COMP1148 – Computer Programming
Stock Control Simulation
Written Report

University of Greenwich
Usman Basharat
000874782

Table of Contents

Introduction

In this report, I am going to identify the description of the stock control simulation program, how I designed my final code, any faults that came across whilst I have been working on it, a UML diagram of my program, testing my program and reflect upon what I have done.

The aims of the Stock Control Simulation are: -

- Being able check what items are in stock
- Being able to purchase an item
- Being able to view the products available
- For staff in the warehouse to have access to updating the stock, once having access, they can add new stock, update and delete stock

Description

Designing the stock control system was designing it on paper before transferring it to code. By planning, it gave me an idea of how to get to where I am right now. The main objective of this stock control simulation is that it allows the user to purchase the item and to update any item. Referring to Figure 1.1, as any new user does not know how to use it can use the Help and Support button which is shown below. Stock List button shows all of the stock available with the database connected with each other.

Figure 1 shows the main program

Referring to Figure 2, this shows the key being typed into the Text Field and the button has been clicked. Once this has been done, the image is displayed too. Once this is complete, the user needs to type in how many of this they need. Referring to Figure 3, this shows the total cost of the items being typed and how many are left.

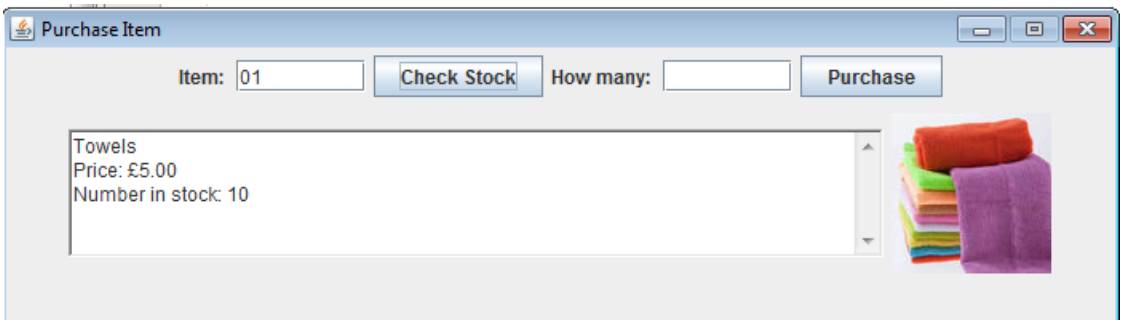


Figure 2 shows the key being typed with an image

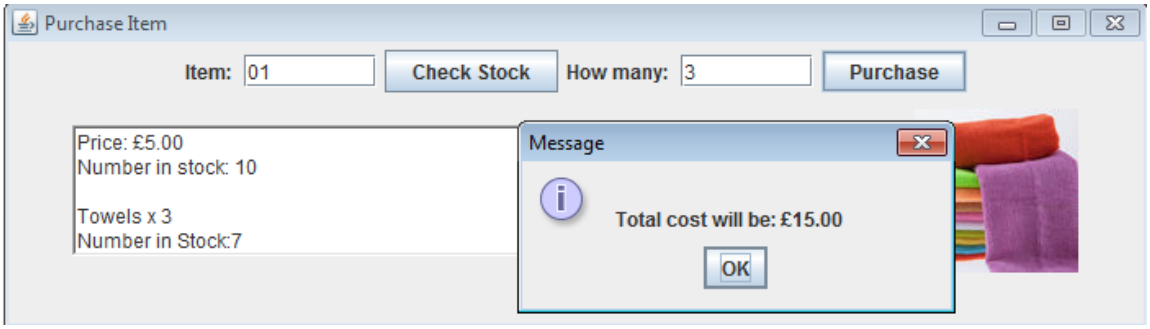


Figure 3 shows the item being purchased

Once purchasing the item, the stock control system would need to update any existing items. Before this, referring to Figure 4, it shows that the warehouse staff needs to be able to type the username and password before doing this. If not successful, the user would not be able to have access to updating stock. Once this is complete, as shown, it gives the user two options. These two options reflect upon what you want to update. If the warehouse staff wants to update the existing stock, he clicks the first one. If he wants to enter a new stock, he can use the update new button. Figure 5 shows us the options of what to update and Figure 6 shows us by using this, you can type the following to add new stock and it being successful. From Figures 7 to 9, it shows each and every one of Figure 5.

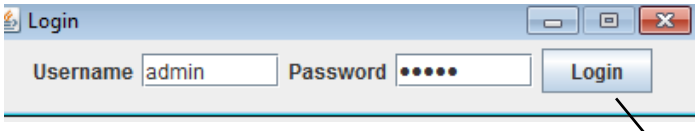


Figure 4 shows the login screen with its steps

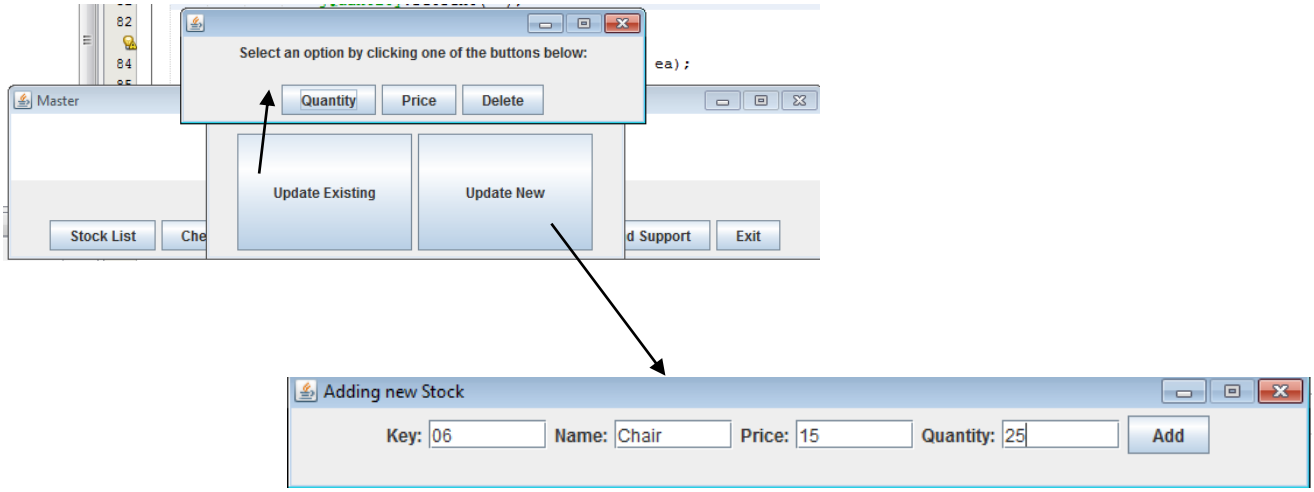


Figure 5 shows the fields of the stuff the user can type to add new stock and once add is pressed, it connects to the database.

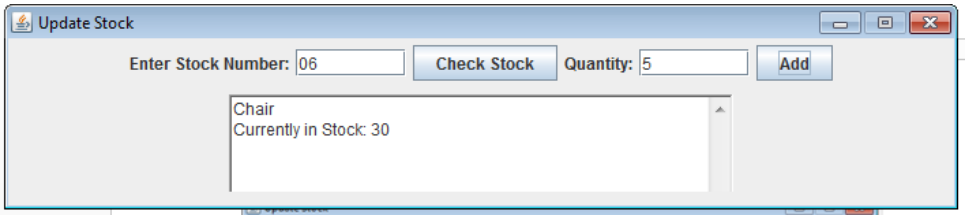


Figure 6 shows quantity being added to the new stock and it being updated once it has been typed in.

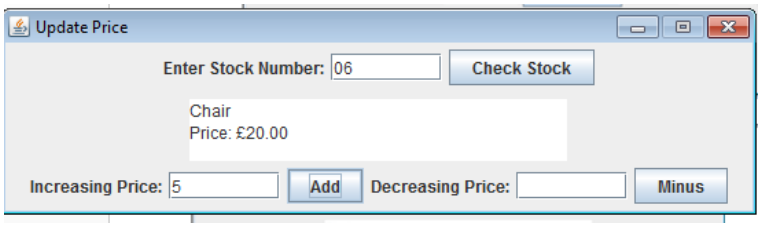


Figure 7 shows the price being updated

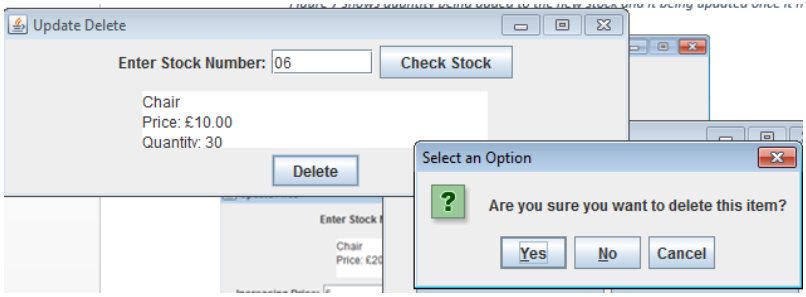


Figure 8 shows the new stock being inserted has been deleted

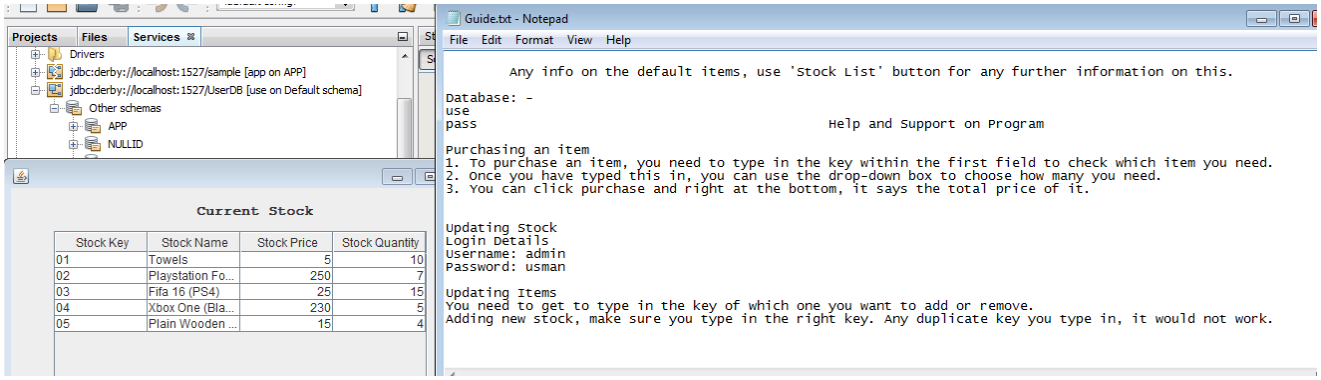


Figure 9 shows Help and Support guide (right) and a Stock List that is connected to a database (right). The Stock List can be updated once a new stock is entered.

Figure 10 shows us that the user can look at the current stock available. Any new stock that is being added to the system, it updates within it. Figure 9 shows us that the new Stock can be deleted too. However, the staff has a confirm option before selecting this. If the user has typed in a key that he/she did not want, he can simply go back and type in the one he/she wants to delete. Figure 8 shows the price being updated. The price can be updated by typing in any sort of integer, or double they want. This is done by returning a double from the StockData using the new update Method for it. This is connected to the database too. The whole of update Stock from Master has been connected to the database. Any updates can be run from there.

Faults and Failures

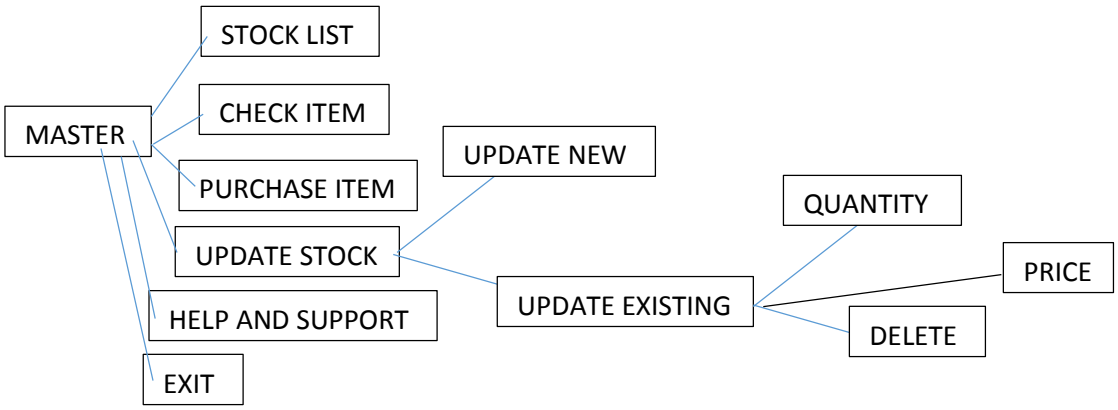
The faults that I came across whilst creating my database was that once I created my database, whenever I tried to log in to another machine, it was not letting me access the database that I created with the location being the same. I tried repeatedly. However, I overcame this fault by trying to change where I saved my project. I put it in my USB and it worked. Therefore, I could access my database every time I logged in on another machine.

The failures that I came across, once I was doing my program, is that I was trying to put the keypad as part of innovations. So the first thing I tried to do is, using Safe.java as a guide, create the buttons as an array. This was successful and I connected the 0-9 buttons with the 'stockNo Text Field' to these buttons that were created. However, the one thing that was not working is it was only typing one key at a time. It was only typing one button at a time. I tried another way to create all those buttons without an array using if statements. This problem was the same problem. Therefore, I then did not put it in my program.

I have tested my program and I ran the program as usual. And, then I went to Check Item and typed in a key. It appeared an error. I checked within the Check Item Class and there was no error. Then, I went towards the StockData to see if the connection had failed and it did. The password was wrong and it gave an error. I then realised that the password needed to be correct. As soon as I put in the correct password and I ran the program again using the same process, it was correct.

Another error that I came across was trying to return a double within my code in UpdatePrice. I wanted to type in £15.50 to add a price. And it was only returning the typed in item as a whole. Therefore, I had to change the data type for my stockPrice within my database from decimal to double. Therefore, it allowed me to do update the price using £15.50 as a price.

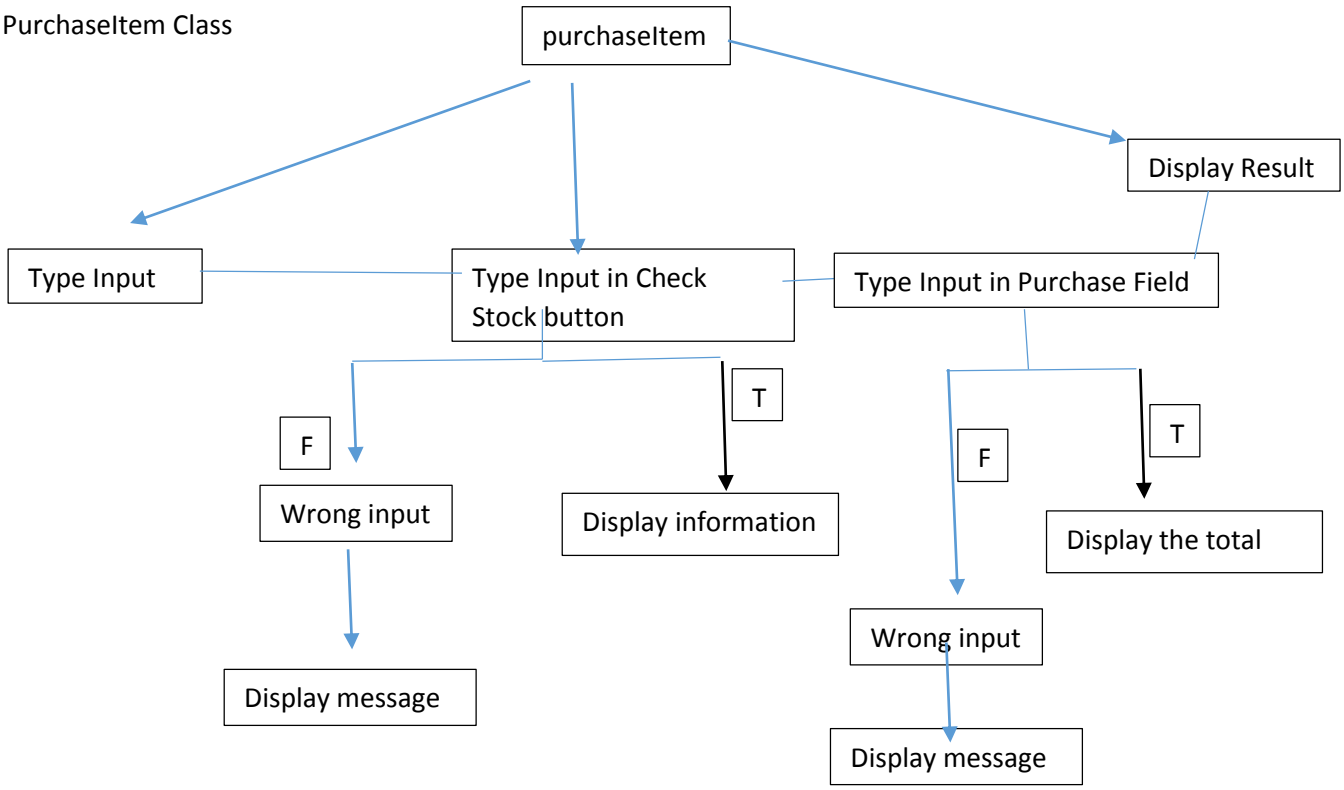
Diagram of my program



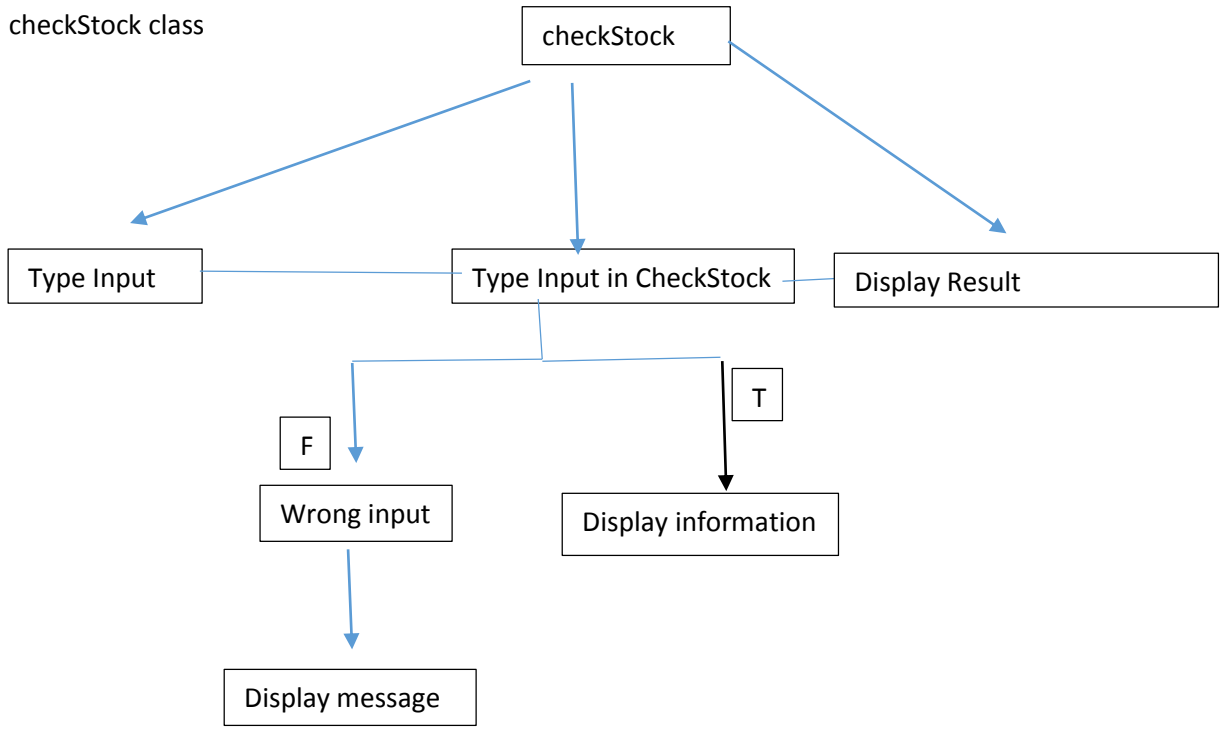
This shows the structure of the whole program. At the end is where the class is displayed.

White Box Testing

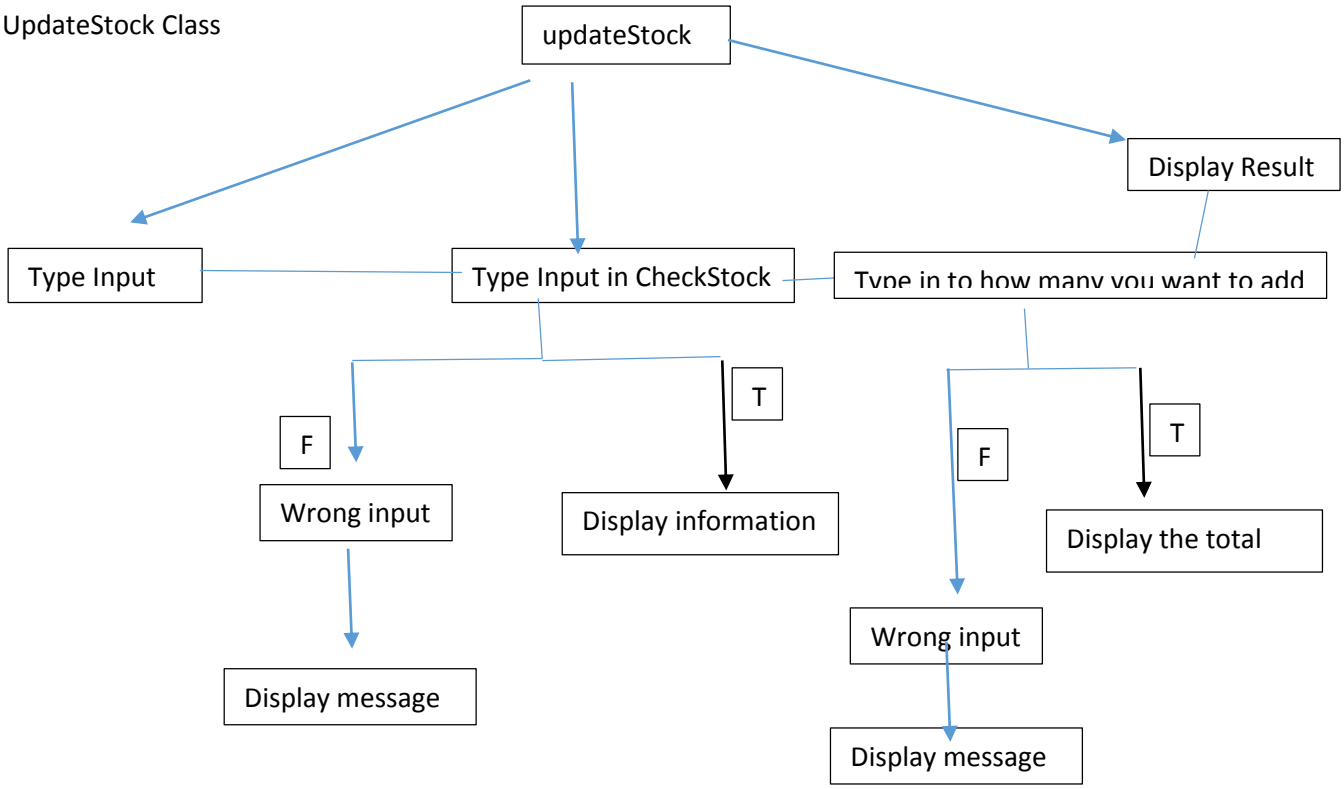
PurchaseItem Class



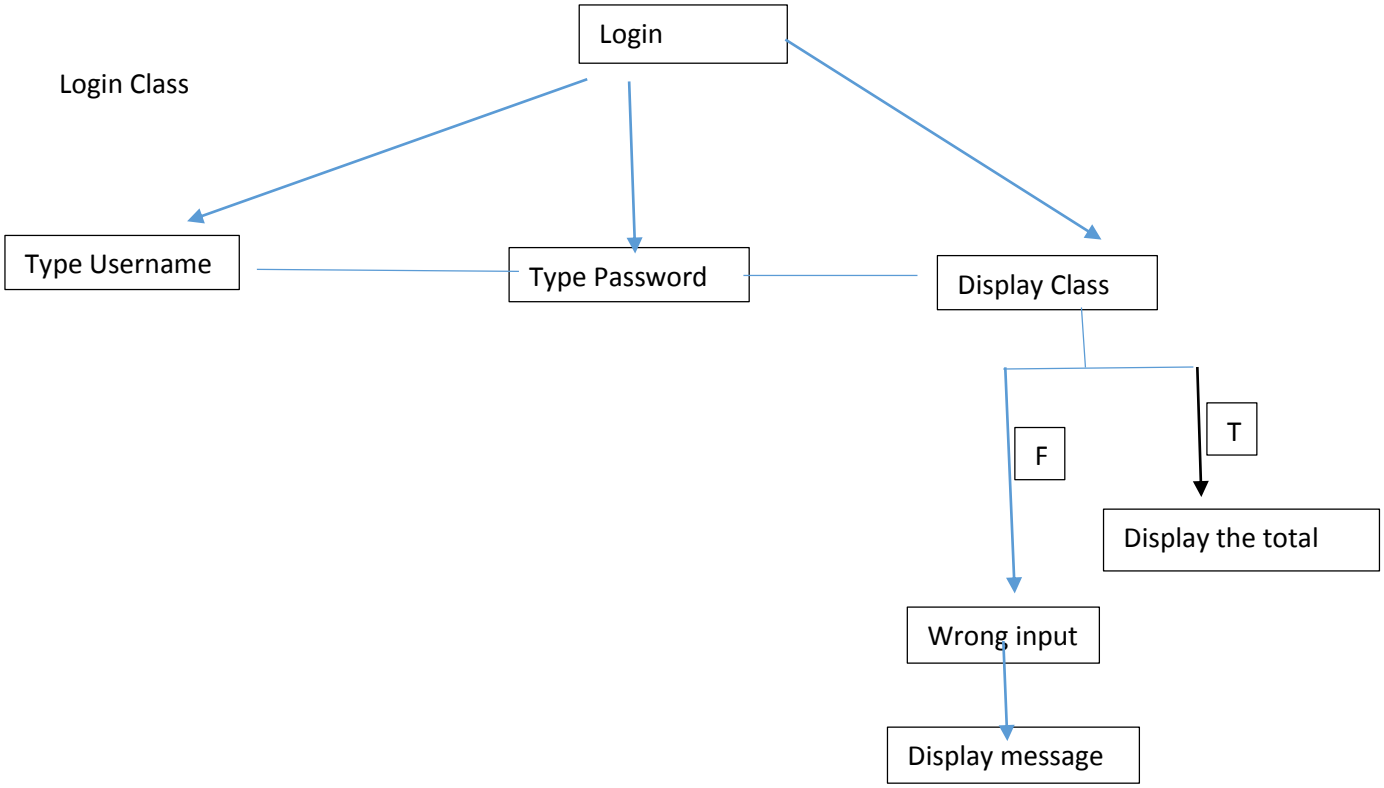
checkStock class



UpdateStock Class



Login Class



These white box testing from the classes above show us how the results are displayed and any errors with it. This is important as it shows where the messages are taking place. If there is not any, the user can put it in.

Conclusion

During this course, I felt that I have developed skills that I never thought I would be able to reach. I feel that being able to put time and effort into something, it would make it easier once the objective is complete. I feel that this part of the course has given me more confidence to practice more. I feel that being able to practice regularly enhances my skills further. Having time doing this was a real bonus. In that time, we got to learn what we had to do from scratch. Even though it sounded hard, by attending lectures and tutorials, it gave a bonus to get better at it and practice each week.

Appendices:

A) Commented Version for CheckStock for Interim Deliverable A

Understanding the code

```
package stock;

//all imports that is ncessary for this class is included below

import java.awt.BorderLayout;

import java.awt.event.ActionListener;

import javax.swing.JFrame;

import javax.swing.JTextField;

import java.awt.TextArea;

import java.awt.event.ActionEvent;

import java.text.DecimalFormat;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;


public class CheckStock extends JFrame implements ActionListener {

                // extends JFrame is used and ActionLister is needed to

                //implement any buttons, or textareas to listen out for

   JTextField stockNo = new JTextField(7);

   //new field for stock

   TextArea information = new TextArea(3, 50);

   //creating a new textarea

   JButton check = new JButton("Check Stock");

   // creating a new button called "Check Stock"

   DecimalFormat pounds = new DecimalFormat("£#,##0.00");
```

```
//formatting decimals

 public CheckStock() {
    setLayout(new BorderLayout());
    //new layout
    setBounds(100, 100, 450, 150);
    //window measurements
    setTitle("Check Stock");
    //title name at the top
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    JPanel top = new JPanel();
    top.add(new JLabel("Enter Stock Number:"));
    // adding label to the top and labelling the label
    top.add(stockNo);
    //adding textfield to the window
    top.add(check);
    //adding button to the window
    check.addActionListener(this);
    //listening out to the check button
    add("North", top);
    //adding all the new features to the top
    JPanel middle = new JPanel();

    middle.add(information);
    add("Center", middle);
    //adding textarea to the middle

    setResizable(false);
    //resizing the window cannot be resized
    setVisible(true);
    //set to true to make the whole window visible
```

```
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String key = stockNo.getText();
        String name = StockData.getName(key);
        if (name == null) {
            information.setText("No such item in stock");
            //if no stock data that is not recognised, the outcome would insert
            //this statement within the textarea information
        } else {
            information.setText(name);
            information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
            information.append("\nNumber in stock: " + StockData.getQuantity(key));
            //or else, add these information once the stock right is recognised.
            //Whatever is recognised is stored within StockData.
            //An example is 01. Once this is inserted, all the information about 01 is inserted.

        }
    }
}
```

B) Test Plan

Test Number	Item to test	Method	Expected Output/Result	Actual Output/Result
1	Opening another class using button e.g. using Master class	e.getSource() addActionListener	Expecting the button to open up a class	The class opens up once clicking on the button
2	Typing in key e.g. 01, 02 once check pressed has been button in any class	String key = stockNo.getText(); String name = StockData.getName(key); information.setText(name);	Expecting the information to open for the specific key	The information is displayed for that specific key e.g. price, name and quantity. However, in the method, this

		information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
		
		information.append("\nQuantity: " + StockData.getQuantity(key));
				shows the name only,

3	Key	String key = stockNo.getText();
		String name = StockData.getName(key);
		information.setText(name);
		information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
		
		information.append("\nQuantity: " + StockData.getQuantity(key));
			Expecting to type in 01 and return Towels.	Returned with Towels.

4	Not typing the specific key that is within the database e.g. -1, any character, 0 or something that doesn't match.	if (name == null) {
		information.setText("No such item in stock");
		}
			Expecting the message to be set once the wrong key is inserted within the textfield	The message was displayed by me typing a wrong key , or any other

5	Image displayed using key	JLabel imageLab = new JLabel(new ImageIcon("images.empty.jpg"));
		
		middle.add(imageLab);
		String image = "Images\\" + key + ".jpg";
		File imageFile = new File(image);
		if (!imageFile.exists()) {
		image = "Images\\empty.jpg";
		}
		imageLab.setIcon(ne
			Expecting the images to be displayed as soon as the key has been pressed under Check Stock	As expected, the image was displayed

		w ImageIcon(image));		
6	Updating the quantity within database	int quantity = Integer.parseInt(update.getText()); StockData.update(key, quantity);	Expecting the number entered to be updated within the database	As expected, the number entered was updated within the database and the quantity was changed.

7	Inserting button	JButton deleteBtn = new JButton("Delete");
		bottom.add(deleteBtn);
		
		deleteBtn.addActionListener(this);
			Expecting the button to appear at the bottom and do what it needs to do	As expected, the button was created and it was at the bottom. It

8	The window class	UpdateDelete d = new UpdateDelete(); setVisible(false);	Expecting the class before to be closed and show the Update Delete class instead	As expected, the class before was not visible once 'delete' button was pressed.

9	Login details being correct and connected to a database	if (z.matches(s) && i.matches(w)) {
		
		display("Login Successful!");
		
		TwoButtons update = new TwoButtons();
			Expecting the username and password of login to be matched with a message saying, "Login Successful".	As expected, the login details were matched and the message was displayed.

10	Total price for Purchase	JOptionPane.showMessageDialog(null, "\nTotal cost will be:" + " " + pounds.format(totalCost));	Expecting the total price to appear for the selected item the user has chosen	The total price was shown for the item that was chosen with the total price being correct

11	Updating Price (add)	information.setText(name);
		double plusInt = Double.parseDouble(plus.getText());
		
		StockData.priceUpdatePlus(key, plusInt);
		
		information.append("\nPrice: " +
			Expecting the price to update and add to the total with it being connected to the database.	As expected, the price has been updated and added to the total and it has been connected to the database.

		pounds.format(StockData.getPrice(key)));		

12	Updating price (minus)	information.setText(name);
		double minusInt = Double.parseDouble(minus.getText());
		
		StockData.priceUpdateMinus(key, -minusInt);
		
		information.append("\nPrice: " + StockData.getPrice(key));
			Expecting the price to update and take away to the total	As expected, the price has been updated and taken away to the total

12	Updating Quantity	int quantity = Integer.parseInt(update.getText());
		
		StockData.update(key, quantity);
		
		information.append("\nCurrently in Stock: " + StockData.getQuantity(key));
			Expecting the quantity to update and add to the total within the database	As expected, the quantity has been updated, added to the total and updated in the database too.

13	Deleting Stock	String sql = "DELETE FROM Stock WHERE stockKey = '" + key + "'"; stmt.execute(sql); JOptionPane.showMessageDialog(null, "You have successfully deleted this item.");	Expecting the key to be deleted within the database too.	The key was deleted.
14	Adding new Stock	Connection con = DriverManager.getConnection(host, j, p); PreparedStatement stmt = (PreparedStatement) con.prepareStatement("INSERT INTO STOCK(stockKey,	Expecting the new item to be added to the database	The new item was added to the database.

		stockName, stockPrice, stockQuantity) " + "VALUES(?,?,?,?)");
		
		stmt.setString(1, keyStr);
		
		stmt.setString(2, nameStr);
		
		stmt.setDouble(3, priceDouble);
		
		stmt.setInt(4, quantityInt);
		
		//executing the statement
		
		stmt.executeUpdate();
		
		JOptionPane.showMessageDialog(null, "You have successfully added a new item");		

15	Guide.txt shows up once Help and Support button is up	if (e.getSource() == help) {
		if (Desktop.isDesktopSupported()) {
		Desktop desktop = Desktop.getDesktop();
		if (Desktop.isDesktopSupported()) {
		Desktop desktop = Desktop.getDesktop();
		
		try {
		
		desktop.edit(new File("Guide.txt"));
			Expecting the file to show was the button is pressed.	The file was shown once the button was pressed

		}catch (Exception ex) {
		
		System.out.println(ex);
		}		

16	Stock List is shown	List list = new List(); setVisible(true);	Expecting the Stock List to show.	The Stock List was shown.

17	Clearing TextField	username.setText("");
		
		password.setText("");
		
			Expecting the username and password textfields to be cleared	The text fields were cleared

18	Connecting to the database and executing the code	String host = "jdbc:derby://localhost:1527/UserDB";
		String j = "use";
		String p = "pass";
		
		Connection con = DriverManager.getConnection(host, j, p);
		
		PreparedStatement stmt = (PreparedStatement) con.prepareStatement("INSERT INTO STOCK(stockKey, stockName, stockPrice, stockQuantity) "
		+
		"VALUES(?,?,?,?)");
		
		stmt.setString(1, keyStr);
		
		stmt.setString(2, nameStr);
		
		stmt.setDouble(3, priceDouble);
		
		stmt.setInt(4, quantityInt);
			Expecting the database to connect and add the new stock	The new stock was added and it conncted to the database. This means, it was added to the database.

C) A full program listing

CheckStock

```
package stock;

//all imports that is ncessary for this class is included below


import java.awt.BorderLayout;

import java.awt.event.ActionListener;

import javax.swing.JFrame;

import javax.swing.JTextField;

import java.awt.TextArea;

import java.awt.event.ActionEvent;

import java.io.File;

import java.text.DecimalFormat;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;


public class CheckStock extends JFrame implements ActionListener {

// extends JFrame is used and ActionLister is needed to

//implement any buttons, or textareas to listen out for


JTextField stockNo = new JTextField(7);

//new field for stock

TextArea information = new TextArea(3, 25);

//creating a new textarea

JButton check = new JButton("Check Stock");

// creating a new button called "Check Stock"

DecimalFormat pounds = new DecimalFormat("£#,##0.00");

JLabel imageLab = new JLabel(new ImageIcon("images.empty.jpg"));
```

```
public CheckStock() {
setLayout(new BorderLayout());
//new layout
setBounds(100, 100, 600, 200);
//window measurements
setTitle("Check Stock");
//title name at the top
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
JPanel top = new JPanel();
top.add(new JLabel("Enter Stock Number:"));
// adding label to the top and labelling the label
top.add(stockNo);
//adding textfield to the window
top.add(check);
//adding button to the window
check.addActionListener(this);
//listening out to the check button
add("North", top);
//adding all the new features to the top
JPanel middle = new JPanel();
middle.add(information);
//adding textarea
middle.add(imageLab);
//adding label
add("Center", middle);
JPanel bottom = new JPanel();
add("South", bottom);
//adding textarea to the middle
setResizable(false);
//resizing the window cannot be resized
```

```
setVisible(true);
//set to true to make the whole window visible
}

@Override
public void actionPerformed(ActionEvent e) {
String key = stockNo.getText();

String name = StockData.getName(key);
if (name == null) {
information.setText("No such item in stock");
//if no stock data that is not recognised, the outcome would insert
//this statement within the textarea information
} else {
information.setText(name);
information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
information.append("\nQuantity: " + StockData.getQuantity(key));
//or else, add these information once the stock right is recognised.
//Whatever is recognised is stored within StockData.
//An example is 01. Once this is inserted, all the information about 01 is inserted.
//gets the image and uses it as a key to display it
String imageStr = "Images\\" + key + ".jpg";
File imageFile = new File(imageStr);
if (!imageFile.exists()) {
imageStr = "Images\\empty.jpg";
}
imageLab.setIcon(new ImageIcon(imageStr));
}
}
}
```

Existing Buttons

```
package stock;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class ExistingButtons extends JFrame implements ActionListener {

//creating buttons
JButton quantity = new JButton("Quantity");
JButton price = new JButton("Price");
JButton delete = new JButton("Delete");

public ExistingButtons() {
setLayout(new BorderLayout());
setSize(400, 100);
JPanel top = new JPanel();
top.add(new JLabel("Select an option by clicking one of the buttons below:"));
add("North", top);

JPanel bottom = new JPanel();
//adding buttons on bottom panel
bottom.add(quantity);
quantity.addActionListener(this);
```

```
bottom.add(price);
price.addActionListener(this);
bottom.add(delete);
delete.addActionListener(this);
add("South", bottom);
setLocationRelativeTo(null);
setResizable(false);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
if (e.getSource() == quantity) {
UpdateStock update = new UpdateStock();
setVisible(false);
} else if (e.getSource() == price) {
UpdatePrice update = new UpdatePrice();
setVisible(false);
} else if (e.getSource() == delete) {
UpdateDelete d = new UpdateDelete();
setVisible(false);
}
//once clicked, displays different classes
}
}
```

List

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package stock;

import javax.swing.JFrame;

/**
*
* @author ub2232e
*/
public class List extends javax.swing.JFrame {

/**
* Creates new form List
*/
public List() {
initComponents();
setVisible(true);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

/**
* This method is called from within the constructor to initialise the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
```

```
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {
bindingGroup = new org.jdesktop.beansbinding.BindingGroup();

UserDBPUEntityManager = java.beans.Beans.isDesignTime() ? null :
javax.persistence.Persistence.createEntityManagerFactory("UserDBPU").createEntityManager();
stockQuery = java.beans.Beans.isDesignTime() ? null :
UserDBPUEntityManager.createQuery("SELECT s FROM Stock s");
stockList = java.beans.Beans.isDesignTime() ? java.util.Collections.emptyList() :
stockQuery.getResultList();
stockQuery1 = java.beans.Beans.isDesignTime() ? null :
UserDBPUEntityManager.createQuery("SELECT s FROM Stock s");
stockList1 = java.beans.Beans.isDesignTime() ? java.util.Collections.emptyList() :
stockQuery1.getResultList();
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

org.jdesktop.swingbinding.JTableBinding jTableBinding =
org.jdesktop.swingbinding.SwingBindings.createJTableBinding(org.jdesktop.beansbinding.AutoBindi
ng.UpdateStrategy.READ_WRITE, stockList1, jTable1);
org.jdesktop.swingbinding.JTableBinding.ColumnBinding columnBinding =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${stockkey}"));
columnBinding.setColumnName("Stock Key");
columnBinding.setColumnClass(String.class);
columnBinding =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${stockname}"));
columnBinding.setColumnName("Stock Name");
columnBinding.setColumnClass(String.class);
```

```
columnBinding =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${stockprice}"));

columnBinding.setColumnName("Stock Price");

columnBinding.setColumnClass(java.math.BigInteger.class);

columnBinding =
jTableBinding.addColumnBinding(org.jdesktop.beansbinding.ELProperty.create("${stockquantity}"));

columnBinding.setColumnName("Stock Quantity");

columnBinding.setColumnClass(Integer.class);

bindingGroup.addBinding(jTableBinding);

jTableBinding.bind();

jScrollPane1.setViewportView(jTable1);


jLabel1.setFont(new java.awt.Font("Rod", 1, 14)); // NOI18N

jLabel1.setText("Current Stock");


javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(186, 186, 186)

.addComponent(jLabel1))

.addGroup(layout.createSequentialGroup()

.addGap(43, 43, 43)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 375,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap(53, Short.MAX_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
.addContainerGap(16, Short.MAX_VALUE)
.addComponent(jLabel1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 275,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(35, 35, 35))
);

bindingGroup.bind();

pack();
}// </editor-fold>

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
/* Set the Nimbus look and feel */
//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
/* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
*/
try {
for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
if ("Nimbus".equals(info.getName())) {
javax.swing.UIManager.setLookAndFeel(info.getClassName());
break;
}
}
} catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new List().setVisible(true);
}
});
}

// Variables declaration - do not modify
private javax.persistence.EntityManager UserDBPUEntityManager;
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private java.util.List<stock.Stock> stockList;
private java.util.List<stock.Stock> stockList1;
private javax.persistence.Query stockQuery;
private javax.persistence.Query stockQuery1;
private org.jdesktop.beansbinding.BindingGroup bindingGroup;
```

```
// End of variables declaration
}
```

Login1

```
package stock;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Login1 extends JFrame implements ActionListener {

JTextField username = new JTextField(7);
JPasswordField password = new JPasswordField(7);
JButton login = new JButton("Login");

public Login1() {
setLayout(new BorderLayout());
setSize(420, 70);
```

```
setTitle("Login");
JPanel top = new JPanel();
top.add(new JLabel("Username"));
top.add(username);
top.add(new JLabel("Password"));
top.add(password);
top.add(login);
login.addActionListener(this);
add("North", top);
setLocationRelativeTo(null);
setResizable(false);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {

if (e.getSource() == login) {
try {
//details below connected to database
String host = "jdbc:derby://localhost:1527/UserDB";
String j = "use";
String p = "pass";
Connection con = DriverManager.getConnection(host, j, p);
Statement stmt = con.createStatement();
String SQL = "SELECT * FROM USE.LOGIN";
ResultSet rs = stmt.executeQuery(SQL);

String z = username.getText();
String i = password.getText();
```

```
rs.next();
String s = rs.getString("username");
String w = rs.getString("password");

if (z.matches(s) && i.matches(w)) {
display("Login Successful!");
TwoButtons update = new TwoButtons();
setVisible(false);
//closes the window once the Login is successful
} else {
display("Wrong details!");
username.setText("");
password.setText("");
//clears once the user enters the wrong details
}
} catch (SQLException evt) {
display(evt.getMessage());
//method used here
}
}
}

private static void display(String s) {
JOptionPane.showMessageDialog(null, s);
//uses method to shorten code
}
}
```

Master

```
package stock;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Desktop;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class Master extends JFrame implements ActionListener {

JButton check = new JButton("Check Item");
JButton list = new JButton("Stock List");
JButton purchase = new JButton("Purchase Item");
JButton stock = new JButton("Update Stock");
JButton quit = new JButton("Exit");
JButton help = new JButton("Help and Support");

public static void main(String[] args) {
Master master = new Master();
}
```

```
public Master() {
setLayout(new BorderLayout());
setSize(700, 150);
setTitle("Master");
JPanel top = new JPanel();
JLabel heading = new JLabel("@Choice");
heading.setFont(new Font("Chiller", Font.PLAIN, 40));
top.add(heading);
heading.setForeground(Color.DARK_GRAY);
top.setBackground(Color.WHITE);
add("North", top);
JPanel middle = new JPanel();
middle.add(new JLabel("Select an option by clicking one of the buttons below"));
add("Center", middle);
JPanel bottom = new JPanel();
bottom.add(list);
list.addActionListener(this);
bottom.add(check);
check.addActionListener(this);
bottom.add(purchase);
purchase.addActionListener(this);
bottom.add(stock);
stock.addActionListener(this);
bottom.add(help);
help.addActionListener(this);
bottom.add(quit);
quit.addActionListener(this);
add("South", bottom);
setLocationRelativeTo(null);
setResizable(false);
setVisible(true);
```

```
}

@Override
public void actionPerformed(ActionEvent e) {
if (e.getSource() == check) {
CheckStock checkStock = new CheckStock();
} else if (e.getSource() == purchase) {
PurchaseItem purchaseItem = new PurchaseItem();
} else if (e.getSource() == stock) {
Login1 Login = new Login1();
} else if (e.getSource() == list) {
List list = new List();
} else if (e.getSource() == help) {
if (e.getSource() == help) {
if (Desktop.isDesktopSupported()) {
Desktop desktop = Desktop.getDesktop();

try {
desktop.edit(new File("Guide.txt"));
} catch (Exception ex) {
System.out.println(ex);
}
} else {
System.out.println("Desktop is not supported");
}
}
} else if (e.getSource() == quit) {
System.exit(0);
}
}
}
```

PurchaseItem

```
package stock;

import javax.swing.JOptionPane;
import javax.swing.*;
import java.awt.BorderLayout;
import static java.awt.Color.white;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.TextArea;
import java.awt.event.ActionEvent;
import java.io.File;
import java.text.DecimalFormat;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class PurchaseItem extends JFrame implements ActionListener {

JTextField stockNo = new JTextField(7);
JTextField number = new JTextField(7);
TextArea information = new TextArea(4, 70);
JButton purchase = new JButton("Purchase");
JButton check = new JButton("Check Stock");
DecimalFormat pounds = new DecimalFormat("£#,##0.00");
JComboBox drop = new JComboBox();
double totalCost = 0.0;
```

```
JLabel imageLab = new JLabel(new ImageIcon("images.empty.jpg"));

public PurchaseItem() {
setLayout(new BorderLayout());
setBounds(100, 100, 700, 200);
setTitle("Purchase Item");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
JPanel top = new JPanel();
top.add(new JLabel("Item: "));
top.add(stockNo);
top.add(check);
check.addActionListener(this);
top.add(new JLabel("How many: "));
top.add(number);
top.add(purchase);
purchase.addActionListener(this);
add("North", top);
JPanel middle = new JPanel();
middle.add(information);
middle.add(imageLab);
add("Center", middle);
JPanel bottom = new JPanel();
add("South", bottom);
setResizable(false);
setVisible(true);

//allows the user not to edit the textfield
information.setEditable(false);
information.setBackground(white);
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
String key = stockNo.getText();
String name = StockData.getName(key);
if (name == null) {
information.setText("No such item in stock");
stockNo.setText("");
}
else if (e.getSource() == check) {
information.setText(name);
information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
information.append("\nNumber in stock: " + StockData.getQuantity(key));

//uses key to add images
String image = "Images\\" + key + ".jpg";
File imageFile = new File(image);
if (!imageFile.exists()) {
image = "Images\\empty.jpg";
}
imageLab.setIcon(new ImageIcon(image));
}
else if (e.getSource() == purchase) {
int numberOfItems = Integer.parseInt(number.getText());
double cost = numberOfItems * StockData.getPrice(key);
if (numberOfItems <= StockData.getQuantity(key)) {
information.append("\n" + name + " " + "x" + " " + numberOfItems);
StockData.update(key, -numberOfItems);
totalCost += cost;
information.append("\nNumber in Stock:" + StockData.getQuantity(key));
JOptionPane.showMessageDialog(null, "\nTotal cost will be:" + " " +
pounds.format(totalCost));
```

```
information.append("\n");
information.append("\nThis purchase has been successful");
} else {
JOptionPane.showMessageDialog(null, "Sorry, we are out of stock! Please, try later!");
}
} else {
JOptionPane.showMessageDialog(null, "Please enter key!");
}
information.append("\n");
}
}
```

StockData

```
package stock;

// Skeleton version of StockData.java that links to a database.
// NOTE: You should not have to make any changes to the other
// Java GUI classes for this to work, if you complete it correctly.
// Indeed these classes shouldn't even need to be recompiled
import java.sql.*; // DB handling package
import java.io.*;
import org.apache.derby.drda.NetworkServerControl;

public class StockData {

private static Connection connection;
private static Statement stmt;

static {
// standard code to open a connection and statement to an Access database
try {
```

```
NetworkServerControl server = new NetworkServerControl();
server.start(null);
// Load JDBC driver
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
//Establish a connection
String sourceURL = "jdbc:derby://localhost:1527/"
+ new File("UserDB").getAbsolutePath() + ";";
Connection UserDB = DriverManager.getConnection(sourceURL, "use", "pass");
stmt = UserDB.createStatement();
} catch (ClassNotFoundException cnfe) {
System.out.println(cnfe);
} catch (SQLException sqle) {
System.out.println(sqle);
} catch (Exception e) {
System.out.println(e);
}
}

// You could make methods getName, getPrice and getQuantity simpler by using an auxiliary
// private String method getField(String key, int fieldNo) to return the appropriate field as a String

public static String getName(String key) {
try {
// Need single quote marks ' around the key field in SQL. This is easy to get wrong!
// For instance if key was "11" the SELECT statement would be:
// SELECT * FROM Stock WHERE stockKey = '11'
ResultSet res = stmt.executeQuery("SELECT * FROM Stock WHERE stockKey = '" + key + "'");
if (res.next()) { // there is a result
// the name field is the second one in the ResultSet
// Note that with  ResultSet we count the fields starting from 1
return res.getString(2);
```

```
} else {
return null;
}

} catch (SQLException e) {
System.out.println(e);
return null;
}
}

public static double getPrice(String key) {
try {
// Similar to getName. If no result, return -1.0
ResultSet rs = stmt.executeQuery("SELECT * FROM Stock WHERE stockKey = '" + key + "'");
if (rs.next()) {
return rs.getDouble(3);
} else {
return -1.0;
}
} catch (SQLException ex) {
System.out.println(ex);
return -1.0;
}
}

public static int getQuantity(String key) {
// Similar to getName. If no result, return -1
try {
ResultSet rs = stmt.executeQuery("SELECT * FROM Stock WHERE stockKey = '" + key + "'");
if (rs.next()) {
return rs.getInt(4);
```

```
} else {
return -1;
}
} catch (SQLException ex) {
System.out.println(ex);
return -1;
}
}

// update stock levels
// extra is +ve if adding stock
// extra is -ve if selling stock
public static void update(String key, int extra) {
// SQL UPDATE statement required. For instance if extra is 5 and stockKey is "11" then updateStr is
// UPDATE Stock SET stockQuantity = stockQuantity + 5 WHERE stockKey = '11'
String updateStr = "UPDATE Stock SET stockQuantity = stockQuantity + " + extra + " WHERE stockKey = '" + key + "'";
System.out.println(updateStr);
try {
stmt.executeUpdate(updateStr);
} catch (SQLException e) {
System.out.println(e);
}
}
public static void priceUpdatePlus(String key, double extra) {
String updateStr = "UPDATE Stock SET stockPrice = stockPrice + " + extra + " WHERE stockKey = '" + key + "'";
System.out.println(updateStr);
try {
stmt.executeUpdate(updateStr);
} catch (SQLException e) {
```

```
System.out.println(e);
}
}
public static void priceUpdateMinus(String key, double extra) {
String updateStr = "UPDATE Stock SET stockPrice = stockPrice + " + extra + " WHERE stockKey = '" + key + "'";
System.out.println(updateStr);
try {
stmt.executeUpdate(updateStr);
} catch (SQLException e) {
System.out.println(e);
}
}
// close the database
public static void close() {
try {
connection.close();
} catch (SQLException e) {
// this shouldn't happen
System.out.println(e);
}
}
}
```

TwoButtons

```
package stock;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
```

```
import javax.swing.JFrame;
import javax.swing.JPanel;

public class TwoButtons extends JFrame implements ActionListener {

JButton update = new JButton("Update Existing");
JButton updateTwo = new JButton("Update New");

public TwoButtons() {
setLayout(new BorderLayout());
setSize(360, 150);
setTitle("Update");
JPanel top = new JPanel();
add("North", top);

JPanel bottom = new JPanel();
bottom.add(update);
update.addActionListener(this);
update.setPreferredSize(new Dimension(150, 100));
bottom.add(updateTwo);
updateTwo.addActionListener(this);
updateTwo.setPreferredSize(new Dimension(150, 100));
add("South", bottom);
setLocationRelativeTo(null);
setResizable(false);
setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
if (e.getSource() == update) {
ExistingButtons edit = new ExistingButtons();
```

```
} else if (e.getSource() == updateTwo) {
newStock n = new newStock();

}
}
}
```

UpdateDelete

```
package stock;
//import java.awt.*;

import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DecimalFormat;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import org.apache.derby.drda.NetworkServerControl;

/**
*
```

```
* @author ub2232e
*/
public class UpdateDelete extends JFrame
implements ActionListener {

JTextField stockNo = new JTextField(7);
JButton checkBtn = new JButton("Check Stock");
JButton deleteBtn = new JButton("Delete");
DecimalFormat pounds = new DecimalFormat("£#,##0.00");
JTextArea information = new JTextArea(3, 25);
private static Connection connection;
private static Statement stmt;

static {
// standard code to open a connection and statement to an Access database
try {
NetworkServerControl server = new NetworkServerControl();
server.start(null);
// Load JDBC driver
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
//Establish a connection
String sourceURL = "jdbc:derby://localhost:1527/"
+ new File("UserDB").getAbsolutePath() + ";";
Connection UserDB = DriverManager.getConnection(sourceURL, "use", "pass");
stmt = UserDB.createStatement();
} catch (ClassNotFoundException cnfe) {
System.out.println(cnfe);
} catch (SQLException sqle) {
System.out.println(sqle);
} catch (Exception e) {
System.out.println(e);
```

```
}
}

public UpdateDelete() {
setLayout(new BorderLayout());
setBounds(200, 200, 500, 150);
setTitle("Update Delete");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

JPanel top = new JPanel();
top.add(new JLabel("Enter Stock Number:"));
top.add(stockNo);
top.add(checkBtn);
checkBtn.addActionListener(this);
add("North", top);
JPanel middle = new JPanel();
middle.add(information);
add("Center", middle);
JPanel bottom = new JPanel();
bottom.add(deleteBtn);
deleteBtn.addActionListener(this);
add("South", bottom);

setResizable(false);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
String key = stockNo.getText();
String name = StockData.getName(key);
```

```
if (name == null) {
information.setText("No such item in stock");
} else if (e.getSource() == checkBtn) {
information.setText(name);
information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
information.append("\nQuantity: " + StockData.getQuantity(key));
}

else if (e.getSource() == deleteBtn) {
if (name != null) {
int confirmOption = JOptionPane.showConfirmDialog(null, "Are you sure you want to delete this item?");
if (confirmOption == JOptionPane.YES_OPTION) {
try {
String sql = "DELETE FROM Stock WHERE stockKey = '" + key + "'";
stmt.execute(sql);
JOptionPane.showMessageDialog(null, "You have successfully deleted this item.");
} catch (Exception ea) {
System.out.println(ea);
}
}
if (confirmOption == JOptionPane.NO_OPTION) {
JOptionPane.showMessageDialog(null, "Please try again to attempt to delete this item from stock");
}
}
}
}
}
```

UpdatePrice

```
package stock;
//import java.awt.*;
```

```
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.text.DecimalFormat;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

/**
*
* @author ub2232e
*/
public class UpdatePrice extends JFrame
        implements ActionListener {

JTextField stockNo = new JTextField(7);
JTextField plus = new JTextField(7);
JTextField minus = new JTextField(7);
JButton check = new JButton("Check Stock");
JButton update = new JButton("Add");
JButton updateTwo = new JButton("Minus");
DecimalFormat pounds = new DecimalFormat("£#,##0.00");
JTextArea information = new JTextArea(3, 25);
```

```
public UpdatePrice() {
setLayout(new BorderLayout());
setBounds(200, 200, 550, 150);
setTitle("Update Price");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

JPanel top = new JPanel();
top.add(new JLabel("Enter Stock Number:"));
top.add(stockNo);
top.add(check);
check.addActionListener(this);

add("North", top);
JPanel middle = new JPanel();
middle.add(information);
add("Center", middle);
JPanel bottom = new JPanel();
bottom.add(new JLabel("Increasing Price:"));
bottom.add(plus);
bottom.add(update);
update.addActionListener(this);
bottom.add(new JLabel("Decreasing Price:"));
bottom.add(minus);
bottom.add(updateTwo);
updateTwo.addActionListener(this);

add("South", bottom);
setResizable(false);
setVisible(true);
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
String key = stockNo.getText();
String name = StockData.getName(key);

if (name == null) {
information.setText("No such item in stock");
} else if (e.getSource() == check) {
information.setText(name);
information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
information.append("\nQuantity: " + StockData.getQuantity(key));
} else if (e.getSource() == update) {
information.setText(name);
double plusDouble = Double.parseDouble(plus.getText());
StockData.priceUpdatePlus(key, plusDouble);
information.append("\nPrice: " + pounds.format(StockData.getPrice(key)));
} else if (e.getSource() == updateTwo) {
information.setText(name);
double minusDouble = Double.parseDouble(minus.getText());
StockData.priceUpdateMinus(key, -minusDouble);
information.append("\nPrice: " + StockData.getPrice(key));
}
}
}
```

UpdateStock

```
package stock;

import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JTextField;
```

```
import java.awt.TextArea;
import java.awt.event.ActionEvent;
import java.text.DecimalFormat;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
*
* @author ub2232e
*/
public class UpdateStock extends JFrame
implements ActionListener {

JTextField stockNo = new JTextField(7);
JTextField update = new JTextField(7);
TextArea information = new TextArea(10, 50);
JButton check = new JButton("Check Stock");
JButton add = new JButton("Add");
DecimalFormat pounds = new DecimalFormat("£#,##0.00");

public UpdateStock() {
setLayout(new BorderLayout());
setBounds(200, 200, 700, 150);
setTitle("Update Stock");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
JPanel top = new JPanel();
top.add(new JLabel("Enter Stock Number:"));
top.add(stockNo);
top.add(check);
check.addActionListener(this);
```

```
top.add(new JLabel("Quantity:"));
top.add(update);
top.add(add);
add.addActionListener(this);
add("North", top);
JPanel middle = new JPanel();
middle.add(information);
add("Center", middle);
JPanel bottom = new JPanel();
add("South", bottom);

setResizable(false);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
String key = stockNo.getText();
String name = StockData.getName(key);

if (name == null) {
information.setText("No such item in stock");
} else if (e.getSource() == check) {
information.setText(name);
information.append("\nCurrently in Stock: " + StockData.getQuantity(key));
} else if (e.getSource() == add) {
information.setText(name);
int quantity = Integer.parseInt(update.getText());
StockData.update(key, quantity);
information.append("\nCurrently in Stock: " + StockData.getQuantity(key));
}
```

```
    }
}
```

newStock

```
package stock;
//import java.awt.*;

import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

/**
 *
 * @author ub2232e
 */
public class newStock extends JFrame
        implements ActionListener {

    JTextField jKey = new JTextField(7);
    JTextField jName = new JTextField(7);
    JTextField jPrice = new JTextField(7);
    JTextField jQuantity = new JTextField(7);
```

```
JButton jButton = new JButton("Add");

public newStock() {
setLayout(new BorderLayout());
setBounds(100, 100, 700, 80);
setTitle("Adding new Stock");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
JPanel top = new JPanel();
top.add(new JLabel("Key:"));
top.add(jKey);
top.add(new JLabel("Name:"));
top.add(jName);
top.add(new JLabel("Price:"));
top.add(jPrice);
top.add(new JLabel("Quantity:"));
top.add(jQuantity);
top.add(jButton);
jButton.addActionListener(this);
add("North", top);
setResizable(false);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
String keyStr = jKey.getText();
String nameStr = jName.getText();
double priceDouble = Double.parseDouble(jPrice.getText());
int quantityInt = Integer.parseInt(jQuantity.getText());

if (e.getSource() == jButton) {
```

```
try {
//connecting database
String host = "jdbc:derby://localhost:1527/UserDB";
String j = "use";
String p = "pass";

Connection con = DriverManager.getConnection(host, j, p);
PreparedStatement stmt = (PreparedStatement) con.prepareStatement("INSERT INTO
STOCK(stockKey, stockName, stockPrice, stockQuantity) "
+ "VALUES(?,?,?,?)");
stmt.setString(1, keyStr);
stmt.setString(2, nameStr);
stmt.setDouble(3, priceDouble);
stmt.setInt(4, quantityInt);

//executing the statement
stmt.executeUpdate();
JOptionPane.showMessageDialog(null, "You have successfully added a new item");

//clearing all textfields once complete
jKey.setText("");
jName.setText("");
jPrice.setText("");
jQuantity.setText("");

} catch (Exception ea) {
JOptionPane.showMessageDialog(null, ea);
}
}
}
}
```