|  |
| --- |
| ADVANCED Java Programming IT351 |
| IT351 Individual Project |
|  |
|  |
| **Nicholas Weinold** |
| **3/22/2017** |

|  |
| --- |
| This is my course Design Document including weekly Class Diagrams, Pseudocode, and Screenshots. The document may change if we do not utilize UML or pseudocode in the later weeks. This is just my functioning template as of now. |

Table of Contents

[Unit 1 Client-Server Applications: 3](#_Toc478021868)

[UML Class Diagram 3](#_Toc478021869)

[Pseudocode 3](#_Toc478021870)

[Screenshots 5](#_Toc478021871)

[Unit 2: Multi-Threaded Server Application 7](#_Toc478021872)

[UML Class Diagram 7](#_Toc478021873)

[Pseudocode 7](#_Toc478021874)

[Screenshots 9](#_Toc478021875)

[Unit 3 : Database-Driven Desktop Applications 12](#_Toc478021876)

[UML Class Diagram 14](#_Toc478021877)

[Pseudocode 15](#_Toc478021878)

[Screenshots 19](#_Toc478021879)

[Unit 4: Database-Driven Web Applications 26](#_Toc478021880)

[UML Class Diagram 26](#_Toc478021881)

[Pseudocode 27](#_Toc478021882)

[Screenshots 27](#_Toc478021883)

[User Installation/Operation Manual 32](#_Toc478021884)

[Unit 5: Database-Driven Web Services 33](#_Toc478021885)

[UML Class Diagram 33](#_Toc478021886)

[Pseudocode 34](#_Toc478021887)

[Screenshots 34](#_Toc478021888)

[Questions 39](#_Toc478021889)

[Test Plan 41](#_Toc478021890)

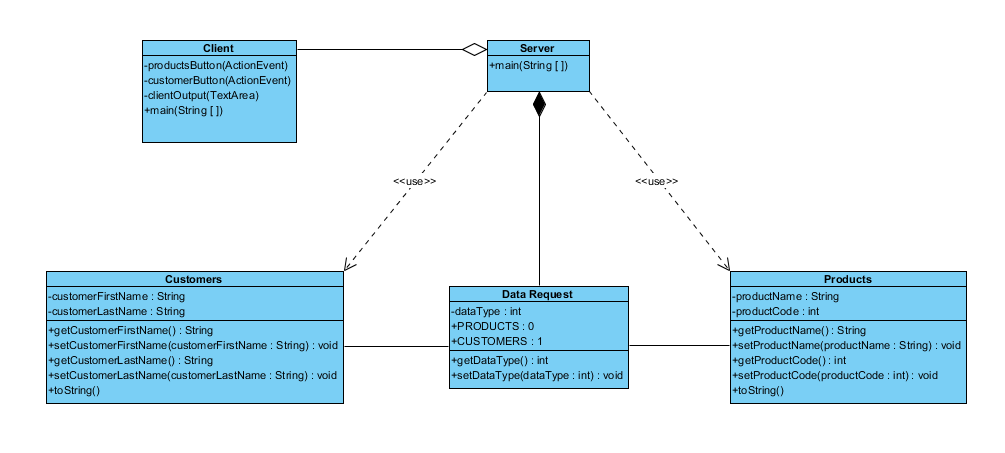
[User Installation/Operation Manual 42](#_Toc478021891)

[References 43](#_Toc478021892)

# Unit 1 Client-Server Applications:

This week we are asked to create a very simple server application that stores Product and Customer data. Then we are asked to create a client application that can communicate with the server application using a TCP or UDP “conversation”. The client application must be able to request and receive server stored data for both Products and Customers. Finally, we are asked to use Swing components (user JFrame Interface).

## UML Class Diagram



## Pseudocode

**Server Application Pseudocode**

START

Create ServerSocket

SET port number to 83

IF receive client READ data request

Get requested data (CUSTOMER or PRODUCTS)

OPEN socket

SEND requested data to client

Close Socket

**Client Application Pseudocode**

START

Create ServerSocket

SET port number to 83

Display JFrame GUI

USER SELECTS products button

OPEN Socket

Send data request for PRODUCTS

IF PRODUCTS data returned by server display on GUI TextArea

Close Socket

IF PRODUCTS data does not exist throw exception

IF no server socket communication throw Exception

USER SELECTS customers button

OPEN Socket

Send data request for CUSTOMERS

IF CUSTOMERS data returned by server display on GUI TextArea

Close Socket

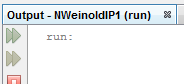
IF CUSTOMERS data does not exist throw exception

IF no server socket communication throw Exception

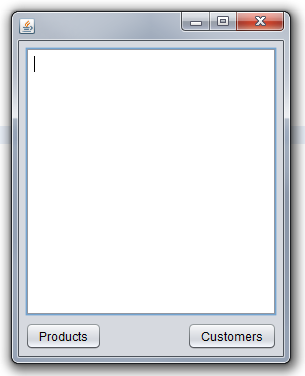
User Selects close window Client Application closes (Server remains open)

## Screenshots

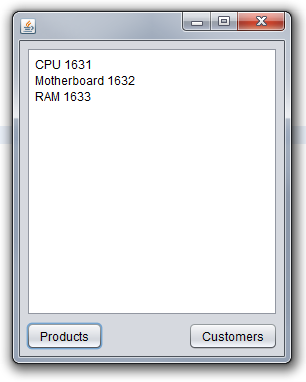
Server Starts



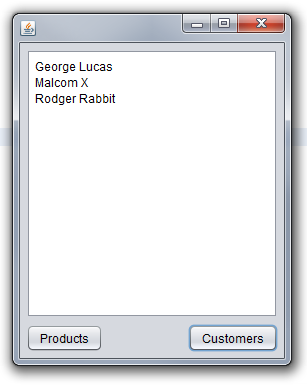
Client Starts



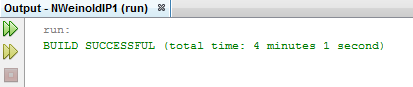
Client Selects Products button product data is displayed



Client Selects Customers Button Customer data is displayed



Client Closes GUI window build ends



# Unit 2: Multi-Threaded Server Application

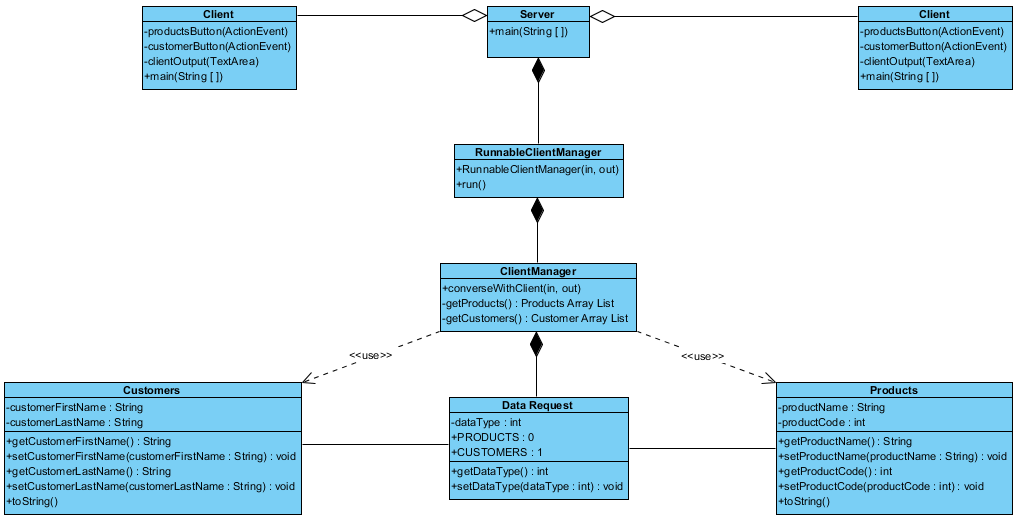
Modify your server to make it multithreaded.

Server should allow more than one client to connect and request product and customer data.

Test your multithreaded server by connecting more than one client to the server.

Take at least 4 screenshots to demonstrate that your server correctly responds to each client’s request.

## UML Class Diagram



## Pseudocode

**Pseudocode for the server:**

Server START

Creates socket 83

Opens socket

Infinitely looks for a new client

IF client connects uses the runnableClientManager to handle the new client

IF no client connects continues to look for new clients until closed.

END

**Pseudocode for the clientManager:**

\*\*\* this is where server data request and customer/product array handling was moved \*\*\*

Get the data type (product or customer)

If PRODUCT write out product data

Flush

IF CUSTOMER write our customer data

Flush

**Pseudocode for runnableClientManager:**

Run clientManager

Passes input stream or output stream from client to server or server to client using clientManager as a thread

**Client Pseudocode:**

START

Creates socket 83

Opens socket

Connects to server

User selects Product or Customer button

Sends request to server for product or customer data

Receives product or customer data and displays in textArea

IF user closes window client socket closes

END

**Second Client Pseudocode: (Runs at same time as first client)**

START

Creates socket 83

Opens socket

Connects to server

User selects Product or Customer button

Sends request to server for product or customer data

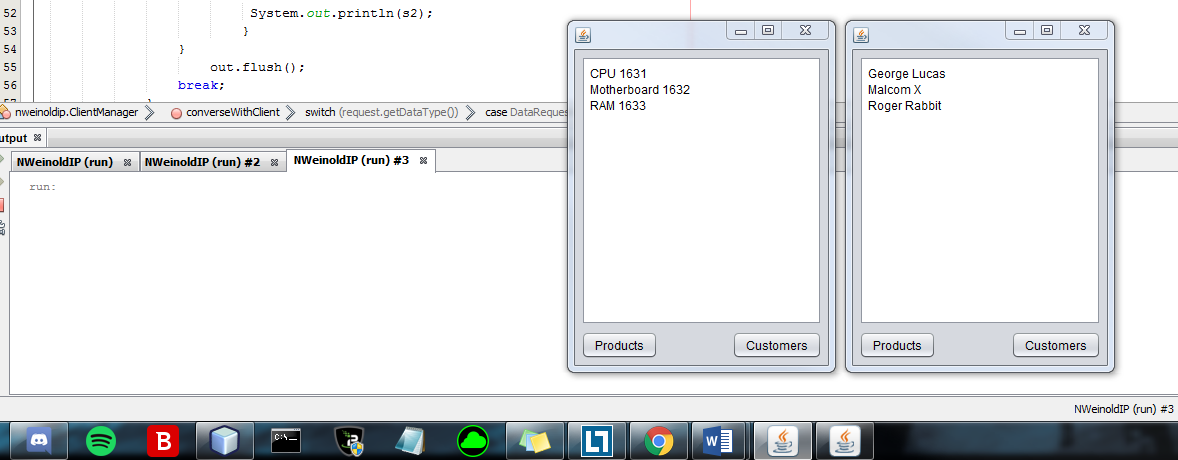
Receives product or customer data and displays in textArea

IF user closes window client socket closes

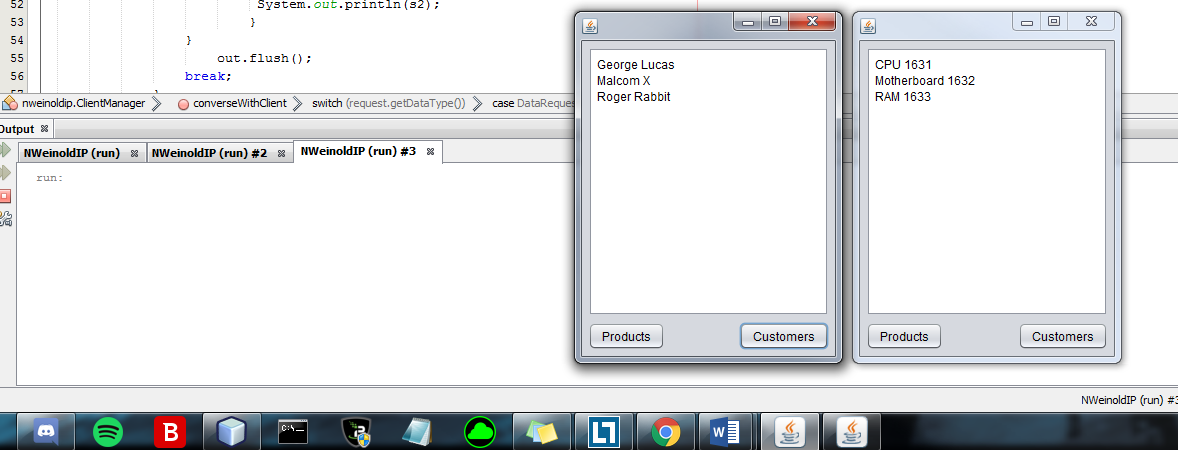
END

## Screenshots

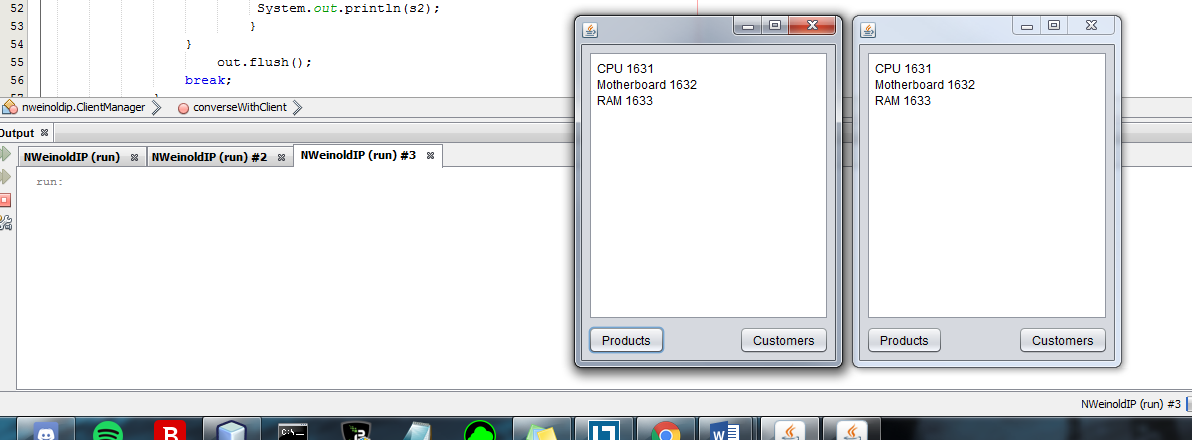
Two clients connected first requests products second requests customers



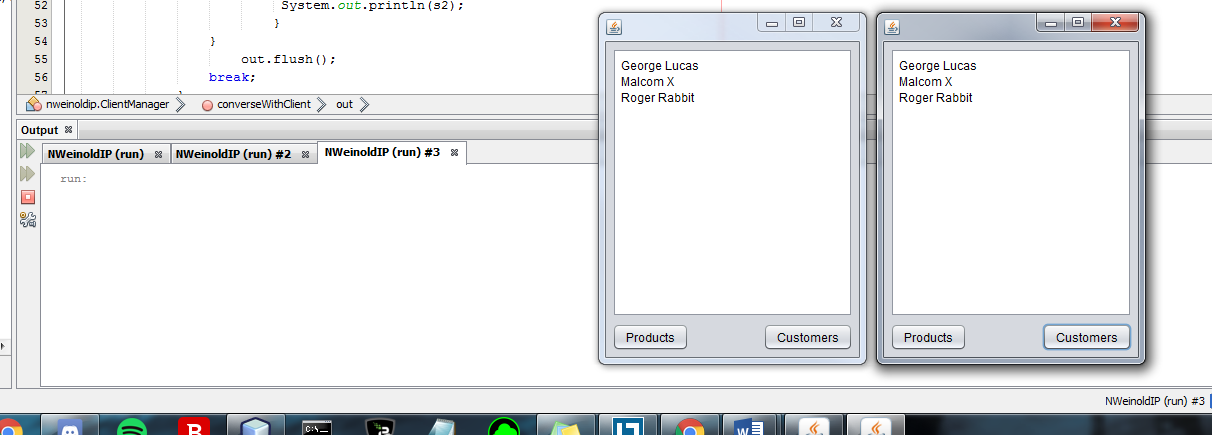
Two clients still connected first now requests customers second now requests products



Two clients connected BOTH request products



Two clients still connected BOTH request customers



As requested in this weeks discussion board and as discussed in our class lecture I have made my attempt at including a thread.sleep() to ensure proper communication on the client pool



# Unit 3 : Database-Driven Desktop Applications

This week I have had quite a few issues with getting the program to work and to get MySQL working on the computer that I am using. I have finally gotten everything functioning and stable. The MySQL server information is

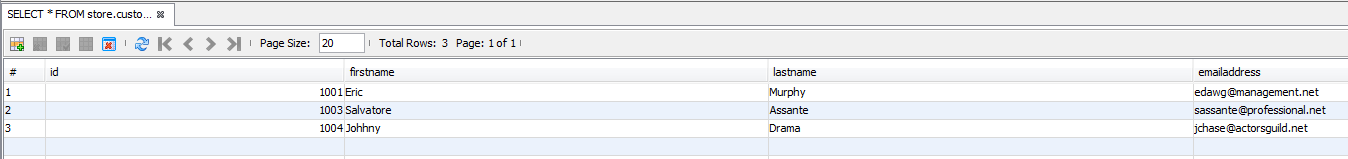
url = "jdbc:mysql://localhost:3306/store"

username = "ctuonline"

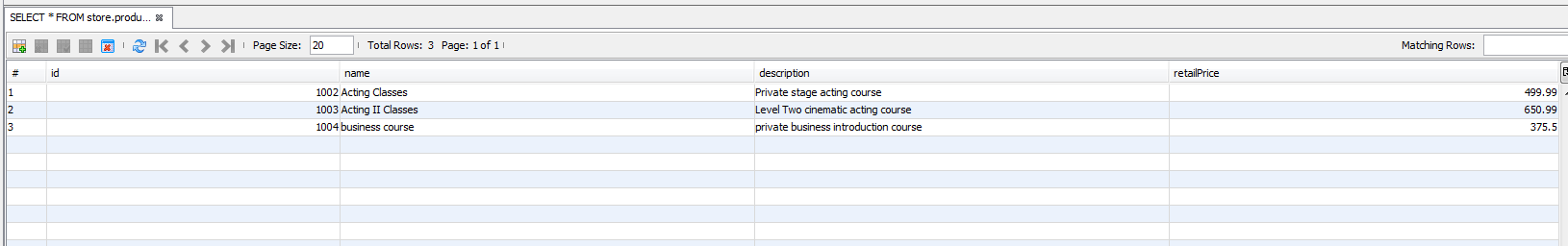
password = "student"

I have created a Store database with a Customer and a Product table

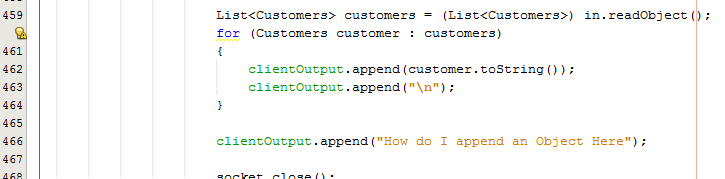
Customer holds id, firstname, lastname, and emailaddress



Product holds id, name, description, and retailPrice

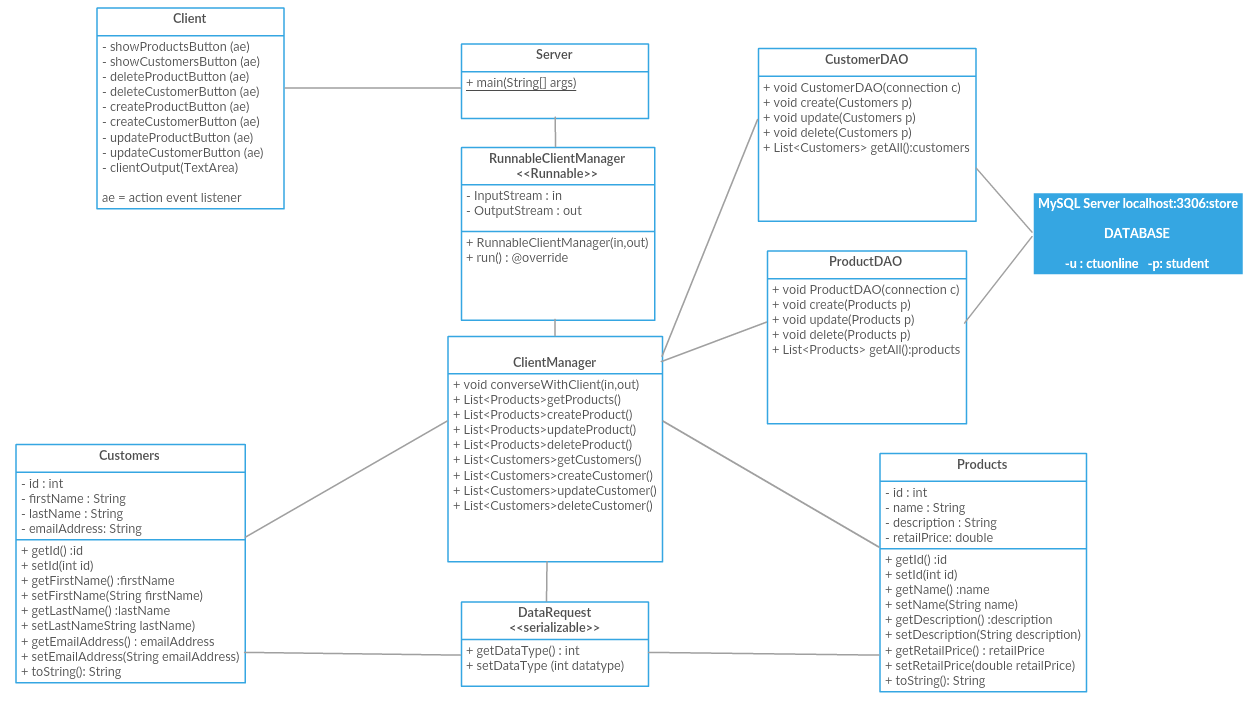


The one issue that remains is found within the Client GUI. I am able to communicate from the client > through the server > to the database > back to the server. For some reason I have not been able to receive the server data to the client text area and display that within the client GUI. This was working fine before but for some reason this bit of code is not collecting the servers response



I have tried many different ways and I am sure it is something simple that I am just missing. As a place holder I have had the program splash “How do I append an Object here” as a reference in the programs function. I assumed I am trying to display the array object Product/Customer getAll() but I cannot figure out a decent way to convert the object to a string vale to append the data to the text area. For now the server communicates all that is requested from the assignment details and all the buttons work/function fully as they are supposed to. I plan to add some more validation and maybe make the interface a little “prettier” before IP5 submission.

## UML Class Diagram



## Pseudocode

**Pseudocode for the server:**

Server START

Creates socket 83

Opens socket

Infinitely looks for a new client

IF client connects uses the runnableClientManager to handle the new client

IF no client connects continues to look for new clients until closed.

END

**Pseudocode for the clientManager:**

Get the data type (product or customer)

**If PRODUCT** call to getProducts()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Return ProductDAO getAll()

Flush

**IF CUSTOMER** call to getCustomer()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Return CustomerDao getAll()

Flush

**IF CREATECUSTOMER** call to createCustomer()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input from ID, First name, Last name, Email

Add input data using create(customers c)

Return CustomerDAO getAll()

**IF CREATEPRODUCT** call to createProduct()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input from ID, name, description, cost

Add input data using create(product p)

Return ProductDAO getAll()

**IF UPDATECUSTOMER** call to updateCustomer()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input from ID, First name, Last name, Email

Add input data using update(customers c)

Return CustomerDAO getAll()

**IF UPDATEPRODUCT** call to updateProduct()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input from ID, name, description, cost

Add input data using update(product p)

Return ProductDAO getAll()

**IF DELETECUSTOMER** call to deleteCustomer()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input selection from ID

Delete selected table row using delete(customers c)

Return CustomerDAO getAll()

**IF DELETEPRODUCT** call to deleteProduct()

Connect to MySQL server jdbc:mysql://localhost:3306/store as user ‘ctuonline’ password ‘student’

Read user input selection from ID

Delete selected table row using delete(product p)

Return ProductDAO getAll()

**Pseudocode for runnableClientManager:**

Run clientManager

Passes input stream or output stream from client to server or server to client using clientManager as a thread

**Client Pseudocode:**

START

Creates socket 83

Opens socket

Connects to server

User selects Product or Customer button

Sends request to server for product or customer data

Receives product or customer data and displays in textArea **(NOT WORKING YET DISPLAYS IN SERVER System out)**

User inputs data into the Product or Customer text fields and select Product Create or Customer Create

Sends request to server for createproduct or createcustomer

Server updates database schema with new user input data

User receives product or customer data and displays in textArea **(NOT WORKING YET DISPLAYS IN SERVER System out)**

User inputs id number into the id text field for product or customer and selects delete

Sends request to server for deleteproduct or deletecustomer

Server deletes selected row from database schema

User receives product or customer data and displays in textArea **(NOT WORKING YET DISPLAYS IN SERVER System out)**

User inputs data into the Product or Customer text fields and select Product update or Customer update

Sends request to server for updateproduct or updatecustomer

Server updates database schema with new user input data

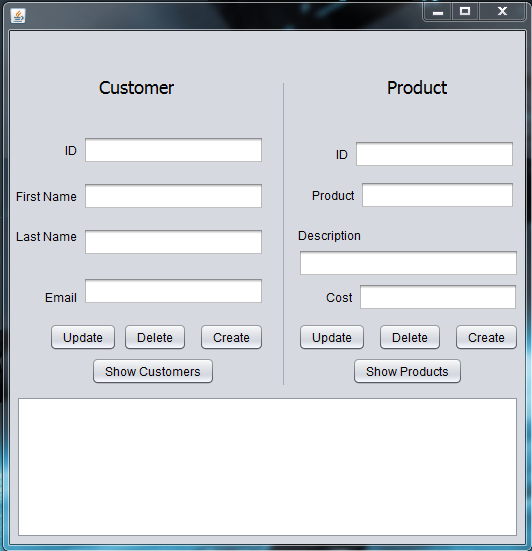
User receives product or customer data and displays in textArea **(NOT WORKING YET DISPLAYS IN SERVER System out)**

IF user closes window client socket closes

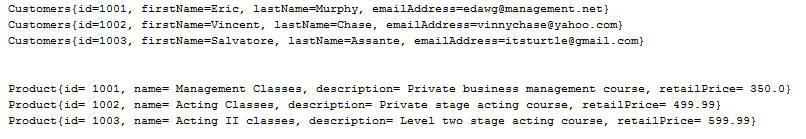
END

## Screenshots

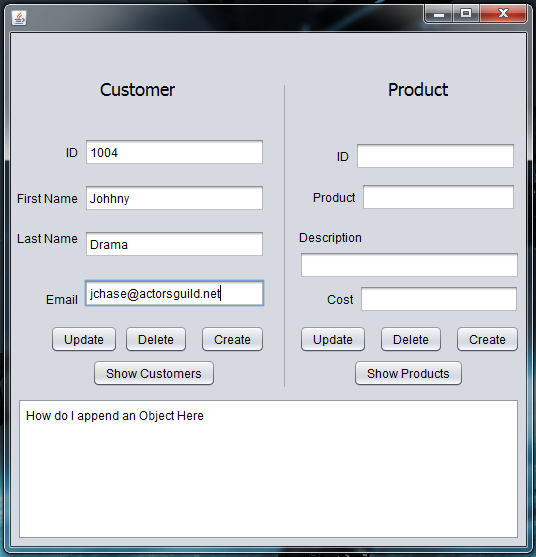
**Selecting Show Customers and Show Products**



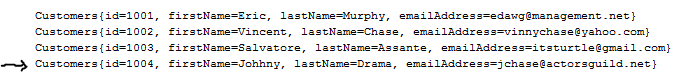
**This where my issue is. I can collect the data but it will not display in the client GUI instead it displays in the system on the server application**



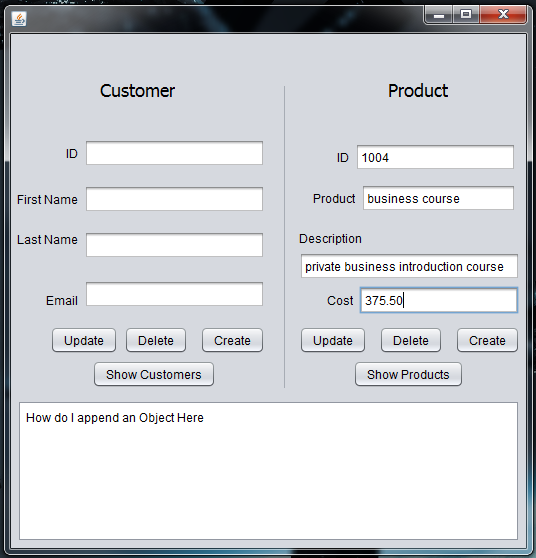
**Creating a new Customer 1004, Johhny, Drama,** [jChase@actorsguild.net](mailto:jChase@actorsguild.net)



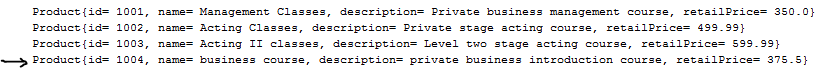
**^ I added the how do I append an object here as a placeholder to see where in the code I need to include the output for the textarea. Below you see johnny has been added.**



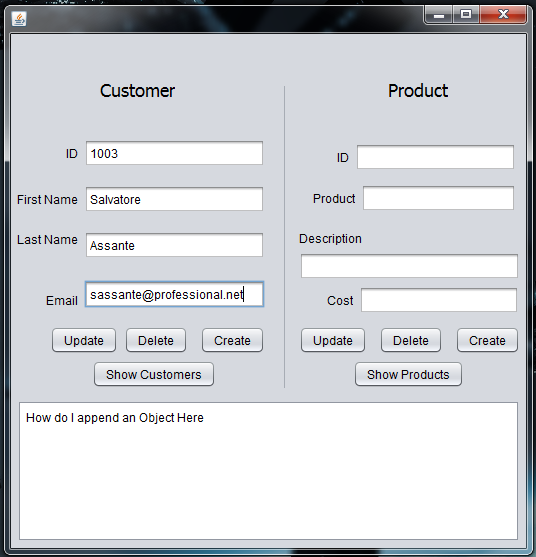
**Creating a new product 1004, business course, “private business introduction course”, 375.50**



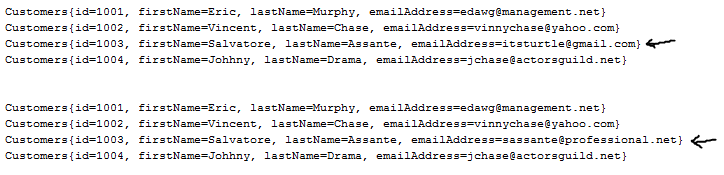
**Below you can see the new product has been added**



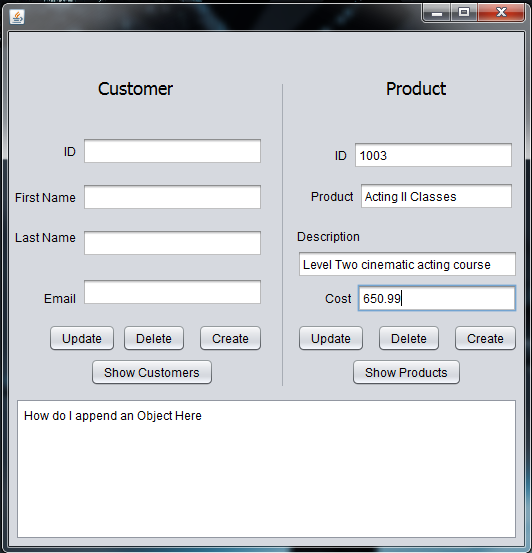
**Updating an existing customer : Salvatore Assante wants a more professional business email so he has input 1003, Salvatore, Assante,** [sassante@professional.net](mailto:sassante@professional.net)



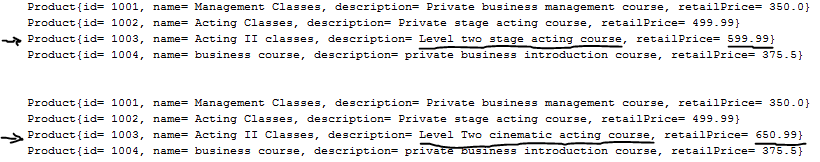
**Below you see the changes have been made**



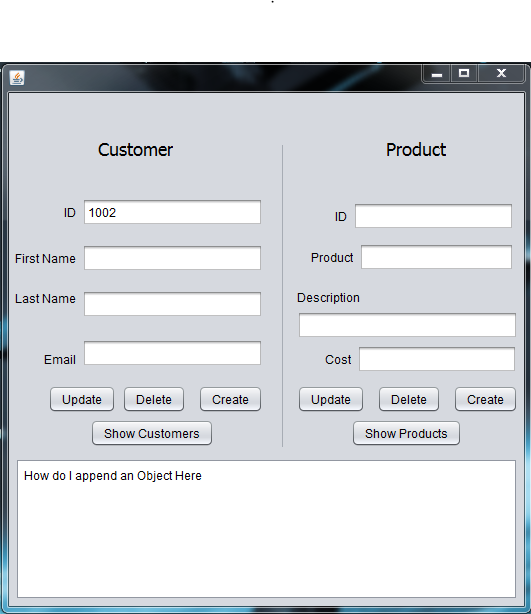
**Updating Products : The Acting two course will no longer be stage acting and instead is cinematic acting, along with the description the price will also change. 1003, Acting II Classes, Level two cinematic acting course, 650.99.**



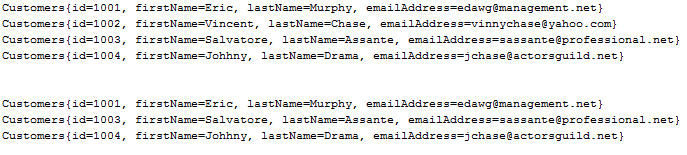
**Below you can see the course has been adjusted to reflect the new information and cost**



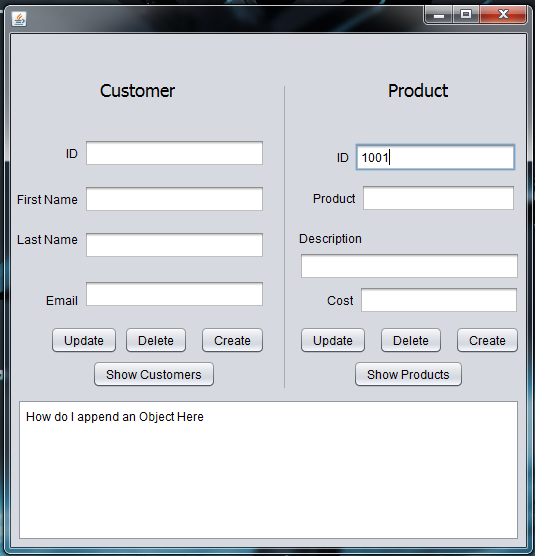
**Deleting a Customer : Vincent Chase no longer wants to take acting classes and is not going to be a customer any more so we can delete his information. We input the id 1002 and select delete.**



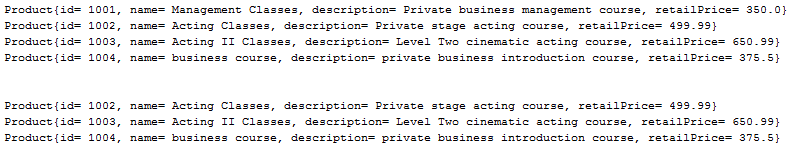
**Below you can see Vincent Chase is no longer in the database**



**Deleting a Product : It turns out no one is really taking the management classes so we are going to remove this product. We input the id 1001 and then select delete.**



**Below you can see the management course has been removed.**

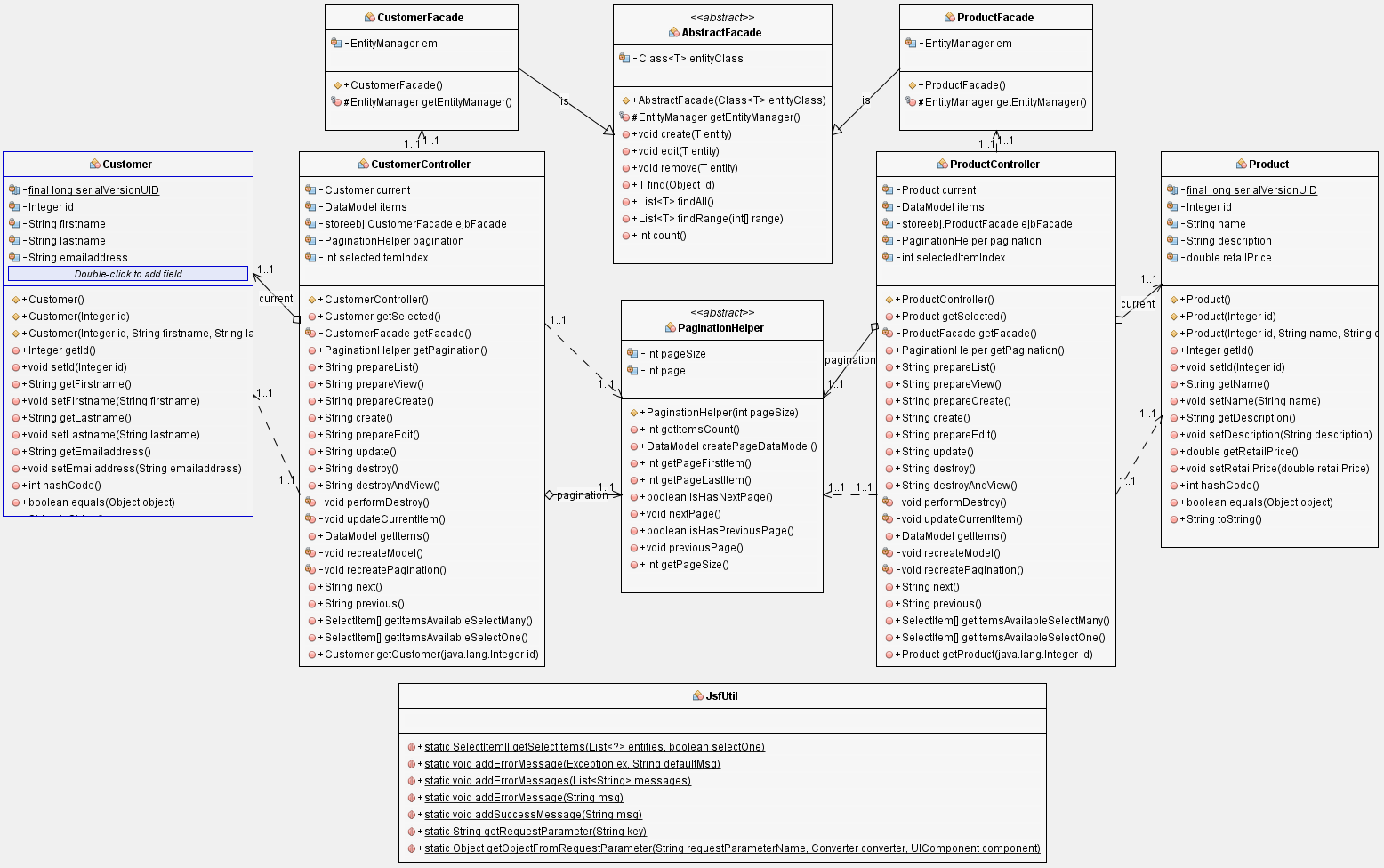


# Unit 4: Database-Driven Web Applications

This week we are changing our user interface to utilize a Java web application to communicate with our MySQL server. The application will be run using Java Servlet Pages (JSP). The JSP Web Application will connect to the localhost MySQL database using the 3306 port and the login credentials -u ‘ctuonline’ -p ‘student’. The user will have access to the Product database form and the Customer database form once connected to the web pages. The assignment includes The UML, pseudocode, application screenshots, NWIT351IP.sql, JSPWebApp Folder, and the user manual. The user manual will explain the installation and how to operate the application. The attached files will include the NWIT351IP.sql file which will be imported through the MySQL workbench (Explained in the Manual), as well as the JSPWebApp folder which will be added to the NetBeansProjects folder this will execute the Java files (explained in the manual).

## UML Class Diagram

\*\* Please zoom in to see all the class details \*\*



## Pseudocode

User runs JSP Web Application

Application connects to MySQL server Store database as user ctuonline

Web application receives store database table names

Web application displays web page interface in browser window

User selects “Show All Customer Items” or “Show All Product Items”

Web page displays Customer or Product table data

User selects “Create New Customer”

Web page displays user input text fields for table column data

User inputs data into text field and selects “save”

Web application sends data to the MySQL server and updates new content

User can select to “view” to see individual table data

Web application requests individual data and web page displays data results

User can select to “Edit” individual table data

Web application sends edit request for individual data and web page displays data results

User edits data and selects save

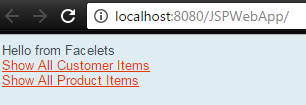
Web application sends data to the MySQL server and updates new content

User can select to “destroy/delete” individual table data

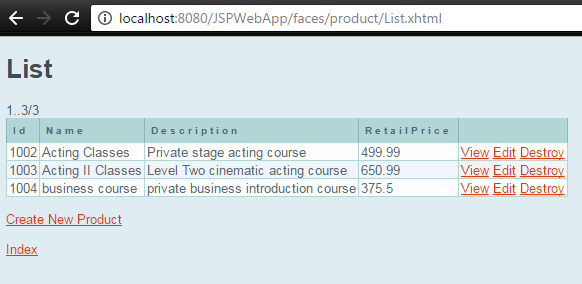
Web application sends delete request to the MySQL server and deletes selected table data

## Screenshots

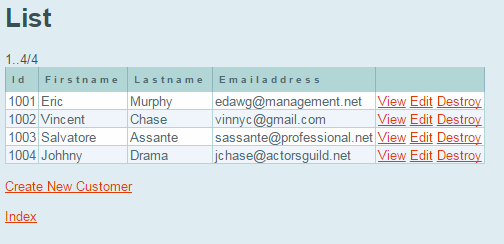
Main Index page



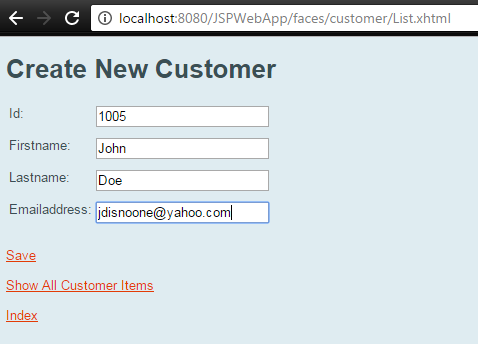
Product data page



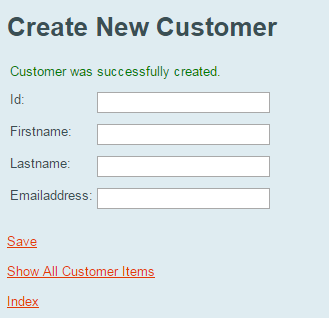
Customer data page



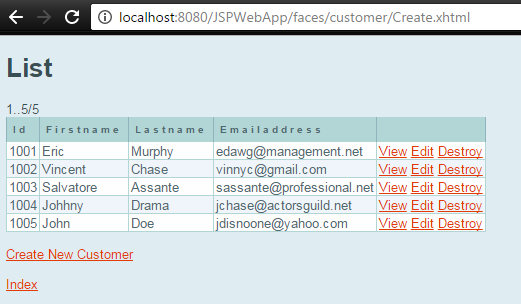
User selects Create new customer and enters new customer data



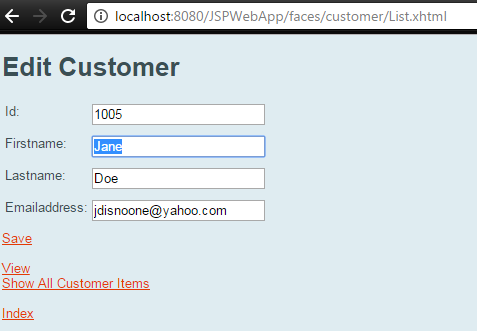
User selects save and save data is confirmed



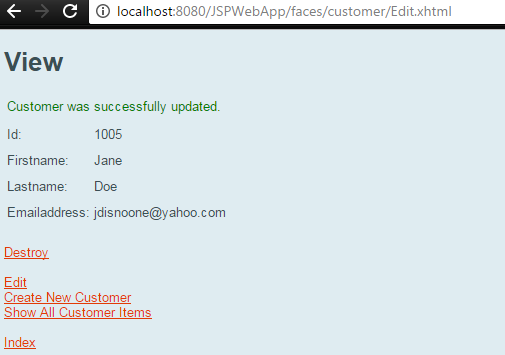
User selects show all customer items and new customer is included



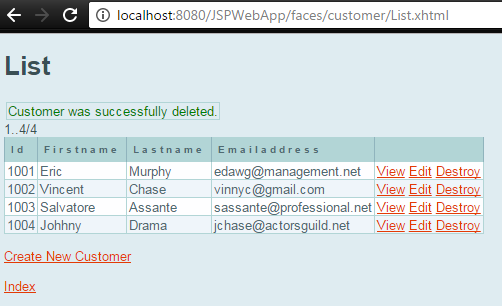
User selects edit customer and changes john to jane



User selects save and customer data is updated



User select destroy on the Jane Doe customer data and customer deletion is confirmed



## User Installation/Operation Manual

First you must ensure you have NetBeans and MySQL workbench installed which you can get here:

<https://netbeans.org/downloads/>

<https://dev.mysql.com/downloads/installer/>

Select your OS platform and install the current versions for your system.

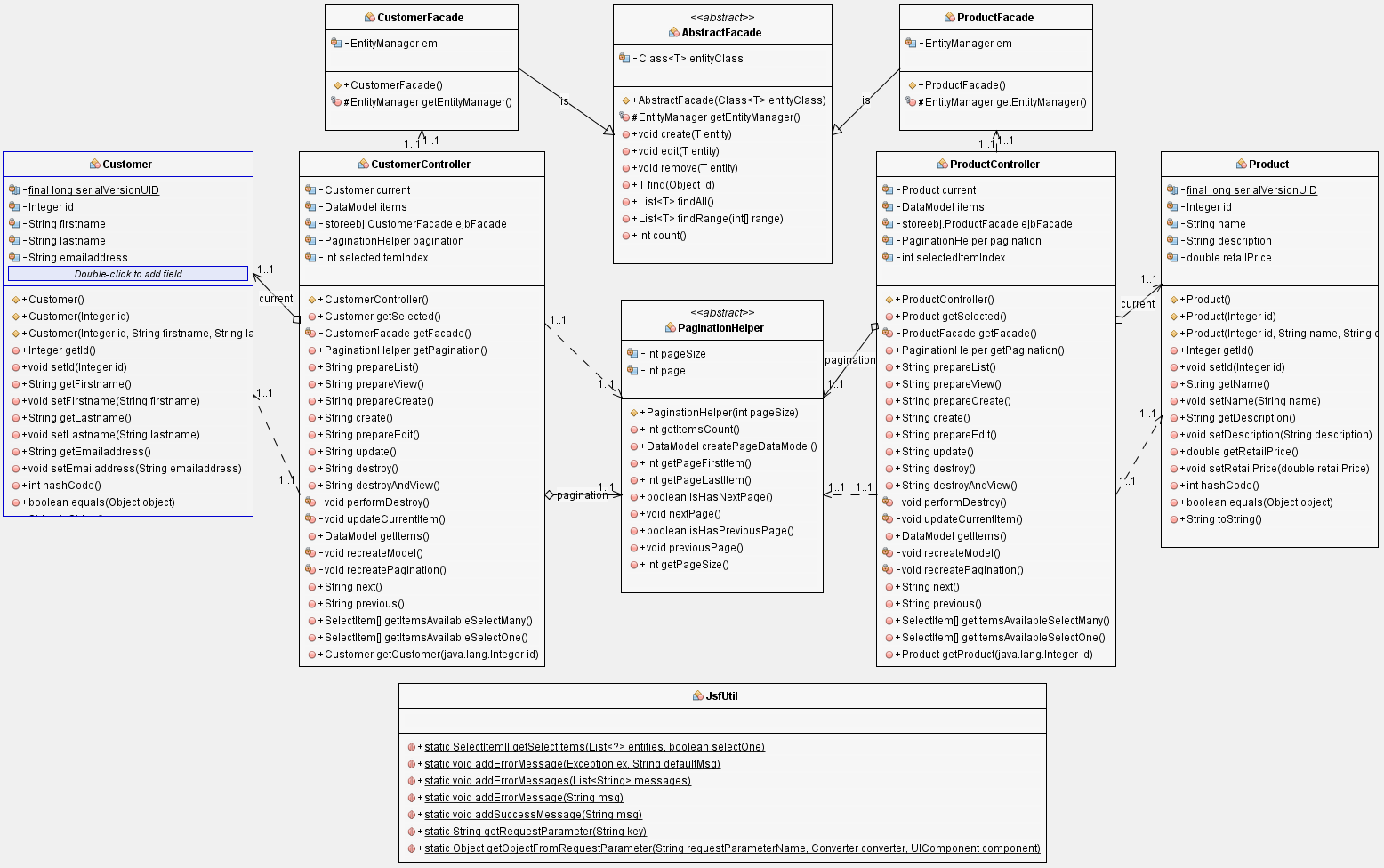
Next follow these steps to install the new Store database and the JSPWebApp

1. Download NWIT351IP.sql file and JSPWebApp.zip Folder
2. Open the MySQL Workbench and create a new localhost server using port 3306
3. Connect to new server using your root login
4. Under Navigator > Management select ‘User and Privileges’
5. At the bottom left select ‘Add Account’
6. Set Login Name: ctuonline, Set Limit to hosts matching: ‘localhost’, and Password: student
7. Select ‘Apply’
8. Under Navigator > Management select ‘Data Import/Restore’
9. Select radio button ‘Import from Self-Contained File
10. Set path to C:\Users\CorruptFile\Downloads\ NWIT351IP.sql
11. Leave Default Target Schema blank and select ‘New…’
12. Under Schemas you now have the Store Database which contains the product and customer tables
13. Unzip JSPWebApp.zip and save the folder in to your C:\Users\User\Documents\NetBeansProjects
14. Run NetBeans
15. Open up the Services tab and right click and select ‘New Connection’
16. Under Driver select ‘MySQL(Connector/J driver)’ and select ‘Next’
17. Set Host to ‘localhost’, Set Port to ‘3306’, Set Database to ‘store’, Set User Name to ‘ctuonline’, Set Password to ‘student’, and select ‘Next’
18. Select ‘Finish’
19. Under Services tab > Databases right click the new database connection and select connect
20. You will now see store
21. Go to the Projects tab and right click the JSPWebApp and select ‘Run’
22. After a few seconds the connection will be established a web page will pop up
23. Navigate through the web application using the displayed buttons
24. Close the web page window to exit the program

Completed you now have access to the new Web Application that communicates and displays the MySQL store database!

# Unit 5: Database-Driven Web Services

## UML Class Diagram



## Pseudocode

User runs JSP Web Application

Application connects to MySQL server Store database as user ctuonline

Web application receives store database table names

Web application displays web page interface in browser window

User selects “Show All Customer Items” or “Show All Product Items”

Web page displays Customer or Product table data

User selects “Create New Customer”

Web page displays user input text fields for table column data

User inputs data into text field and selects “save”

Web application sends data to the MySQL server and updates new content

User can select to “view” to see individual table data

Web application requests individual data and web page displays data results

User can select to “Edit” individual table data

Web application sends edit request for individual data and web page displays data results

User edits data and selects save

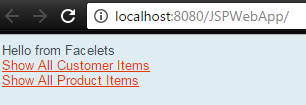
Web application sends data to the MySQL server and updates new content

User can select to “destroy/delete” individual table data

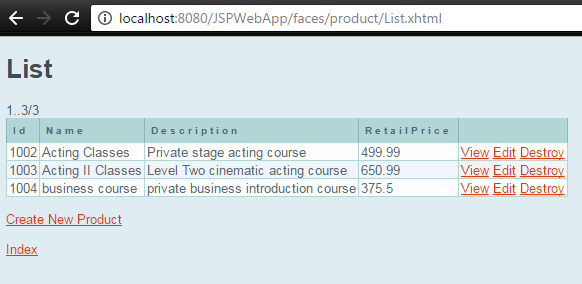
Web application sends delete request to the MySQL server and deletes selected table data

## Screenshots

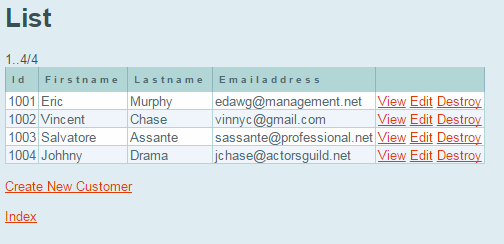
Main Index page



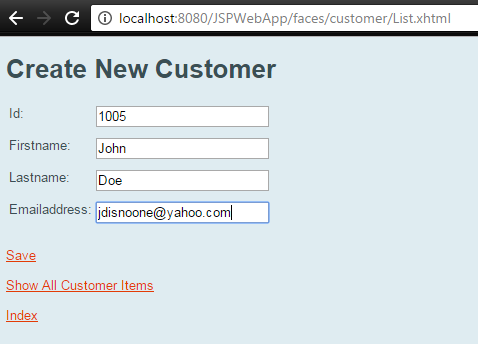
Product data page



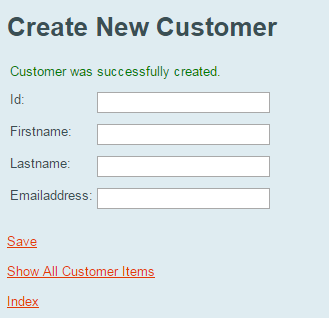
Customer data page



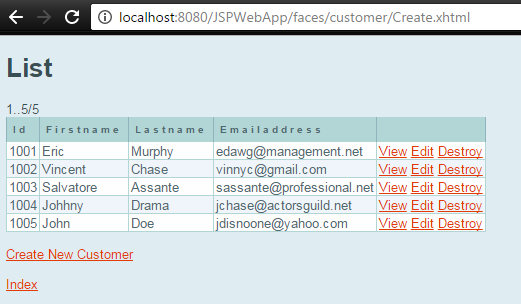
User selects Create new customer and enters new customer data



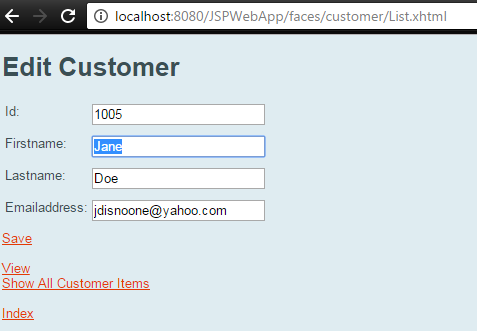
User selects save and save data is confirmed



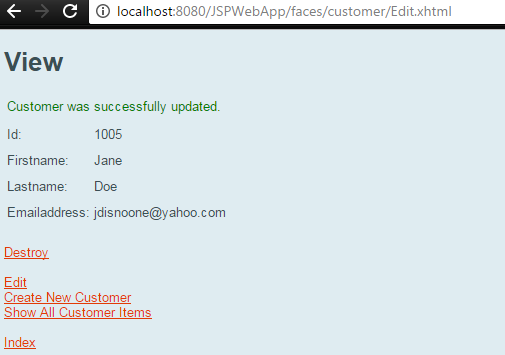
User selects show all customer items and new customer is included



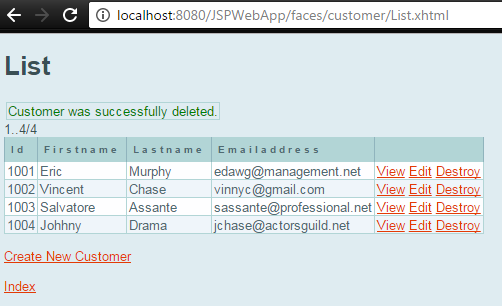
User selects edit customer and changes john to jane



User selects save and customer data is updated



User select destroy on the Jane Doe customer data and customer deletion is confirmed



# Questions

**Java Technologies that were used within this program –**

GlassFish Server 4.1.1 was used to establish the web page connection to the database within NetBeans. GlassFish was used because of the type of web page it displays and the minimal amount of work that needed to be done to create a functioning web program connection in comparison to TomCat which is simple HTTP server using servlets, whereas GlassFish is a complete Java EE application server that includes and EJB container and easily implements hibernate. Glass fish is a bit larger in size but for how simple our program is this was the best choice for end user experience. I could have also gone with JBoss which is a Java EE application server as well that utilizes an EJB container also. I did not need to use something as advanced as WebLogic for this assignment because of the minimal needs that the assignment requirements had, WebLogic would have included more tools and interface options than needed. To clarify the persistence of the server data is made possible through component communication from the web service application to the MySQL server that is persistently running.

**Future Development Plan –**

Multithreading the server would be done a few different ways. One way that would be very easy is to use hibernate to set up a persistence framework and within a class object for each client I could set up a timer that leaves windows of communication. This type of addition to the web program would need to be implemented using Spring to allow declarative transactions from the client set on timers.

Another way would be to using a timer service again but this time it would set a task queue from each client connected and a first come first served approach would take place. Each user would have this as a background task running and it would not be limited to any number of client connections.

Finally we could create a separate thread for each user using the EJB method invocation Asynchronous, I do not know this one very well but from what I have read you can inject ManagedExecutorService to execute a Runnable with the submit method and from the runnable you can manage the timer if needed but alone this should allow each user to send data through individual thread “commands”.

**Reflection Questions**

**Explain how web services are used in the real world. Provide at least 3 real world examples.**

Some real-world examples of how SOAP and REST are used within the real world are Web stores, mobile applications, and bank transaction services. Specifically SOAP utilizes XML and is a bit more complex in use. In the real-world SOAP based web services will be full websites and may be seen more in sales sites and banking sites. There is a little more security in XML if properly coded due to legacy implementation and formal contracting between the application and a user. An example of a SOAP based real world application would be PayPal and Docusign. Rest is a little less complex and uses JSON which means it’s a much lighter web service interface and now with better understanding of SQL injection methods REST is a very secure web service approach. Some real world examples beyond most mobile applications would be Amazon and Google which have both recently made moves towards REST from SOAP.

**What are the benefits of web services?**

From everything I have read today the biggest benefit over http protocol is that web services like SOAP use less bandwidth and REST uses even less. This is a huge focus for modern web development because of data usage metrics that cause user and owner costs on data charges. Also, this allows for more users and less server cost in performance needs. Another key feature is that they allow communication on different platforms an example would be .NET communications with a Java web service.

**How would you troubleshoot and test a web service? Would you use a debugger? Why or why not?**

Troubleshoot and testing would require using special request or print outs from the web service server. The best testing would be done using a test plan and letting a 3rd party review the processes. This is why there are alpha and beta launches to software and websites. You can use a debugger or an Ide like IntelliJ Idea or Eclipse to load a program and run a remote debugging tool. Sometimes just manual use of the program and then testing that function within the debugging tool can be an effective process if you have time within your project.

**Do you think that your client/server application could be enhanced to consume a web service or be used as a web service? Why or why not?**

Yes, I think that it can because as mentioned above using the managedExecutorService and a submit method you can execute a runnable which would allow our class program to used because we used a runnable client manager that could pass the communications. It would be a little bit like a launcher communication when you run a game through a game launcher interface. I am not sure it would be very useful for many other uses other than this type of launcher to program launch type of communication I think the user to server communication would be done within the program or the web service interface not back and forth consistently between both.

**Discuss the Java EE technologies you used and the reasons for your choice. Explain how these technologies achieved data persistence.**

Again, as mentioned above the GlassFish using the EJB components and communication with the MySQL server. The MySQL server provides a persistence for the GlassFish web service to communicate with using client or root login information that is passed.

# Test Plan

|  |  |
| --- | --- |
| User Action | Expected Result |
| User follows installation manual below | Program is running and user is on web page main index page |
| User selects “Show all customer items” | Web page changes to customer items page and displays current data within the customer table |
| User selects “create new customer” | Create new Customer input fields are displayed |
| User inputs new customer data and selects “save” | Customer data is updated to the server database and “customer was successfully created” is displayed and text fields are cleared |
| User selects “Edit” on customer data | Edit customer page is displayed and existing data can be changed |
| User selects “save” on Edit page | Customer data is updated on the server database and “Customer was successfully updated” is displayed |
| User selects “index” | Returns user to main index page |
| User selects “Show all product items” | Web page changes to product items page and displays current data within product table |
| User selects “create new product” | Create new product input fields are displayed |
| User inputs new product data and selects “save” | product data is updated to the server database and “product was successfully created” is displayed and text fields are cleared |
| User selects “Edit” on product data | Edit product page is displayed and existing data can be changed |
| User selects “save” on Edit page | product data is updated on the server database and “product was successfully updated” is displayed |
| User selects “index” | Returns user to main index page |
| User closes web page | Client connection is closed |

## User Installation/Operation Manual

First you must ensure you have NetBeans and MySQL workbench installed which you can get here:

<https://netbeans.org/downloads/>

<https://dev.mysql.com/downloads/installer/>

Select your OS platform and install the current versions for your system.

Next follow these steps to install the new Store database and the JSPWebApp

1. Download NWIT351IP.sql file and JSPWebApp.zip Folder
2. Open the MySQL Workbench and create a new localhost server using port 3306
3. Connect to new server using your root login
4. Under Navigator > Management select ‘User and Privileges’
5. At the bottom left select ‘Add Account’
6. Set Login Name: ctuonline, Set Limit to hosts matching: ‘localhost’, and Password: student
7. Select ‘Apply’
8. Under Navigator > Management select ‘Data Import/Restore’
9. Select radio button ‘Import from Self-Contained File
10. Set path to C:\Users\CorruptFile\Downloads\ NWIT351IP.sql
11. Leave Default Target Schema blank and select ‘New…’
12. Under Schemas you now have the Store Database which contains the product and customer tables
13. Unzip JSPWebApp.zip and save the folder in to your C:\Users\User\Documents\NetBeansProjects
14. Run NetBeans
15. Open up the Services tab and right click and select ‘New Connection’
16. Under Driver select ‘MySQL(Connector/J driver)’ and select ‘Next’
17. Set Host to ‘localhost’, Set Port to ‘3306’, Set Database to ‘store’, Set User Name to ‘ctuonline’, Set Password to ‘student’, and select ‘Next’
18. Select ‘Finish’
19. Under Services tab > Databases right click the new database connection and select connect
20. You will now see store
21. Go to the Projects tab and right click the JSPWebApp and select ‘Run’
22. After a few seconds the connection will be established a web page will pop up
23. Navigate through the web application using the displayed buttons
24. Close the web page window to exit the program

Completed you now have access to the new Web Application that communicates and displays the MySQL store database!

# References

Lowe, A. (2016) *“IT351 Phase 1” CTU Java YouTube Channel, videos 1-9*. Retrieved from <https://www.youtube.com/playlist?list=PLfkYWLGFGjwHkG-xTX_5gubwIf6S-wFRJ>

Angela Griffin (2017) “IT351 week 5 Different Protocols and Markup Languages” Live chat Retrieved from <http://ctuadobeconnect.careeredonline.com/p51hq47k149/?launcher=false&fcsContent=true&pbMode=normal>