PROJECT REPORT

CSCI-531 Advanced Programming with Java

(Order Management Application)

By

Mohammed Abdul Qader	**50145529**
Shariqh Ahmed Shariff	**50144479**
Mohammed Abdul Aziz Syed	**50143871**

Professor

Dr. Ray Maleh (Ph.D.)

(Department of Computer Science and Information System,

Texas A & M University –Commerce)

1. Introduction

Order Management is the core of any business these days.

It is very important for applications to access/update a centralized Inventory/Database, where product information is stored.
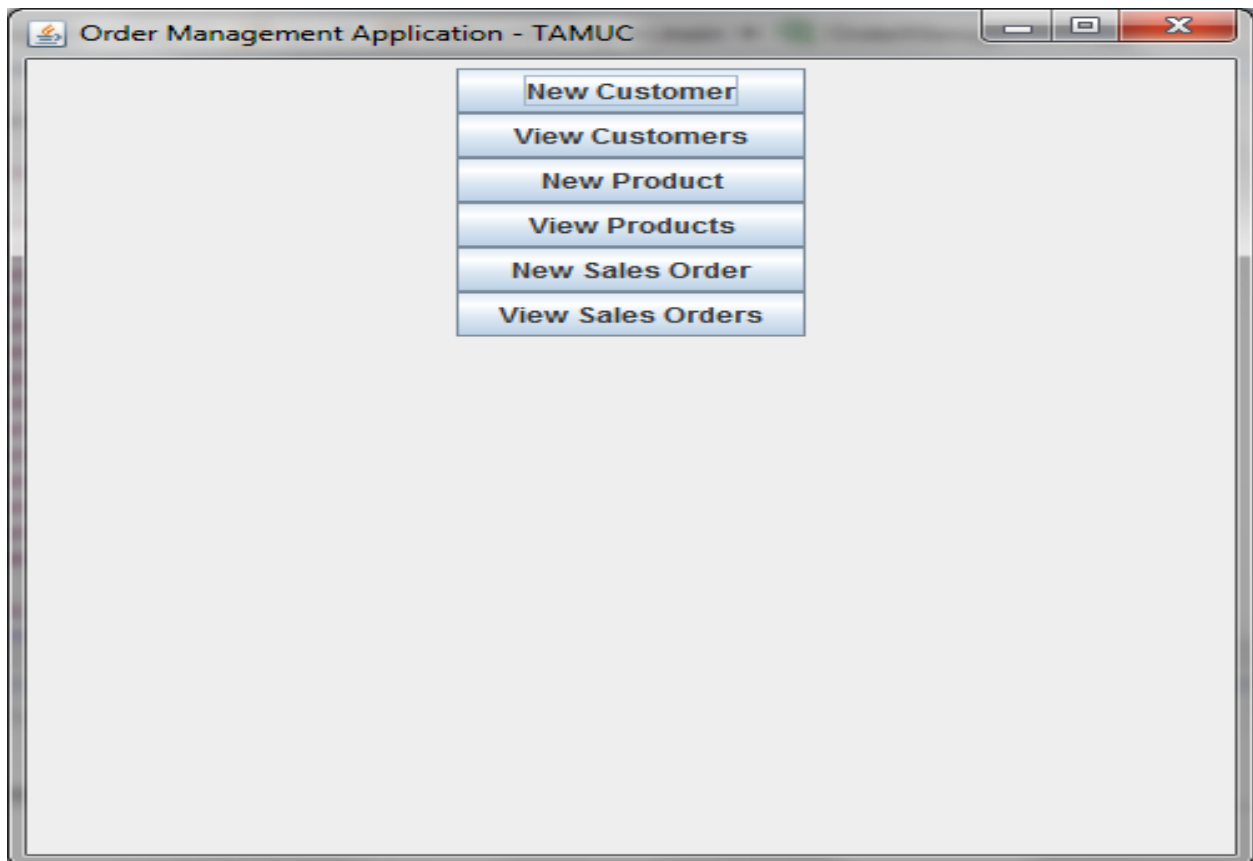
This project is aimed to serve the same purpose.

Key Modules:

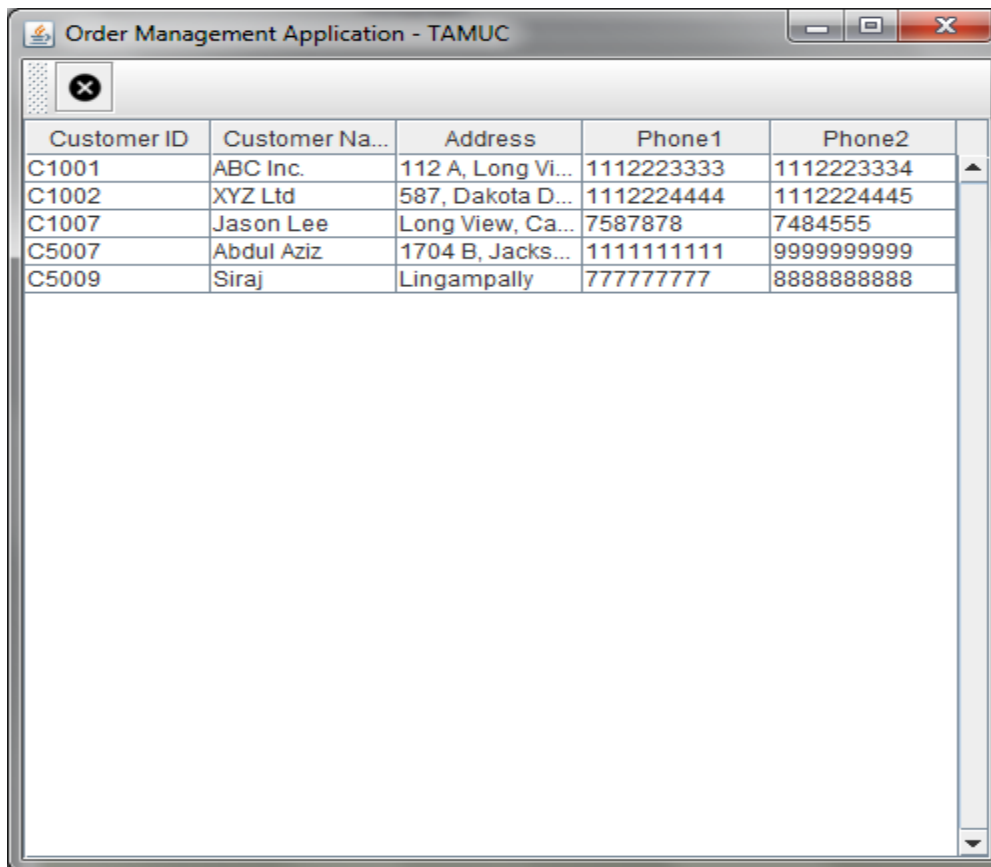 a. Customer Module

 b. Product Module

 c. Sales Order Module

All these modules interact with a central data repository, which is a MySQL database named retail_db

2. User Guide
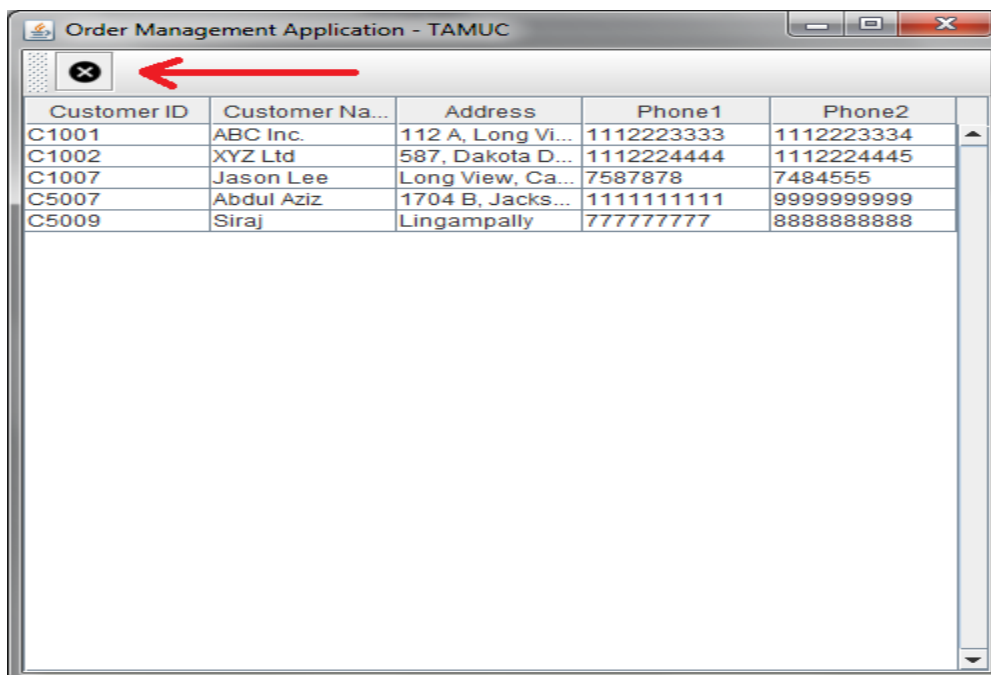

This is the Home Screen of Order Management Application

- View Existing Customers – Click "View Customers"

- Closing current screen takes back to Home Screen

- Adding a new Customer – Click "New Customer"
- Enter the details, hit "Save"

- Update Customer Details:
  - View Customer's
  - Select the customer record to be updated

- Change the address and hit "Save" again.
- It updates the existing Customer information, as shown below

- Deleting an existing customer
  - View Customer's
  - Select the customer to be deleted; Hit "Delete" button
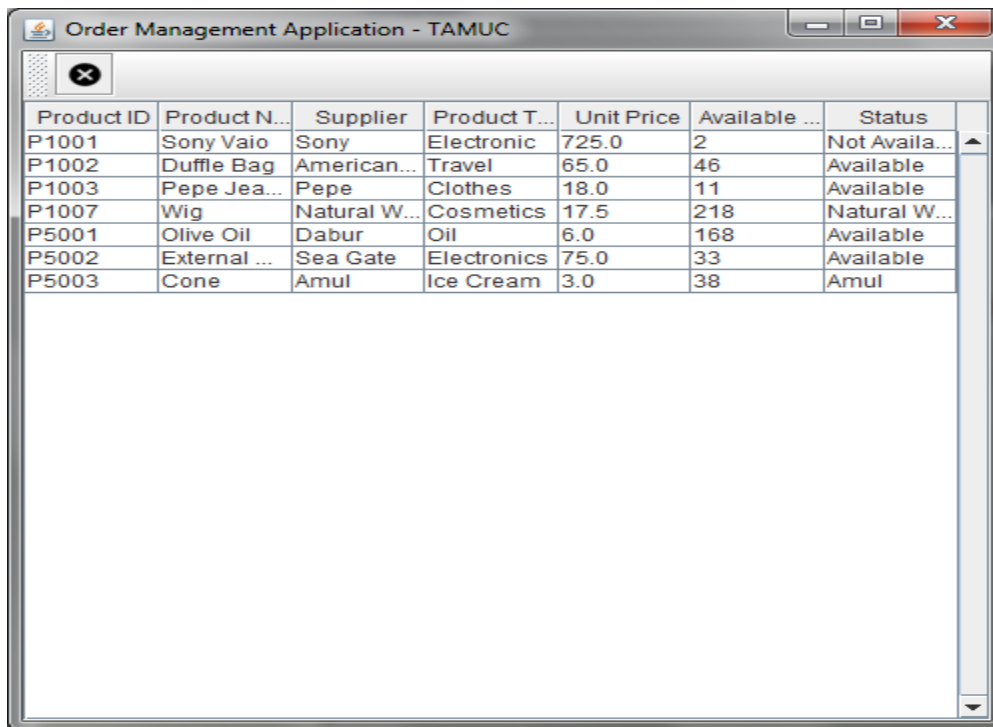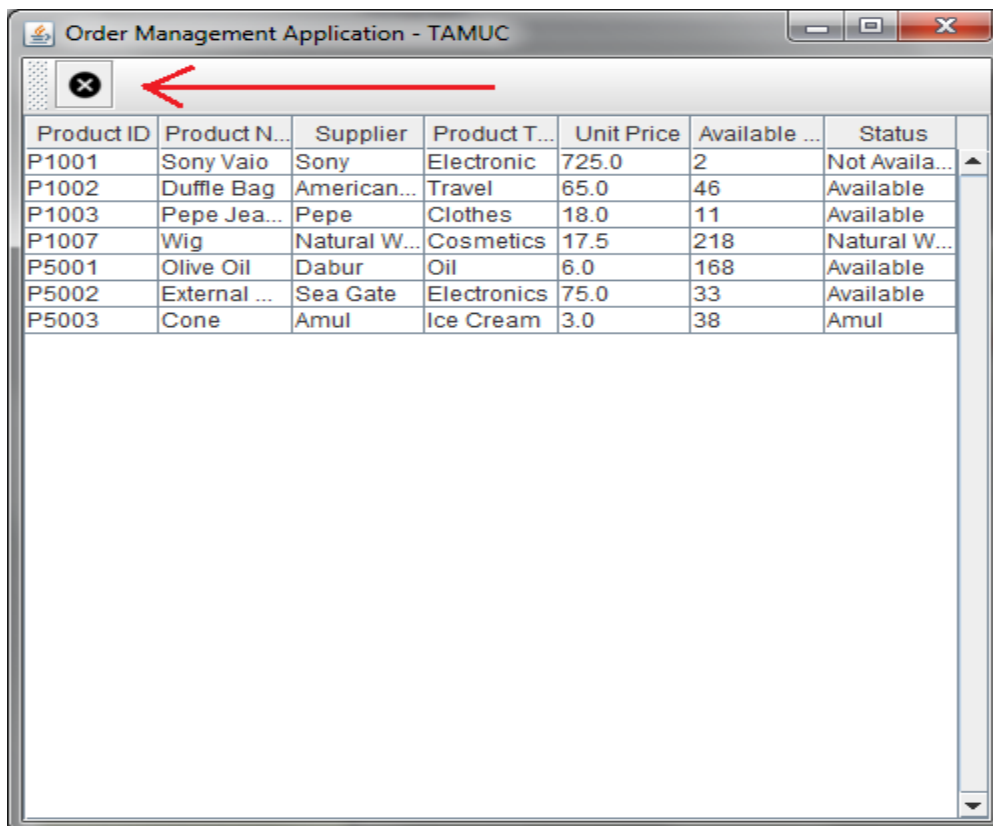
- View Existing Products – Click "View Products"

- Closing current screen takes back to Home Screen

- Adding a new Product – Click "New Product"
- Enter the details, hit "Save"

- Update Product Details:
  - View Product's
  - Select the product record to be updated

- Change the available quantity and hit "Save" again.

- It updates the existing Product information, as shown below

- Deleting an existing product

  - View Product's

  - Select the product to be deleted; Hit "Delete" button

■ View Existing Sales Orders – Click "View Sales Orders"

Order Management Application - TAMUC

Order ID	Customer ID	Total Price
SO1002	C1002	992.0
SO1003	C1001	101.0
SO1004	C5007	87.0
SO1005	C5009	746.0
SO1006	C5009	1551.0
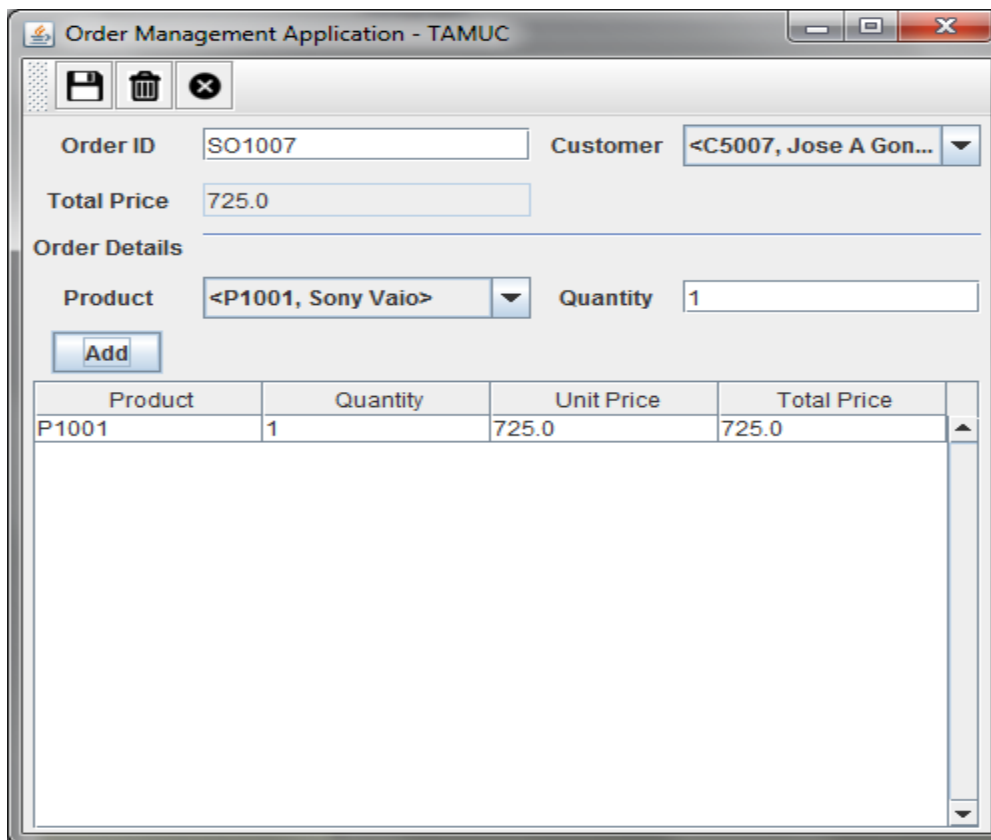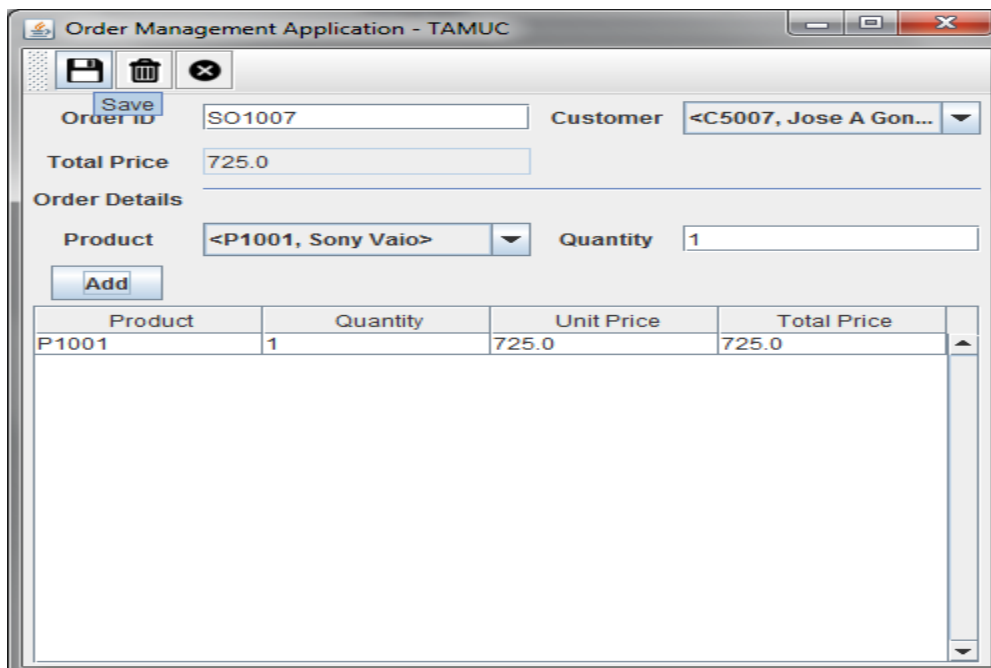
■ Closing current screen takes back to Home Screen

Order Management Application - TAMUC

Close der ID	Customer ID	Total Price
SO1002	C1002	992.0
SO1003	C1001	101.0
SO1004	C5007	87.0
SO1005	C5009	746.0
SO1006	C5009	1551.0

- Adding a new Sales Order – Click "New Sales Order"

- Enter the Order ID

- Select the Customer

- Select the Product to be ordered

- Enter the quantity

- Add item to the cart/order

- Hit "Save" to confirm the order

- View submitted Sales Order (Read Only)
  - View Sales Order's
  - Select the sales order record to be viewed

Left window:

Order ID	Customer ID	Total Price
SO1002	C1002	992.0
SO1003	C1001	101.0
SO1004	C5007	87.0
SO1005	C5009	746.0
SO1006	C5009	1551.0
SO1007	C5007	116.0

Right window:

Order ID SO1007   Customer <C1001, ABC Inc.>

Total Price 116.0

Order Details

Product <P1001, Sony Vaio>   Quantity

Add

Product	Quantity	Unit Price	Total Price
P1002	1	65.0	65.0
P1003	2	18.0	36.0
P5702	2	7.5	15.0

- Deleting an existing sales order
  - View Sales Order's
  - Select the sales order to be deleted; Hit "Delete" button

Left window:

Order ID	Customer ID	Total Price
SO1002	C1002	992.0
SO1003	C1001	101.0
SO1004	C5007	87.0
SO1005	C5009	746.0
SO1006	C5009	1551.0
SO1007	C5007	116.0

Right window:

Delete

Order ID SO1007   Customer <C1001, ABC Inc.>

Total Price 116.0

Order Details

Product <P1001, Sony Vaio>   Quantity

Add

Product	Quantity	Unit Price	Total Price
P1002	1	65.0	65.0
P1003	2	18.0	36.0
P5702	2	7.5	15.0

3. Technical Description

Following technologies are used in each layer, the detailed architecture is described in further slides

a. Presentation is done using Java Swings

b. Business Logic is written in Java

c. Data is stored in MySQL Database

Database Tables v/s Java Entities

- Following tables are sufficient to capture and store data belonging to all the modules of the application
  - Product
  - Customer
  - SalesOrder
  - OrderItem
- The corresponding entities mapped in Java are present com.order.mgmt.entities package
  - ProductBean
  - CustomerBean
  - SalesOrderBean
  - OrderItemBean

Database Access – MySQL

- All the Database activities have to go through the centralized connection, which is provided by following class
  - **com.order.mgmt.db.DataBase**
- This class cannot be instantiated. This is achieved by restricting the class constructor to be private

- However, the Database connection is returned via static getConnection() method
- This method takes care of loading DB driver, establishing the connection and setting transaction's auto commit to true

DAO Design Pattern to perform all Database Operations
- We followed Data Access Object (DAO) design pattern to perform all database activities – insert, update, delete, select
- DAO's for each of the available entities are listed below
    - ProductDAO
    - CustomerDAO
    - SalesOrderDAO
    - OrderItemDAO
- These DAOs are accessed via respective forms for each module by a helper/utility class named OrderManagementHelper

Utility Classes
- There are two utility classes used in our application
- They are packaged in **com.order.mgmt.utils**

1. IOrderManagement
    - Contains all named constants/literals

2. OrderManagementHelper
    - Provides functionalities common to each module, like access to corresponding DAO objects to insert/select/update/delete entries from database
    - Also, interconversion of form field values to respective objects and vice versa

User Forms for Data capture

- Forms are the standard way through which users can enter/modify the data
- All the classes are packaged in **com.order.mgmt.gui.forms**
- Following forms are provided, corresponding to each of the module
    - ProductForm
    - CustomerForm
    - SalesOrderForm
- All these forms inherit from a class called FormEditor, which is inturn Jpanel. It contains functionality common to all the forms like, Save, Delete, Close button etc.
- Each of the Product, Customer and Sales Order form have respective fields and labels
- The form labels and fields are grouped together in Grid Bag Layout Format
- GridBagConstraints class is used to beautify and constrain the components that are laid out on the GUI
- SalesOrderForm is a bit advanced screen, which uses JComboBox to list all available Customers and Products
- DefaultTableModel is used in background to save the order, in memory

DTO Design Pattern

- Data Transfer Object (DTO) design pattern is used to bind the data entered by users via respective form's
- The same DTOs are used to unbind the data when fetched from database and present to the user by populating appropriate fields on screen
- All DTOs are grouped in **com.order.mgmt.dto** package
- Each Entity has corresponding DTO as listed below
    - ProductDTO

- CustomerDTO
- SalesOrderDTO
- OrderItemDTO

Action Listeners
- The only way to respond to user actions is via Action Listeners
- All the action listeners required for the application are packaged in **com.order.mgmt.listeners**
- Following are few of them
    - CloseAction – to close current screen and redirect user to home screen
    - AddOrderAction – add the selected order item to table
    - ProductAction – redirect user to ProductForm
    - CustomerAction – redirect user to CustomerForm
    - SalesOrderAction – redirect user to SalesOrderForm

4. Conclusion

What we have learnt:
- Building an application end to end, from scratch
- This involved how to think about entities in real world, mapping the data captured from application screens till the point it is persisted in a Database
- Various Design Patterns like DAO, DTO etc.
- How layered architecture works and interactions take place
- Integrating modules from across the tiers

What could have done better?
- We could have added more modules
- The application would have been more interesting if we had some sort

of Analytics features like displaying Bar Graphs indicating products sold, quarterly/yearly sales etc..

Deployment Instructions:

1. Install **MySQL** database

2. Login to **MySQL** database, and execute the contents present in **retail_db.sql** script (This file is present in **OrderManagementApp\sql** folder)

3. Unzip **OrderManagementApp.zip** folder

4. Open Eclipse, go to File->Import
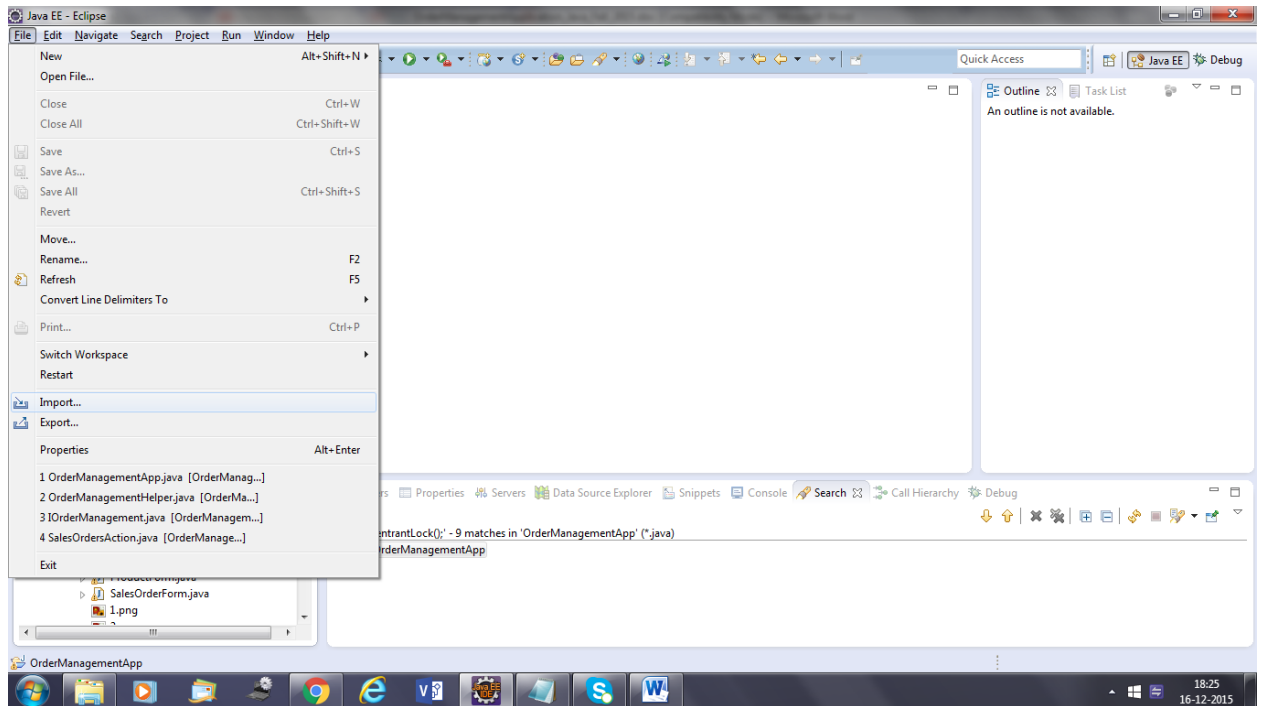
5. Select Existing Projects into Workspace
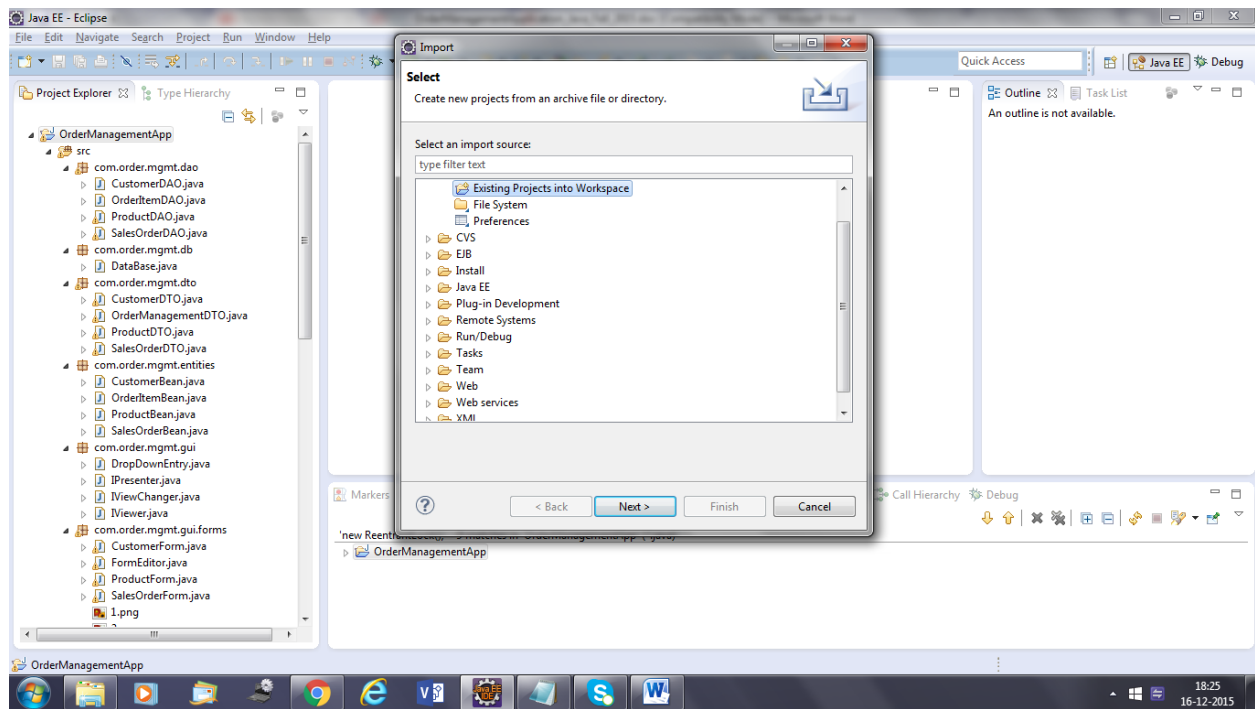
6. Click on Next, browse the directory in which the unzipped folder **OrderManagementApp** is present

7. Open the main class `OrderManagementApp` and run it by "CTRL + F11"