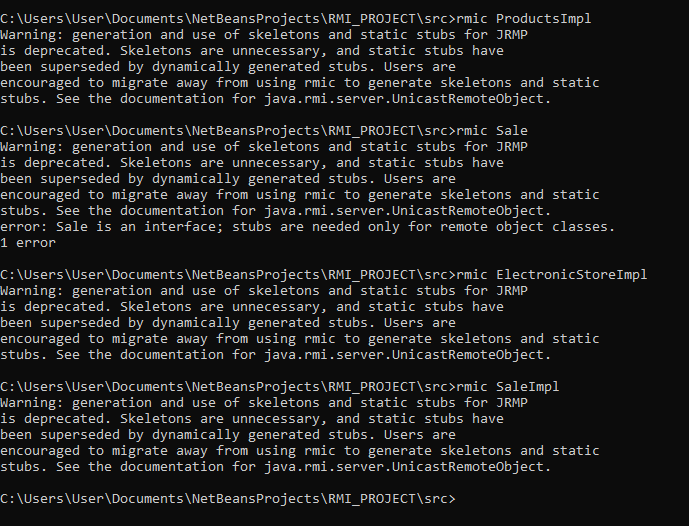
The directory of the RMI PROJECT where all the commands will be excecuted is :

*C:\Users\User\Documents\NetBeansProjects\RMI\_PROJECT\src*

The compilation of each class from **cmd** is shown below:



Stubs and skeletons are generated by the **rmic** compiler.



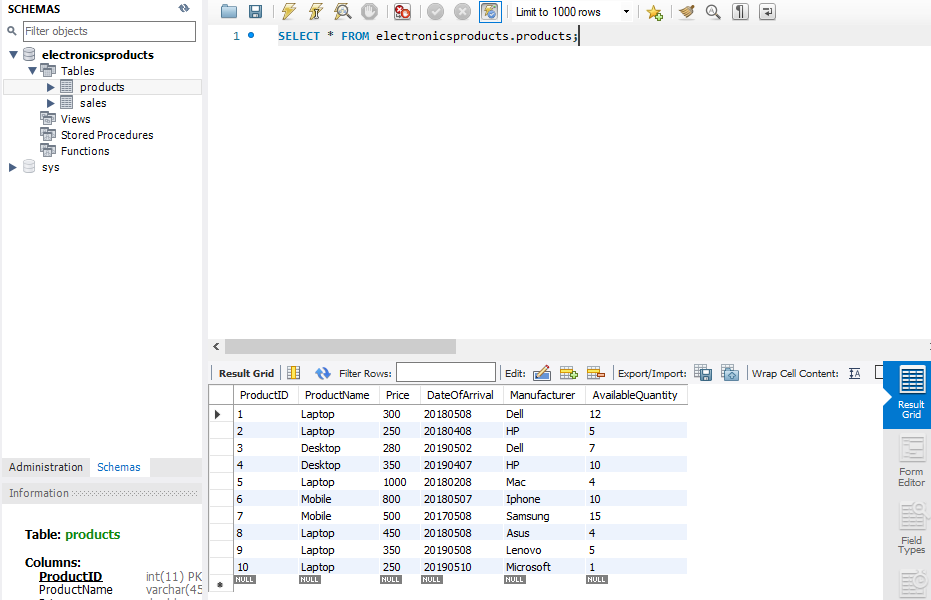
Compile the server

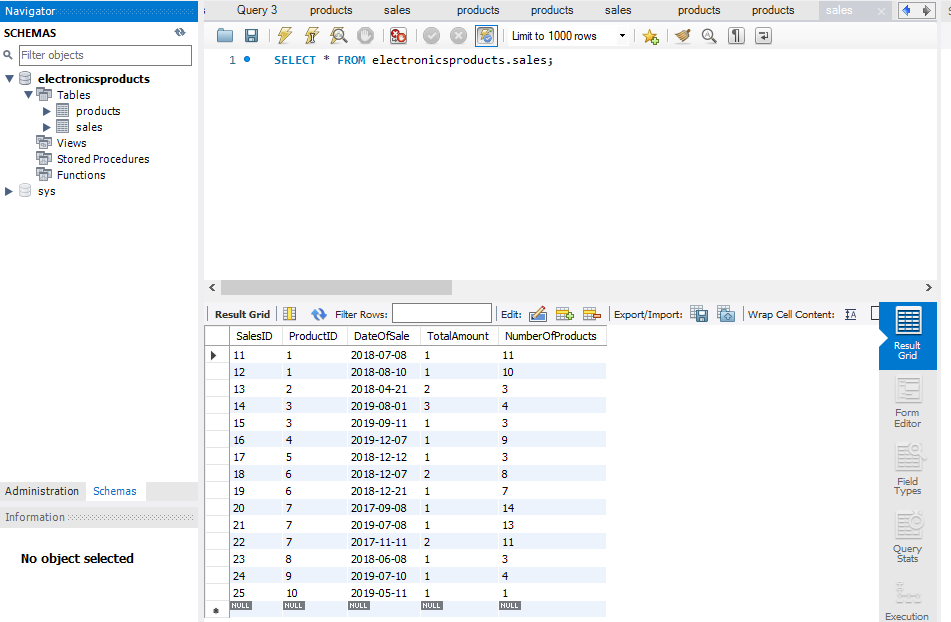


And then run the registry:



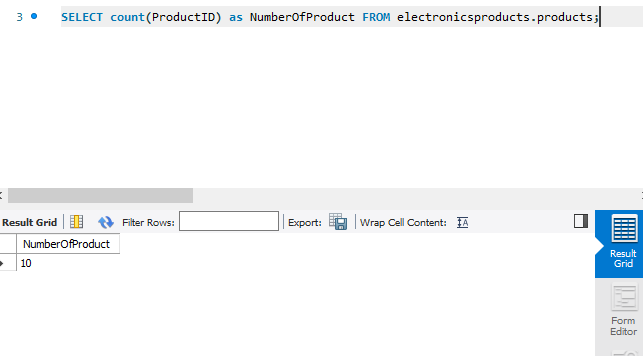
The database and the tables for this project are :



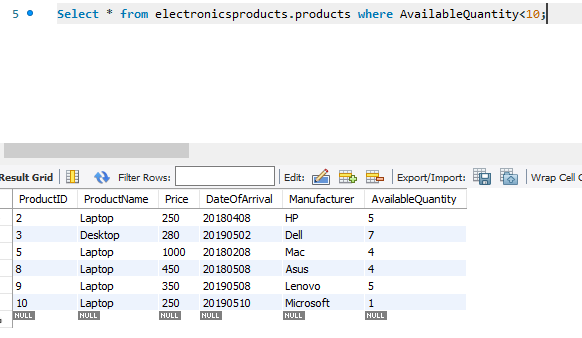


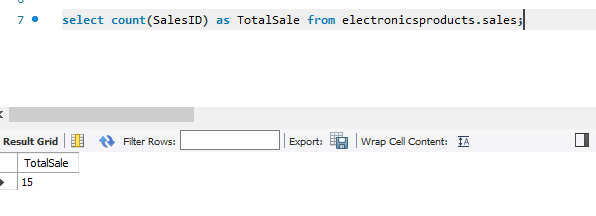
The queries required in the questions of the project:

---------------------------------------------------------------------------------------------------------

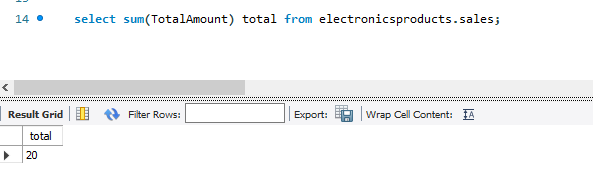


---------------------------------------------------------------------------------------------------------

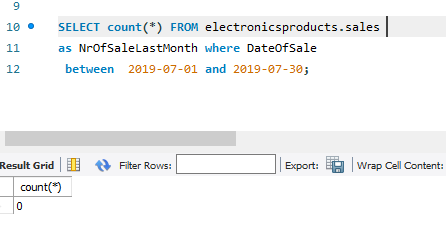




---------------------------------------------------------------------------------------------------------

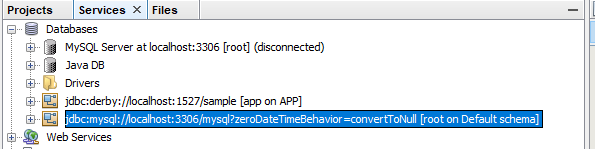


---------------------------------------------------------------------------------------------------------

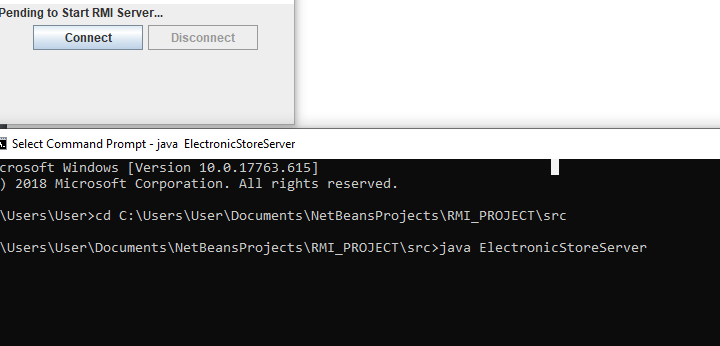


In order for Java program is properly set to connect to MySQL it is needed to add the

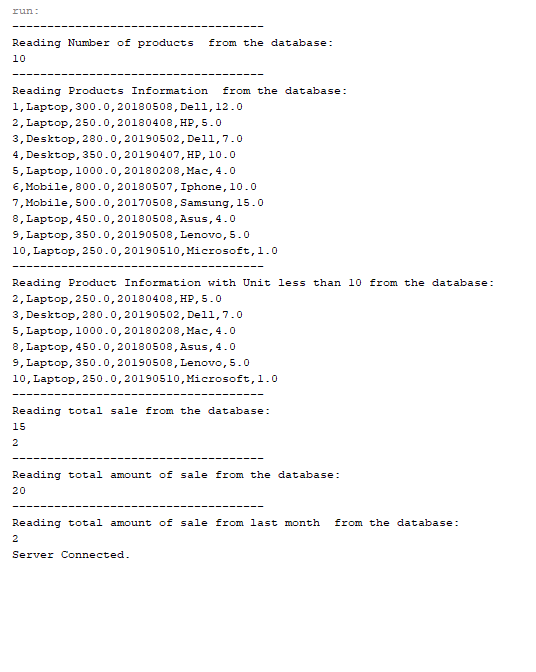
file : *mysql-connector-java-8.0.17* in Libraries and also it should be connect with DB in Services in Netbeans as shown in figure:



**Start the server:**



**And the result is:**



# References

affiliates, O. a. (2018). *https://docs.oracle.com.* Retrieved from https://docs.oracle.com/javase/7/docs/api/java/rmi/Naming.html: https://docs.oracle.com/javase/7/docs/api/java/rmi/Naming.html

Biba, M. (2019). *Remote Method Invocation.* Retrieved from http://www.marenglenbiba.net/distsys/LAB%20MANUAL%20RMI%20FULL.pdf.

Corporation, O. (n.d.). *Mysql.com*. Retrieved from https://dev.mysql.com/downloads/connector/j/

*https://www.oracle.com*. (n.d.). Retrieved from https://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html: https://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html

*Java Remote Method Invocation - Distributed Computing for Java*. (n.d.). Retrieved from https://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html: https://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html

University, D. o. (n.d.). *http://web.cs.iastate.edu/~yingcai/cs587x/notes/rmi.pdf.* Retrieved from http://web.cs.iastate.edu/~yingcai/cs587x/notes/rmi.pdf

1. <https://www.oracle.com/ocom/groups/public/@otn/documents/digitalasset/146937.gif> [↑](#footnote-ref-1)