BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		Surname	Given Names	Student ID Number
	1.	Jusman	Christy Natalia	2301890365

Course Code	**: COMP6510**	**Course Name**	**: Programming Languages**
Class	**: L2AC**	**Name of Lecturer(s)**	**: Jude Joseph Lamug Martinez**
Major	**: Computer Science**		
Title of Assignment (if any)	: Warehouse Database Program		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 20-06-2020**	**Submission Date**	**: 20-6-2020**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.

2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Christy Natalia Jusman

"Warehouse Database Program"
Name : Christy Natalia Jusman
ID : 2301890365

A. Project Specifications

1. Project Description

As the final project, I have made a warehouse database program. This program is using database so every data that user input, delete or update will be saved. This program will be very useful for a warehouse that want to control the number of products on their inventory without making any human errors on calculating.

This project is using MySQLWorkbench to hold and mantain the data. By using MySQLWorkbench, it will be easier for me to control the data inside the database. Then, I've installed connection.jar from mysql in this java program to connect the program with MySQLWorkbench.

B. Solution Design

I'm using JAVAFX on designing this program. This javafx feature can create a GUI that displays objects that convey information, and represent actions that can be taken by the user. For the action, It will connect the control objects with the action in the controller page.

The design for this program is very simple. Users only have to click on the button that they want to use and input the data that they want. Unfortunately, this program is case sensitive (lowercase, uppercase, and any symbols). So, user have to be careful when input some data or an alert message will appear. For the color of this program is the combination of black and white which is the default color. The text

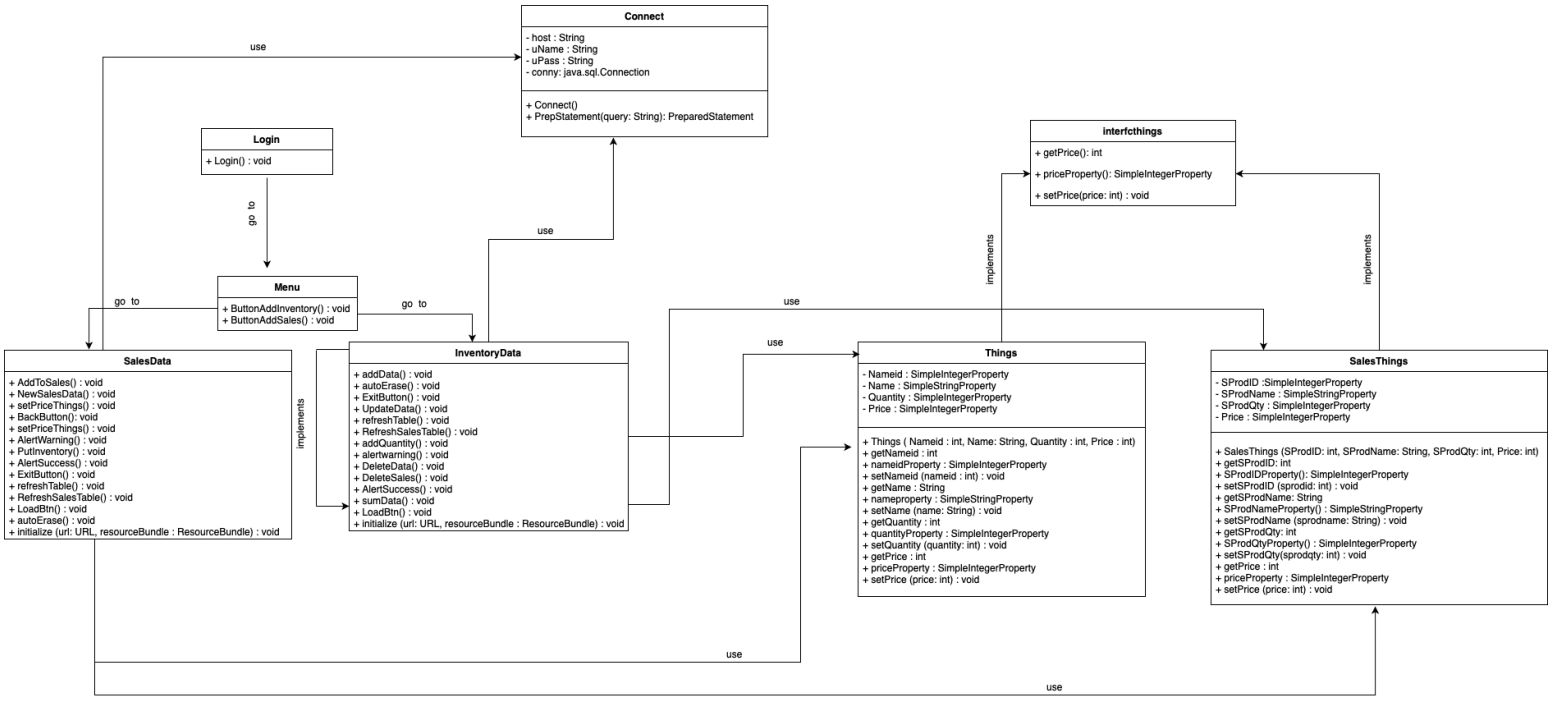
is also readable so it won't be hard for user to read it.

C. How It Works

1. Library:

- Java.sql: To connect this java program with mysql database.
- Javafx.collections: to create an observable list for table view and also choice box.
- Javafx.fxml & javafx.scene: this is the default library for javafx to show the fxml file (design).
- Javafx.io: This library is for the try and catch. If it is an error, it will show what causes the errors.
- Javafx.scene.Parent & javafx.scene.Scene: To control which one is the current scene and the Parent page
- Javafx.scene.control.*: initialize and control the controllers that we use in the GUI.
- Java.util : I use this library for the option of the alert.

2. UML Class Diagram



3. Explaination of the code:

```java
public class Connect{
    String host = "jdbc:mysql://localhost:3306/FinalProjectData?serverTimezone=UTC";
    String uName = "root";
    String uPass = "christy123";
    java.sql.Connection conny = null;

    public Connect () {
        try {
            conny = DriverManager.getConnection(host, uName, uPass);
            System.out.println("Success connect to database!");
        } catch (SQLException err) {
            System.out.println(err.getMessage());
        }
    }
```

Creating a class called Connect. Then, I initialize the attributes such as host which is the location and also the name of my database, uName and uPass for the username and password of those database. Then, set the conny into null. We will use conny to connect this java program with database. If it is successful, it will print the message on the console, otherwise, it will print the error messages. By creating this class, it will make us easier to access the database without writing down the function again. We can access this on another class by calling the class and the function.

```
public PreparedStatement Prepstatement(String query) {
    try {
        PreparedStatement prepStat = conny.prepareStatement(query);
        return prepStat;
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return null;
}
```

Next, we create a new function inside Connect class which is Prepstatement that implements PreparedStatement and takes a parameter called query with String type. Query is the blob text that we will use to insert, update, delete, and etc to edit database from this java program. If there is an error on the query, it will print the error message so it will be easier to find the error.

```
public void Login(){
    if (TFUsername.getText().equals("admin") && TFPassword.getText().equals("admin"))
    {
        Alert al = new Alert(Alert.AlertType.INFORMATION);
        al.setTitle("INFORMATION");
        al.setHeaderText("Success!");
        al.setContentText("Successfully login!");
        Optional<ButtonType> rs = al.showAndWait();
        ButtonType button = rs.orElse(ButtonType.OK);
        if (button == ButtonType.OK) {
            Stage stage = (Stage) BtnLogin.getScene().getWindow();
            stage.close();
            try {
                FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource( name: "Menu.fxml"));
                Parent root1 = fxmlLoader.load();
                Stage st = new Stage();
                st.setScene(new Scene(root1));
                st.setTitle("Inventory Database Program");
                st.show();
            } catch (Exception e) {
                System.out.println("Can't open");
            }
        }
    }
    else
    {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setTitle("WARNING!");
        a.setContentText("Incorrect Login Information");
        a.show();
```

For the parent page, we have to create a class called Login. So the default username and password are admin. I've decided not to create a register part for the new user so only authorized person are allowed to edit and change this data. If it is not correct, it will show the warning message. Otherwise, the stage will be changed to the menu. On the menu page, user have to choose which data that they want to edit.

```
public void addData() {
    int nameid = Integer.parseInt(tfnameid.getText());
    String name = tfname.getText();
    int qty = Integer.parseInt(tfqty.getText());
    int price = Integer.parseInt(tfprice.getText());

    //check if the Nameid already exist in database or not
    String mysql = "SELECT * FROM things_table WHERE Nameid=?";
    PreparedStatement pr1 = connectt.Prepstatement(mysql);
    try{
        pr1.setInt(1,nameid);
        ResultSet rs1 = pr1.executeQuery();
        if(rs1.next())
        {
            //if already exist, it will return an error messages.
            Alert al = new Alert(Alert.AlertType.WARNING);
            al.setTitle("WARNING!");
            al.setContentText("Data already exist!");
            al.show();
        }
```

The next step, I've created a class called InventoryData. On this class, we will create the actions for all of the buttons in the .fxml file. On this class, we can update the database from this java program by using sql statements. The primary key for this database is Product ID. That means, you can't input the same product ID and it has to be unique. The first one is AddNewData(). In this function, users can add some new data that are not available in the inventory table. In the beginning, create 4 new variables to hold the data from user input. Then, on mysql variable, we create a text to select all of data on things_table with the given product ID that users input. On things_table, the column name for product ID is Nameid and set the value of it as '?' because we will use it as a dynamic SQL statement which can be changed depends on the user inputs. After that we create a PreparedStatement. PreparedStatement represents a precompiled SQL statement which is mysql object. connectt represents an active connection from this java projects to the database. After that, we set a parameter because I'm using a dynamic SQL statement. 1 represents the position of '?'. So, it will replace the first '?' with nameid.

ResultSet is a library that can return a value if we're using SELECT as our SQL Statement. On this part, I create a new object called rs then execute the PreparedStatement. The next() method of the ResultSet interface moves the pointer of the current (ResultSet) object to the next row, from the current position. In this part, I will use it to check whether the data already exist or not. If it is exists, an alert message will be shown.

```
        else
        {
            //else, it will create the new data.
            String sql = "INSERT INTO things_table(Nameid, Name, Quantity, Price)" + "VALUES(?,?,?,?)";
            PreparedStatement prepstat = connectt.Prepstatement(sql);
            try {
                prepstat.setInt(1, nameid);
                prepstat.setString(2, name);
                prepstat.setInt(3, qty);
                prepstat.setInt(4, price);
                prepstat.executeUpdate();
                refreshTable();
                autoErase();

            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }
    catch(SQLException es)
    {
        System.out.println(es.getMessage());
    }
}
```

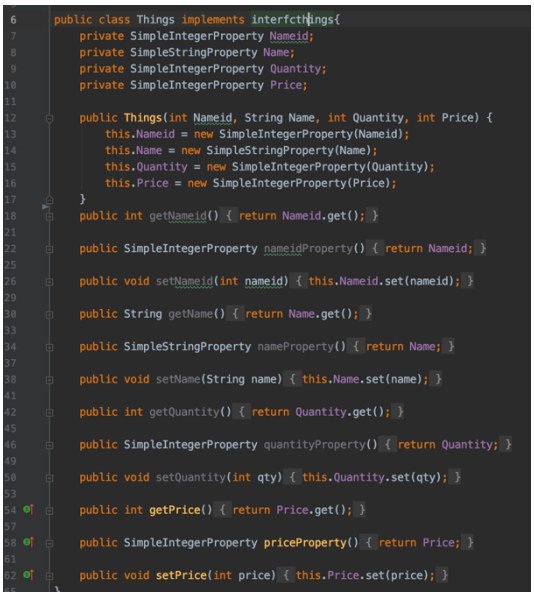
Otherwise, it will insert a new data into our database. In this part we create a new object called sql with String data type. The SQL Statement is:

```
"INSERT INTO things_table(Nameid, Name, Quantity, Price)" + "VALUES(?,?,?,?)";
```

So, We will insert all of those data into things_table which is the inventory data. Then, the column names are Nameid , Name , Quantity , and Price. Since we want to insert 4 values, there will be 4 '?'. After we finish on creating the SQL Statement, we will create a PreparedStatement to execute SQL Statement. Set the '?' value in order with the column's name in the database with each own data type. The value is taken from the text field that users input. Then, it will execute the SQL statement with all of those parameters. If there is an error with the code, it will return an error message because I'm using SQLException. SQLException will detect and print the message of the error so the coder don't have to find the error from the beginning.

After we insert the data into the database, we also have to update the table so it won't confuse the user. Every time user makes a change, both of sales and inventory table will be updated. In this part, we called refreshTable() function. Then, we're using autoErase function. This function will erase all of those text field after the user click the button to make some changes of the data.

All of the things in here are almost same. The difference only on the SQL statements which are INSERT INTO, UPDATE, DELETE, and etc.



```
public class Things implements interfcthings{
    private SimpleIntegerProperty Nameid;
    private SimpleStringProperty Name;
    private SimpleIntegerProperty Quantity;
    private SimpleIntegerProperty Price;

    public Things(int Nameid, String Name, int Quantity, int Price) {
        this.Nameid = new SimpleIntegerProperty(Nameid);
        this.Name = new SimpleStringProperty(Name);
        this.Quantity = new SimpleIntegerProperty(Quantity);
        this.Price = new SimpleIntegerProperty(Price);
    }
    public int getNameid() { return Nameid.get(); }

    public SimpleIntegerProperty nameidProperty() { return Nameid; }

    public void setNameid(int nameid) { this.Nameid.set(nameid); }

    public String getName() { return Name.get(); }

    public SimpleStringProperty nameProperty() { return Name; }

    public void setName(String name) { this.Name.set(name); }

    public int getQuantity() { return Quantity.get(); }

    public SimpleIntegerProperty quantityProperty() { return Quantity; }

    public void setQuantity(int qty) { this.Quantity.set(qty); }

    public int getPrice() { return Price.get(); }

    public SimpleIntegerProperty priceProperty() { return Price; }

    public void setPrice(int price) { this.Price.set(price); }
```

```
public class SalesThings implements  interfcthings{
    private SimpleIntegerProperty SProdID;
    private SimpleStringProperty SProdName;
    private SimpleIntegerProperty SProdQty;
    private SimpleIntegerProperty Price;

    public SalesThings(int SProdID, String SProdName, int SProdQty, int Price) {
        this.SProdID = new SimpleIntegerProperty(SProdID);
        this.SProdName = new SimpleStringProperty(SProdName);
        this.SProdQty = new SimpleIntegerProperty(SProdQty);
        this.Price = new SimpleIntegerProperty(Price);
    }
    public int getSProdID() { return SProdID.get(); }

    public SimpleIntegerProperty SProdIDProperty() { return SProdID; }

    public void setSProdID(int sprodid) { this.SProdID.set(sprodid); }

    public String getSProdName() { return SProdName.get(); }

    public SimpleStringProperty SProdNameProperty() { return SProdName; }

    public void setSProdName(String sprodname) { this.SProdName.set(sprodname); }

    public int getSProdQty() { return SProdQty.get(); }

    public SimpleIntegerProperty SProdQtyProperty() { return SProdQty; }

    public void setSProdQty(int sprodqty) { this.SProdQty.set(sprodqty); }
    @Override
    public int getPrice() {
        return Price.get();
    }
    @Override
    public SimpleIntegerProperty priceProperty() { return Price; }
    @Override
    public void setPrice(int price) {
        this.Price.set(price);
```

Things.java *SalesThings.java*

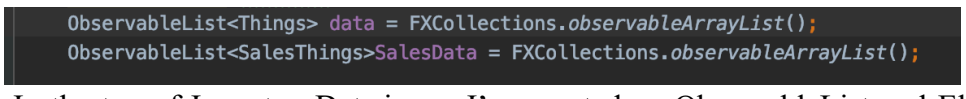
Then, I created a class called Things.java. On this class, we will create a setter and getter method for the database. In this class we will be implementing interfcthings interface.

```
package sample;

import javafx.beans.property.SimpleIntegerProperty;

public interface interfcthings {
    public int getPrice();
    public SimpleIntegerProperty priceProperty();
    public void setPrice(int price);
    }
```

On this interface we will only create a function for the Price because the name of the column in the database are same. So, this interface will be implemented by Things.java and also SalesThings.java



```
ObservableList<Things> data = FXCollections.observableArrayList();
ObservableList<SalesThings>SalesData = FXCollections.observableArrayList();
```

In the top of InventoryData.java , I've created an ObservableList and Elements data type is from Things.Class. Then, set the object name as data and initialize it with observable Array list.

```
public void refreshTable() {
    data.clear();
    String sql = "SELECT * FROM things_table";
    PreparedStatement prepstatt = connectt.Prepstatement(sql);
    try {

        ResultSet rs = prepstatt.executeQuery();
        while (rs.next()) {
            data.add(new Things(rs.getInt( s: "Nameid"), rs.getString( s: "Name"), rs.getInt( s: "Quantity"),
                    rs.getInt( s: "Price")));
        }
        sumdata();
        autoErase();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

This function is called refreshTable(). As the name, this function will refresh things_table which is Inventory table. To keep the data updated, we will clear all of the value in the observable list called data. Then, we're going to select all of the data by using the SQL Statement:

```
"SELECT * FROM things_table";
```

It means, it will select all of the data from things_table and the pointer is on the first column and first row. Then, I'm using ResultSet to make the pointer move from one cell to another cells. If the value on those row is not empty, we will add those row data into the list called data.

```
ObservableList typelist = FXCollections.observableArrayList( ...items: "Inventory", "Sales");
```

In the top of InventoryData.class , create a list with the name *typelist* and set the items as "Inventory" and also "Sales". In here, I'm trying to create a choicebox to show the table based on the user preferences.

```
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        choicebox1.setItems(typelist);
    }
}
```

In this part, we create an initialize function. This function will be automatically working without any response from the user. I have made a choicebox and then the id is choicebox1. By using this function, I want to set the items with the list called typelist. So, the value of those choicebox1 is "Inventory" and also "Sales".

```
public void LoadBtn(){
    int sumdata1 = 0;
    String choiceboxValue = choicebox1.getValue();
    if (choiceboxValue == "Inventory") {
        refreshTable();
        TableColumn NameIDCol = new TableColumn( text: "Product ID");
        NameIDCol.setMinWidth(50);
        NameIDCol.setCellValueFactory(
                new PropertyValueFactory<Things, Integer>("Nameid"));

        TableColumn NameCol = new TableColumn( text: "Product Name");
        NameCol.setMinWidth(100);
        NameCol.setCellValueFactory(
                new PropertyValueFactory<Things, String>("Name"));

        TableColumn QtyCol = new TableColumn( text: "Quantity");
        QtyCol.setMinWidth(100);
        QtyCol.setCellValueFactory(
                new PropertyValueFactory<Things, Integer>("Quantity"));

        TableColumn PriceCol = new TableColumn( text: "Price");
        PriceCol.setMinWidth(50);
        PriceCol.setCellValueFactory(
                new PropertyValueFactory<Things, Integer>("Price"));


        tvResult.getColumns().setAll(NameIDCol, NameCol, QtyCol, PriceCol);
        tvResult.setItems(data);

        sumdata();
        labelcount.setText(String.valueOf(sumdata1));
    }
```

Next, we're going to show all of the data from the database in the table view. My table view name is tvResult. Then, if the choicebox value is Inventory and the user clicked on the load data button, it will show things_table on the table view.

First, create a new object from TableColumn. Set the label for each column. Then, take the value from the database by calling Things.java class and set the data type. After that fill in the parameter with the corresponding name in the database. Do the same things to the rest of it.

```
tvResult.getColumns().setAll(NameIDCol, NameCol, QtyCol, PriceCol);
```

This code is to get all of the column in the tableview and then set the column with the Object that we've made.

```
tvResult.setItems(data);
```

This code is to set all of those column's value with the observable list called data.

```
public void sumdata(){
    int sumdata1 = 0;
    String choiceboxValue = choicebox1.getValue();
    if (choiceboxValue == "Inventory") {
        String sql1 = "SELECT * FROM things_table";
        PreparedStatement prepstat = connectt.Prepstatement(sql1);
        try {
            ResultSet rs = prepstat.executeQuery();
            while (rs.next()) {
                sumdata1 = sumdata1 + rs.getInt( s: "Quantity");
            }
        } catch (SQLException es) {
            System.out.println(es.getMessage());
        }
        labelcount.setText(String.valueOf(sumdata1));
    }
    else if (choiceboxValue == "Sales")
    {
        String sql1 = "SELECT * FROM Sales_table";
        PreparedStatement prepstat = connectt.Prepstatement(sql1);
        try {
            ResultSet rs = prepstat.executeQuery();
            while (rs.next()) {
                sumdata1 = sumdata1 + rs.getInt( s: "SProdQty");
            }
        } catch (SQLException es) {
            System.out.println(es.getMessage());
        }
        labelcount.setText(String.valueOf(sumdata1));

    }
    }
```

sumData() function is to count the number of the product in both of those table separately. The number of data will be shown based on the choicebox value. For example the first one is to count total data in the inventory table, then SELECT all of those data from things_table by using SQL Statements. Then, I create a temporary variable called sumdata1 to store the calculation. After that, it will replace the label under the table with the sumdata1.

```
public void BackButton() throws IOException {
    // to go back to menu page
    Stage stage = (Stage) BtnBack.getScene().getWindow();
    stage.close();
    FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource( name: "Menu.fxml"));
    Parent root1 = fxmlLoader.<~>load();
    Stage st = new Stage();
    st.setScene(new Scene(root1));
    st.setTitle("Inventory Database Program");
    st.show();

}
```

BackButton() is a function to go to the menu page. The first thing to do is to get the stage where the button is clicked. In this file, the button name is BtnBack. Then, it will close the stage that BtnBack is located. By using FXMLLoader, we will load Menu.fxml. Calling the Parent library to load the fxmlLoader object. Lastly, call Stage library to create a new stage and assign Parent object which is root1 as a new scene. So, Menu.fxml will be appeared.

D. Evidence of Working Program

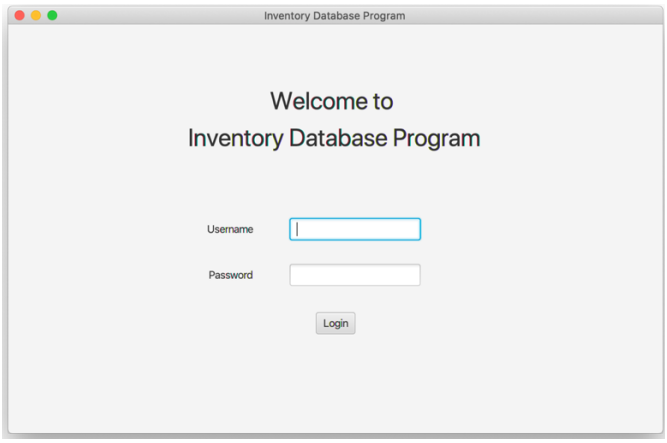


Figure 1. The main page.

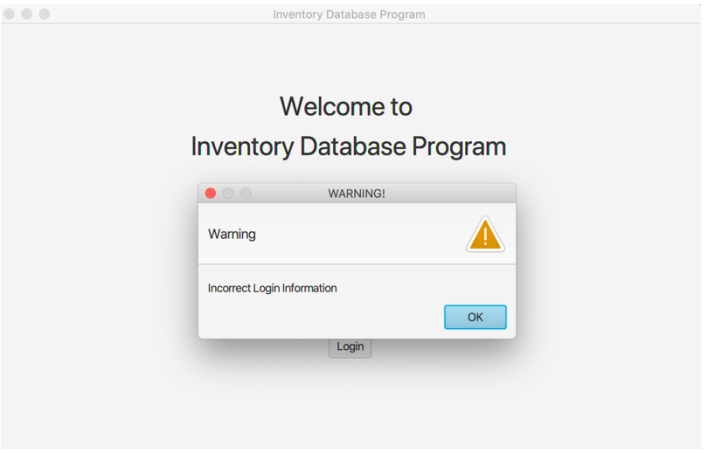


Figure 2. Warning message if the login information is not correct.

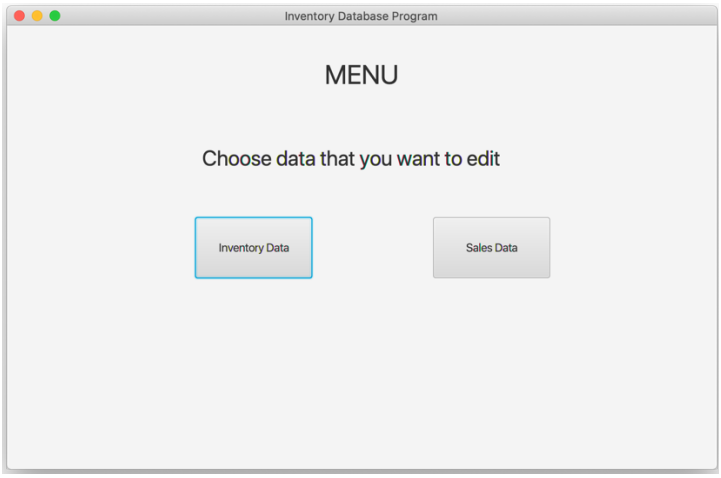


Figure 3. Menu page.

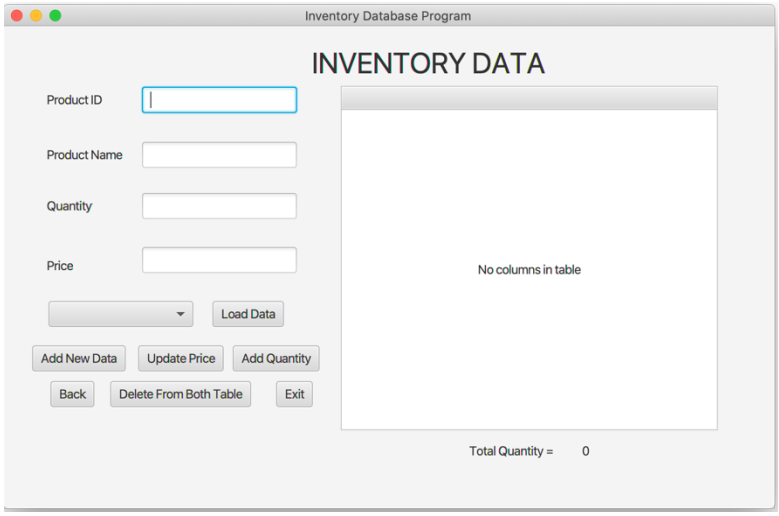


Figure 4. If the users click on inventory data, this menu will be appeared

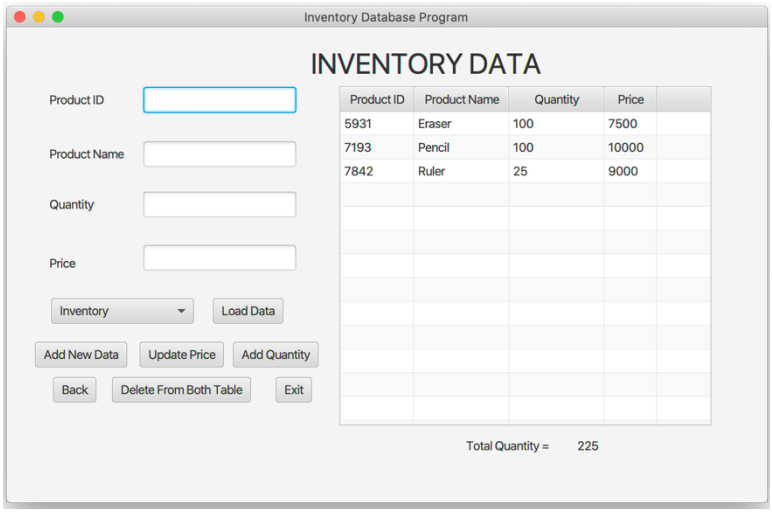
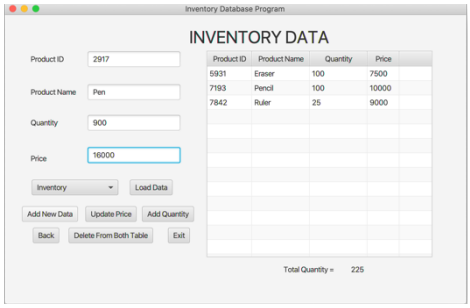


Figure 5. To show the table of the data, users have to click load data button and also choose the table to show in the choice box. In the bottom, it will show the total number of products on the corresponding database.



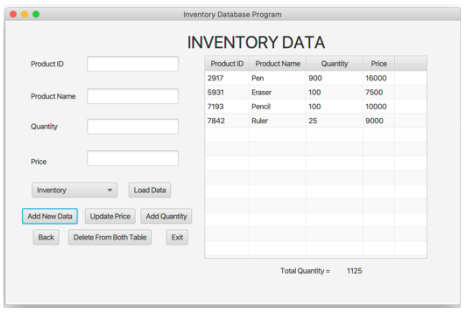


Figure 6 & 7. When the user wants to add new data, they only have to type the new data and then click on add new data button. It will appear by automatically in the table.

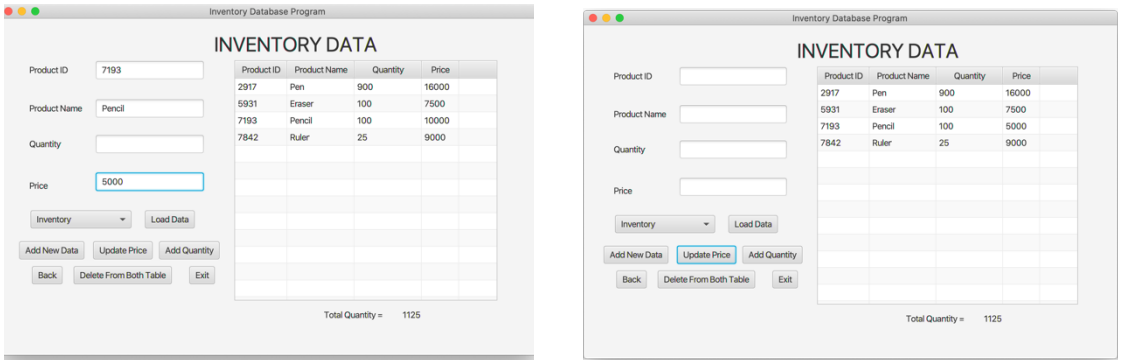


Figure 8 & 9. Users can update the price for the data that already exist.

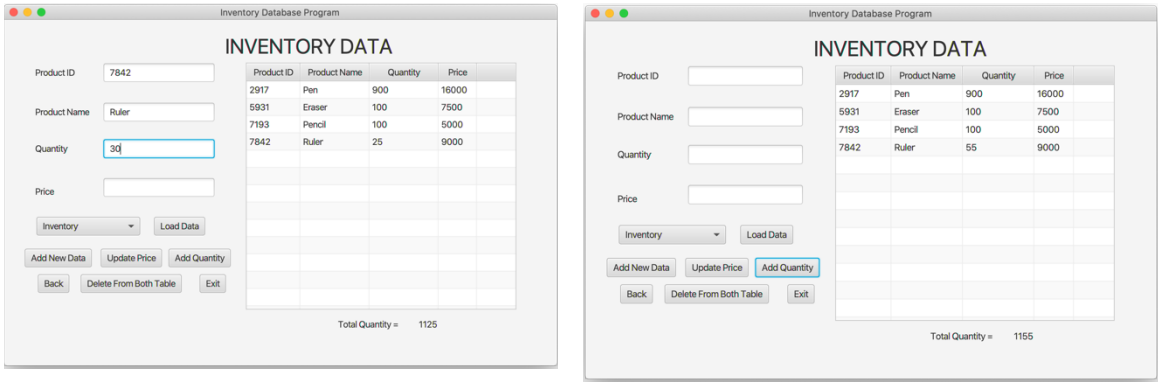


Figure 10&11 To add more quantity, user only have to input a valid product id and name. It will sum the total by automatically.

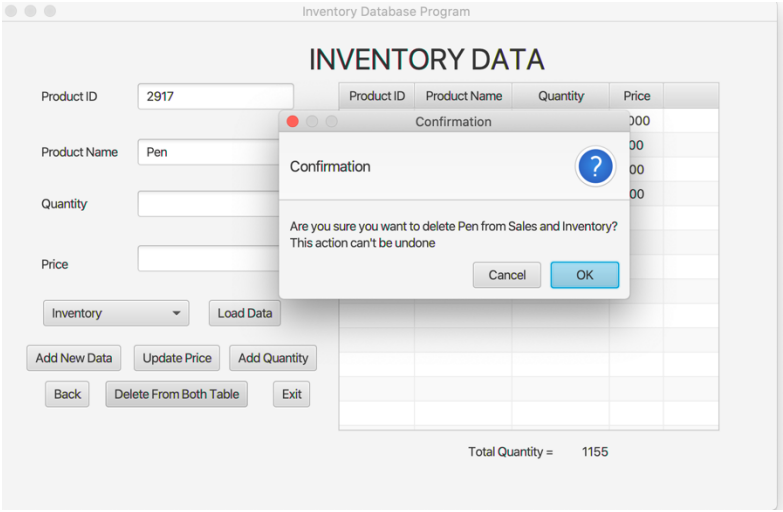
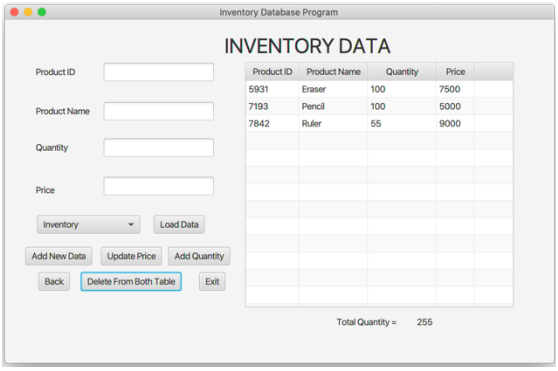


Figure 12. Confirmation if the user wants to delete the data.



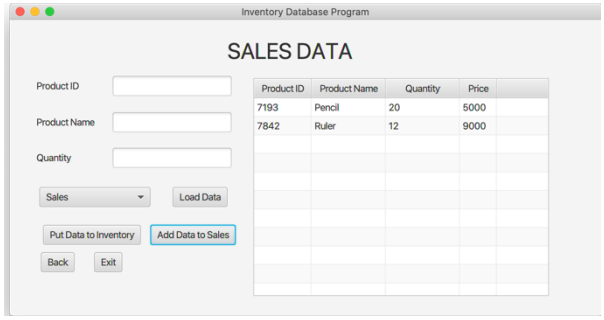


Figure 13&14. When the user click the ok button, it will delete the data from both of this table.

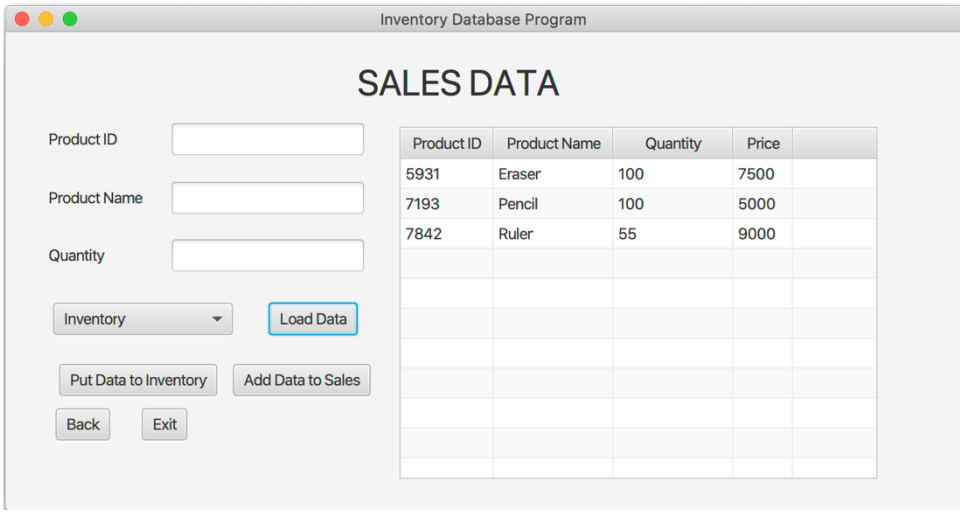
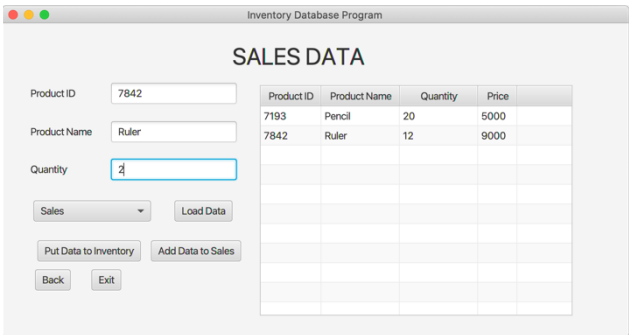


Figure 15. On the menu page, if the user click on sales data, it will show this menu.



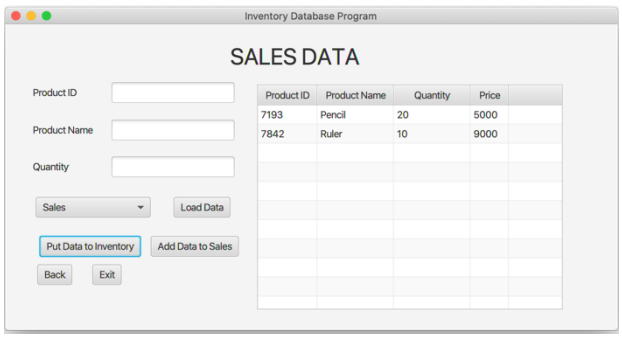


Figure 16&17. You can put back some amount of quantity to inventory from sales.

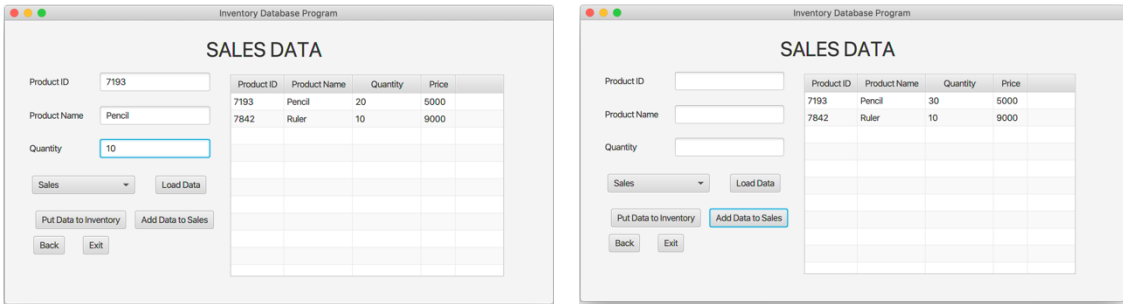


Figure 18& 19. Users can also move some amount of quantity from inventor to sales. It will reduce the quantity from inventory and add it on sales data.

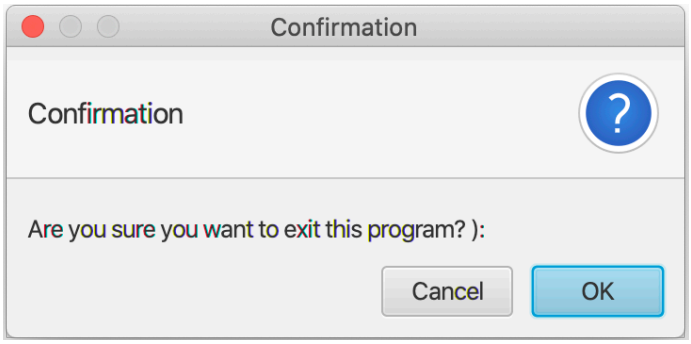


Figure 20. Confirmation to exit the program. If the user click ok, it will close this program.

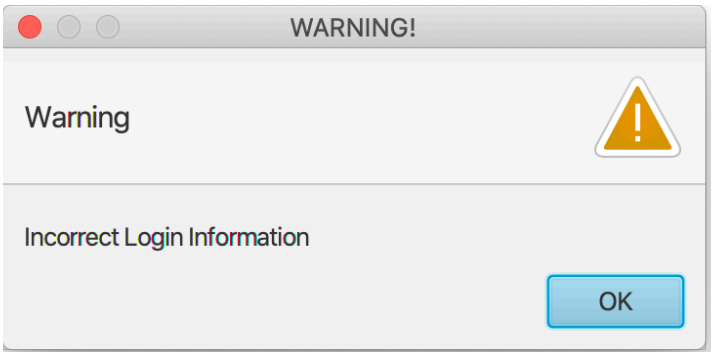


Figure 21. Every time the data that user input is not valid, it will show this warning message.

Demo video: https://youtu.be/0MAXCZn_9ug

E. Conclusion

To conclude, creating a database program is not that easy. It needs some extra works especially for me to connect the database with this java program. I think, this program will be useful for people who wants to control the data inside their inventory. People don't need any paper and pen anymore to do this.

F. Source Code

https://www.tutorialspoint.com/jdbc-class-forname-vs-drivermanager-registerdriver
(to connect the database with the java program)
https://www.youtube.com/watch?v=9sbUsbDWTE8&t=568s (to download and setup the mysql database things)
https://www.w3schools.com/sql/sql_intro.asp (for the SQL Statement)