



**Individual Assignment**  
**Technology Park Malaysia**  
**AAPP004-4-2-JP**  
**Java Programming**

Hand Out Date: 22<sup>nd</sup> January 2021

Hand In Date: 16<sup>th</sup> April 2021

Weightage: 70%

---

Instructions to Candidates:

1. Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).
2. Late submission will be awarded (0) unless Extenuating Circumstances (EC) are upheld.
3. Cases of plagiarism will be penalised.
4. The assignment should be bound in an appropriate style (comb bound or stapled).
5. Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.

# **Table of Contents**

1.0 Sample Outputs of Program.....	3
1.1 Customer.....	3
1.1.1 Menu .....	3
1.1.2 Payment Page.....	4
1.1.3 Message Box.....	5
1.2 Staff.....	6
1.2.1 Login Page .....	6
1.2.2 Message Box after Login Button .....	7
1.2.3 Main Page .....	8
1.2.4 View Products.....	9
1.2.5 Editing or Uploading New Products .....	11
1.2.6 Restocking Products.....	12
1.2.7 Exit.....	13
2.0 Objected-Oriented Programming.....	14
2.1 Abstract.....	14
2.2 Encapsulation.....	15
2.3 Inheritance.....	17
2.4 Polymorphism .....	19
3.0 Additional Features .....	20
3.1 To View Data from Text File.....	20
3.2 Searching for a Data with a Textbox .....	21
3.3 Restricting the Text Box with the Data Type It Is Allowed (Price) .....	22
3.4 Restricting the Customers from Buying Products with No Stock .....	23
3.5 Deducting Stock from the System as Customers Buys.....	25
3.6 Editing Table Header .....	27
4.0 Assumptions.....	28

## 1.0 Sample Outputs of Program

### 1.1 Customer

#### 1.1.1 Menu

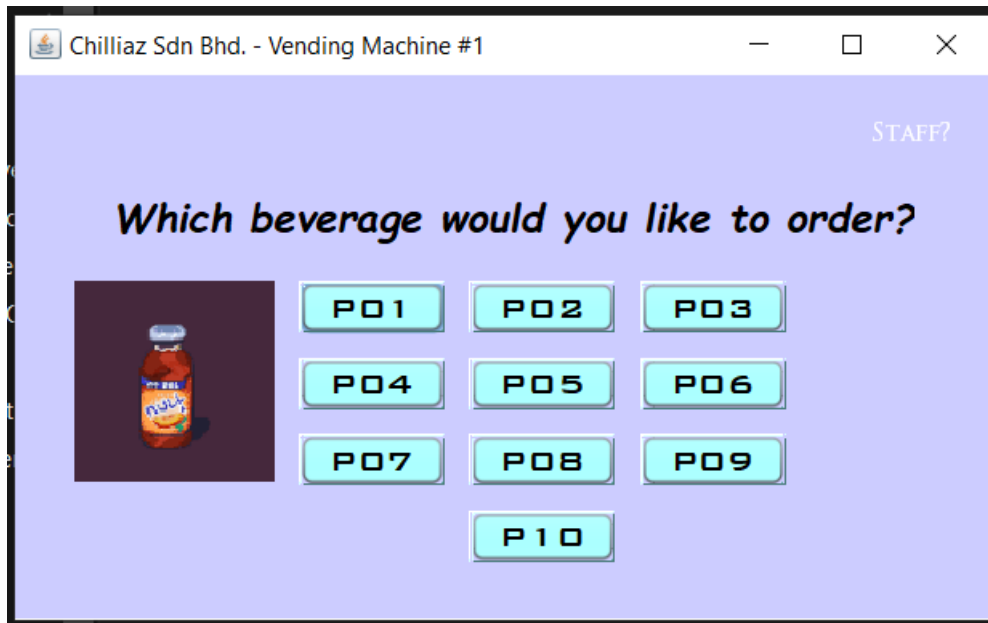


Diagram 1: The Menu Page

This is the page where the customer will first see, where they can select the buttons to order the product that they want.

At the top right corner, there is a label written "Staff?". That label is linked to the staff's login page once clicked.

### 1.1.2 Payment Page



Diagram 2: The Details of the Order

At this page, the customers are shown with the details of their order. They are supposed to enter the money or select cancel at this page. When the cancel button is selected, they will be directed back to the menu.

### **1.1.3 Message Box**

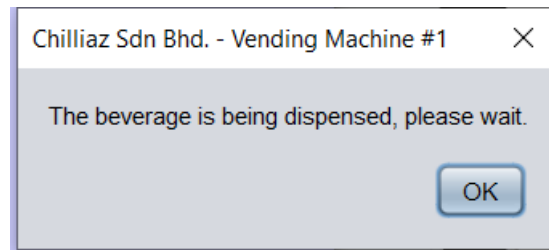


Diagram 3: Message box after payment

After payment, a message box will pop up to let the customer to know to wait until the product is dispensed from the machine.

## 1.2 Staff

### 1.2.1 Login Page

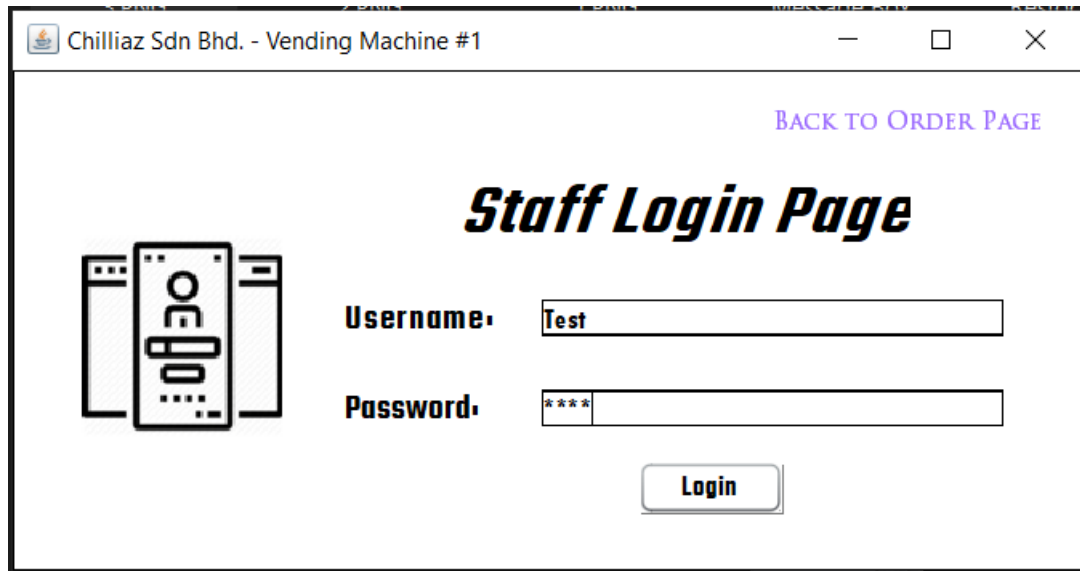


Diagram 4: Login Page

This is the page where the staff is supposed to insert the login credentials that are set. After input the correct username and password, select the login button to browse the actions that are available.

At the top right, there is a label "Back to Order Page". That label will direct the user back to the menu.

### **1.2.2 Message Box after Login Button**

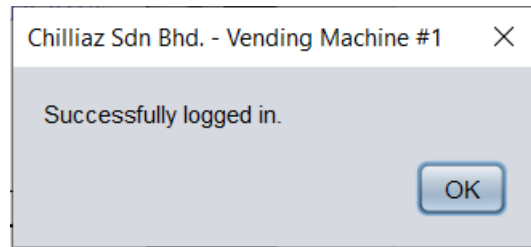


Diagram 5: Message box after inserting correct login credentials

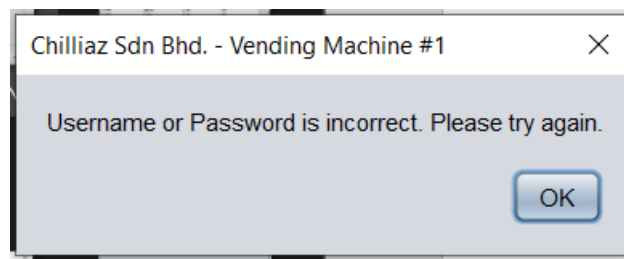


Diagram 6: Message box after inserting incorrect login credentials

### 1.2.3 Main Page

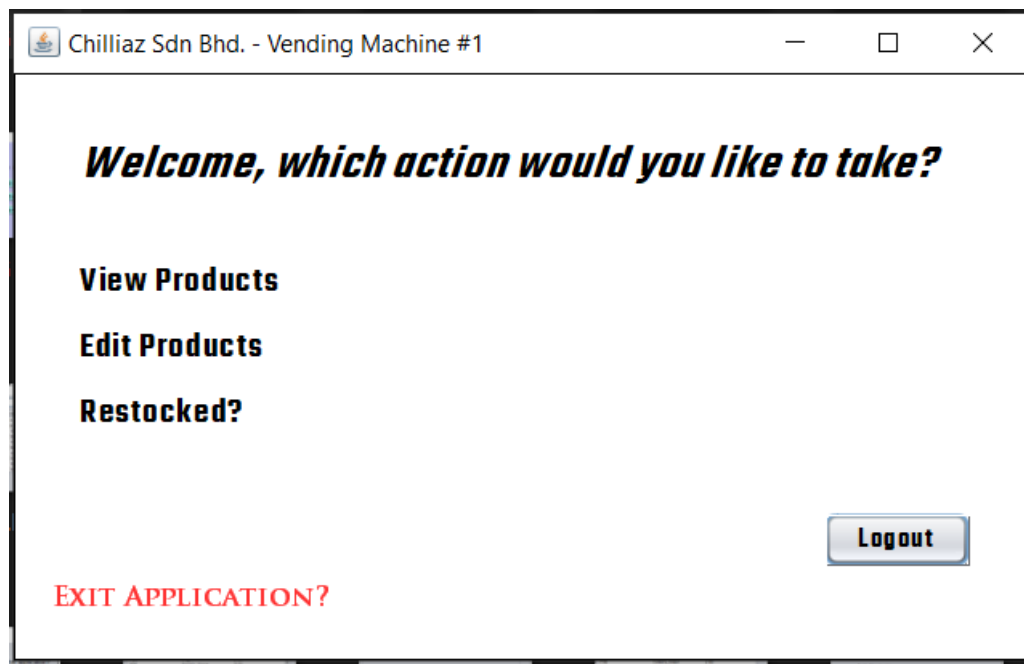


Diagram 7: Main Page with List of Actions that can be taken

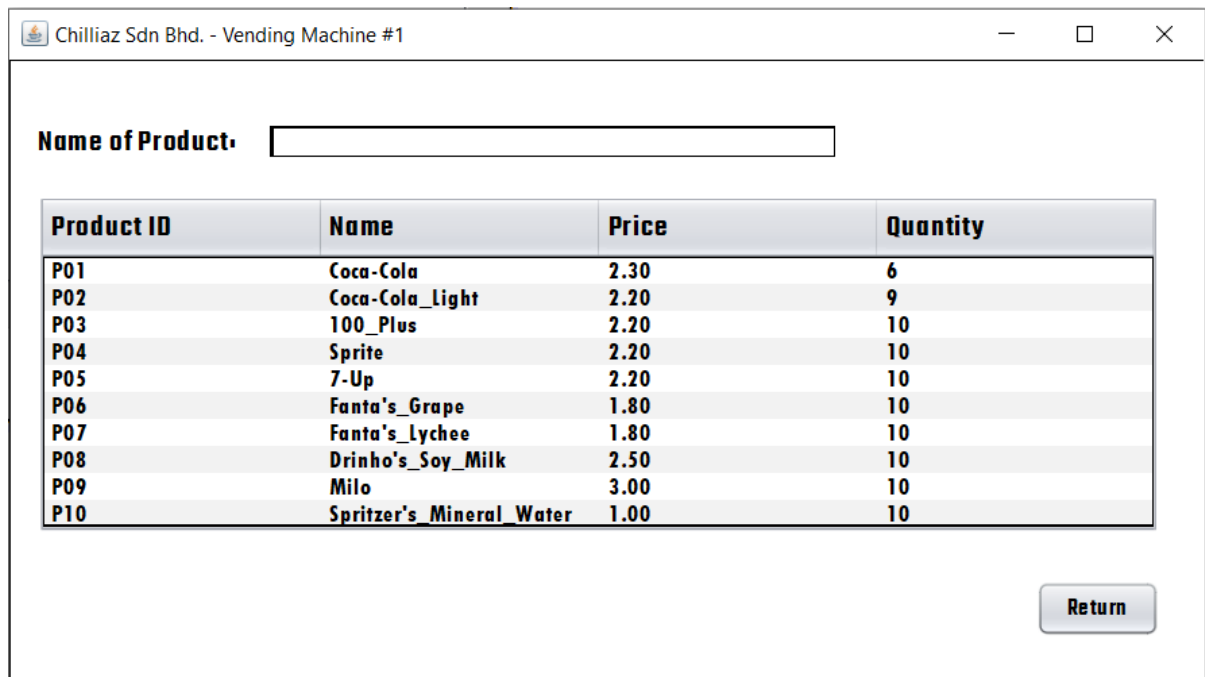
There are 5 actions that can be taken in this page. First, the “View Products” label. It will direct to the page to view the list of products available. Then, the “Edit Products” label. It will direct to the page to edit the product details. There is also the “restocked?” label. It will direct to the page to restock the chosen products.

At the right, there is a button with the name “Logout”. After clicking, the staff will be logged out and be directed back to the login page.

At the bottom left, there is a “Exit Application?” label. The label will close down the program once clicked.



## 1.2.4 View Products



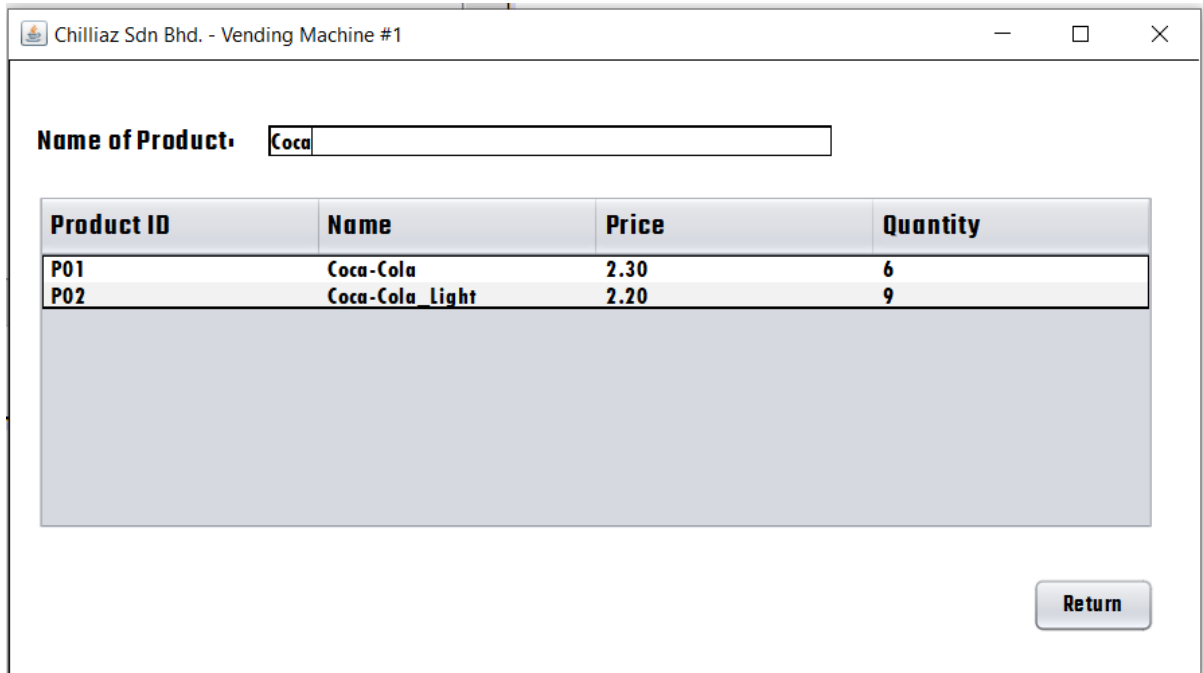
**Name of Product:**

Product ID	Name	Price	Quantity
P01	Coca-Cola	2.30	6
P02	Coca-Cola_Light	2.20	9
P03	100_Plus	2.20	10
P04	Sprite	2.20	10
P05	7-Up	2.20	10
P06	Fanta's_Grape	1.80	10
P07	Fanta's_Lychee	1.80	10
P08	Drinho's_Soy_Milk	2.50	10
P09	Milo	3.00	10
P10	Spritzer's_Mineral_Water	1.00	10

**Return**

Diagram 8: The List of Products

There is a table that displays all of the products details. At the top, there is a text box that is placed to search products. At the bottom right, there is also a button “Return” that guides the user back to the main page.



Product ID	Name	Price	Quantity
P01	Coca-Cola	2.30	6
P02	Coca-Cola Light	2.20	9

Diagram 9: The Search Function

To search, simply type in the text box, it will display the products that the user searched in the table.

## 1.2.5 Editing or Uploading New Products

The screenshot shows a window titled "Chilliaz Sdn Bhd. - Vending Machine #1". Inside, there is a table with four columns: Product ID, Name, Price, and Quantity. The table contains five rows of data. Below the table, there is a form for editing a product. The "Product ID" field is set to "P01". The "Name" field contains "Coca-Cola", the "Price" field contains "2.30", and the "Quantity" field contains "5". There are "Update" and "Return" buttons at the bottom of the form.

Product ID	Name	Price	Quantity
P01	Coca-Cola	2.20	5
P02	Coca-Cola_light	2.20	9
P03	100_Plus	2.20	10
P04	Sprite	2.20	10
P05	7-Up	2.20	10

Product ID: P01

Name: Coca-Cola

Price: 2.30

Quantity: 5

Update

Return

Diagram 10: The List to Edit

To edit or upload any product, select the chosen product ID by clicking on the product ID in the table. Then, change the name, price and quantity to the desired data. Lastly, select the “Update” button to successfully update the product details.

At the bottom right, the button “Return” will direct back to the main page.

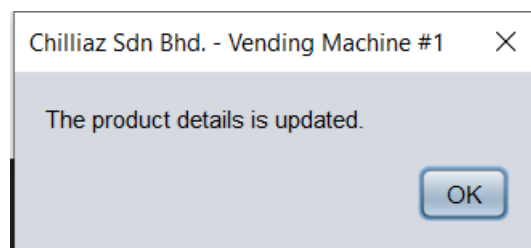
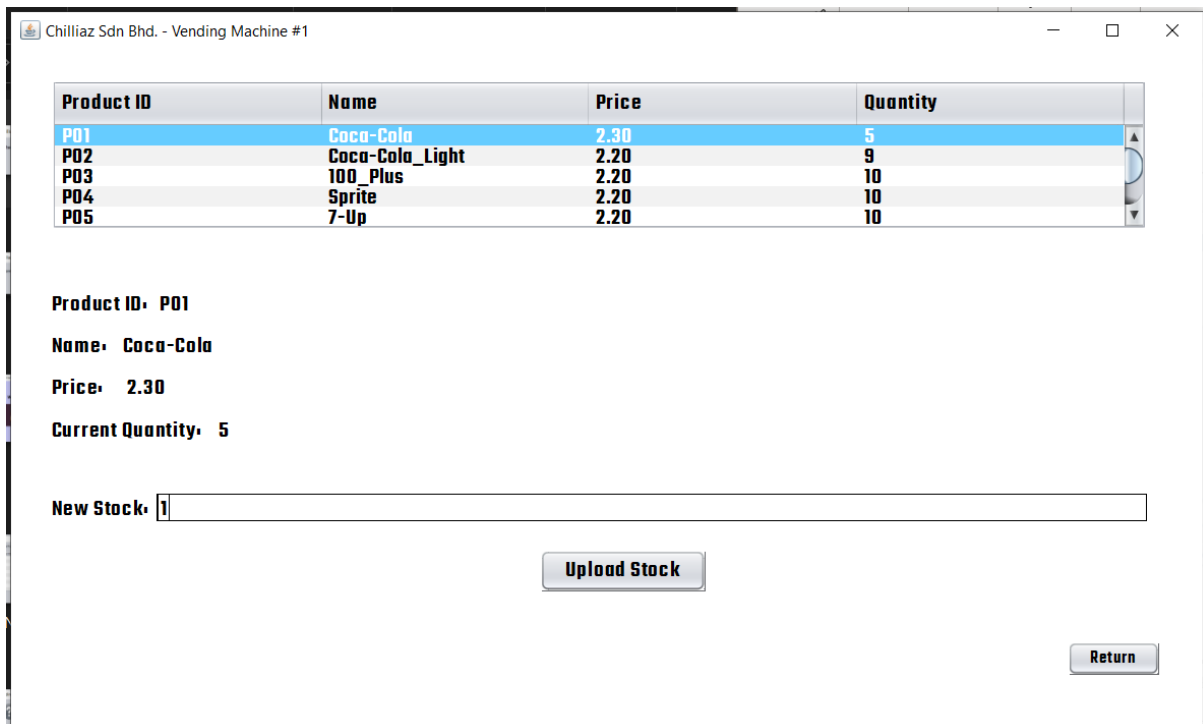


Diagram 11: Message Box after Updated

## 1.2.6 Restocking Products



The screenshot shows a window titled "Chilliaz Sdn Bhd. - Vending Machine #1". It contains a table with the following data:

Product ID	Name	Price	Quantity
P01	Coca-Cola	2.30	5
P02	Coca-Cola_Light	2.20	9
P03	100_Plus	2.20	10
P04	Sprite	2.20	10
P05	7-Up	2.20	10

Below the table, the selected product details are displayed:

Product ID: P01  
Name: Coca-Cola  
Price: 2.30  
Current Quantity: 5

A "New Stock" label is followed by a text input field containing the value "1".

At the bottom center is an "Upload Stock" button, and at the bottom right is a "Return" button.

Diagram 12: The Page to Restock

To restock any product, select the chosen product ID by clicking on the product ID in the table. Then, input the additional stock amount into the textbox provided. Lastly, select the “Update Stock” button to successfully update the product’s stock.

At the bottom right, the button “Return” will direct back to the main page.

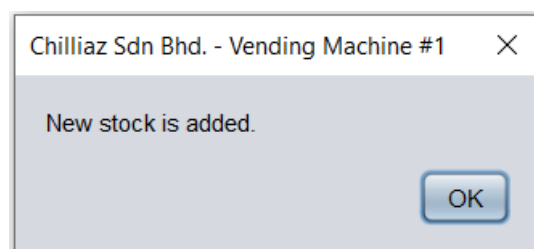


Diagram 13: Message Box after Restocked

### **1.2.7 Exit**

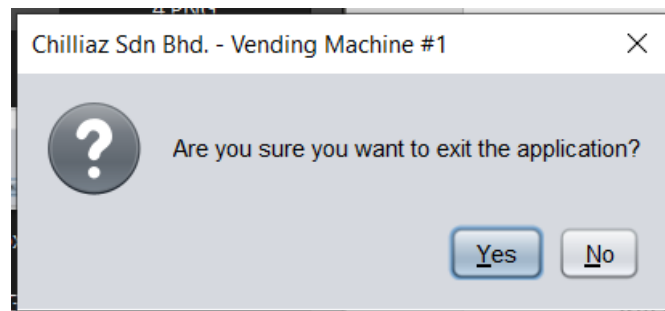


Diagram 14: Message Box to Confirm before Exit

If the user wants to exit the program, they should select “Yes”. Else, the “No” button will direct them back to the main page.

## **2.0 Objected-Oriented Programming**

### **2.1 Abstract**

```
public class OrderPage extends javax.swing.JFrame {  
  
    public static String PID;  
  
    public static String Name;  
  
    public static String Price;  
  
    public static String Quantity;}  

```

The lines of codes above is from OrderPage.

```
void FillData()  
  
    {  
  
        lblName.setText(OrderPage.Name);  
  
        lblStatedPrice.setText(OrderPage.Price);  
  
    }  

```

The lines of codes above are from OrderDetailsPage.

The variables are set public and data is stored into it. The data in the variables are then taken from another page.

## **2.2 Encapsulation**

```
private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String ProductID = lblPID.getText();  
  
    DefaultTableModel Table = (DefaultTableModel)tableView.getModel();  
  
    int SelectedRow = tableView.getSelectedRow();  
  
    if(SelectedRow >= 0) {  
  
        String Name = txtName.getText();  
  
        String Price = txtPrice.getText();  
  
        String Quantity = txtQuantity.getText();  
  
        Table.setValueAt (ProductID, SelectedRow, 0);  
  
        Table.setValueAt (Name, SelectedRow, 1);  
  
        Table.setValueAt (Price, SelectedRow, 2);  
  
        Table.setValueAt (Quantity, SelectedRow, 3);  
  
        JOptionPane.showMessageDialog(null, "The product details is updated.", "Chilliaz Sdn  
Bhd. - Vending Machine #1", JOptionPane.PLAIN_MESSAGE);  
  
    }  
  
    Export();  
  
}
```

The variable are stored with data, then used again in the same method using getter and setter method.



## **2.3 Inheritance**

```
void Export()
```

```
{
```

```
    String filePath = "D:\\APU Stuff\\Semester 5\\Java Programming (AAPP004-4-2-JP)\\Mr. Usman Hashmi\\Assignment\\ProductsInformation.txt";
```

```
    File file = new File(filePath);
```

```
    try {
```

```
        FileWriter fw = new FileWriter(file);
```

```
        BufferedWriter bw = new BufferedWriter(fw);
```

```
        for(int i = 0; i < tableView.getRowCount(); i++){
```

```
            for(int j = 0; j < tableView.getColumnCount(); j++){
```

```
                bw.write(tableView.getValueAt(i, j).toString()+" ");
```

```
            }
```

```
            bw.newLine();
```

```
        }
```

```
        bw.close();
```

```
        fw.close();
```

```
    } catch (IOException ex) {
```

```
        Logger.getLogger(EditProductPage.class.getName()).log(Level.SEVERE, null, ex);
```

```
    }
```

```
}
```

The method “Export” is set as public in a page.

```
private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String ProductID = lblPID.getText();  
    DefaultTableModel Table = (DefaultTableModel)tableView.getModel();  
  
    int SelectedRow = tableView.getSelectedRow();  
  
    if(SelectedRow >= 0) {  
  
        String Name = txtName.getText();  
        String Price = txtPrice.getText();  
        String Quantity = txtQuantity.getText();  
  
        Table.setValueAt (ProductID, SelectedRow, 0);  
        Table.setValueAt (Name, SelectedRow, 1);  
        Table.setValueAt (Price, SelectedRow, 2);  
        Table.setValueAt (Quantity, SelectedRow, 3);  
  
        JOptionPane.showMessageDialog(null, "The product details is updated.", "Chilliaz Sdn  
Bhd. - Vending Machine #1", JOptionPane.PLAIN_MESSAGE);  
  
    }  
    Export();  
}
```

The method is then later used in another method.

## **2.4 Polymorphism**

The concept “polymorphism” is not used in this system.

## **3.0 Additional Features**

### **3.1 To View Data from Text File**

```
String filePath = "D:\\APU Stuff\\Semester 5\\Java Programming (AAPP004-4-2-JP)\\Mr.  
Usman Hashmi\\Assignment\\ProductsInformation.txt";
```

```
File file = new File(filePath);
```

```
try {
```

```
    FileReader fr = new FileReader(file);
```

```
    BufferedReader br = new BufferedReader(fr);
```

```
    DefaultTableModel model = (DefaultTableModel)tableView.getModel();
```

```
    Object[] lines = br.lines().toArray();
```

```
    for(int i = 0; i < lines.length; i++){
```

```
        String[] row = lines[i].toString().split(" ");
```

```
        model.addRow(row);
```

```
    }
```

```
} catch (FileNotFoundException ex) {
```

```
    java.util.logging.Logger.getLogger(ViewProductPage.class.getName()).log(java.util.logging.  
    Level.SEVERE, null, ex);
```

```
}
```

## **3.2 Searching for a Data with a Textbox**

```
DefaultTableModel Search = (DefaultTableModel)tableView.getModel();
```

```
        TableRowSorter<DefaultTableModel> TR = new  
        TableRowSorter<DefaultTableModel>(Search);
```

```
        tableView.setRowSorter(TR);
```

```
        TR.setRowFilter(RowFilter.regexFilter(txtSearch.getText().trim()));
```

### **3.3 Restricting the Text Box with the Data Type It Is Allowed (Price)**

```
char L = evt.getKeyChar();
```

```
    if(Character.isLetter(L)&&!evt.isAltDown()||evt.isShiftDown()){
```

```
        evt.consume();
```

```
    }
```

### **3.4 Restricting the Customers from Buying Products with No Stock**

```
try {
```

```
    FileReader fr = new FileReader("D:\\APU Stuff\\Semester 5\\Java Programming  
(AAPP004-4-2-JP)\\Mr. Usman Hashmi\\Assignment\\ProductsInformation.txt");
```

```
    BufferedReader br = new BufferedReader(fr);
```

```
    String Row;
```

```
    try {
```

```
        while ((Row = br.readLine()) != null){
```

```
            String [] Data = Row.split(" ");
```

```
            String NotAvailableQuantity = "0";
```

```
            if (Data[0].equals(OrderPage.PID)){
```

```
                if (Data[3].equals(NotAvailableQuantity)){
```

```
                    btnPay.setEnabled(false);
```

```
                }
```

```
            else
```

```
            {
```

```
                btnPay.setEnabled(true);
```

```
            }
```

```
        }
```

```
    }
```

```
        br.close();
    }

    catch (IOException ex) {

        Logger.getLogger(OrderPage.class.getName()).log(Level.SEVERE, null, ex);

    }

}

catch (FileNotFoundException ex) {

    Logger.getLogger(OrderPage.class.getName()).log(Level.SEVERE, null, ex);

}
```



### **3.5 Deducting Stock from the System as Customers Buys**

```
public void UpdateStock(String editItem) {

    String FilePath = "D:\\APU Stuff\\Semester 5\\Java Programming (AAPP004-4-2-
JP)\\Mr. Usman Hashmi\\Assignment\\ProductsInformation.txt";

    String TempFile = "D:\\APU Stuff\\Semester 5\\Java Programming (AAPP004-4-2-
JP)\\Mr. Usman Hashmi\\Assignment\\ForUpdating.txt";

    File currentFile = new File(FilePath);

    File replaceFile = new File(TempFile);

    try {

        FileWriter FW = new FileWriter(TempFile, true);

        BufferedWriter BW = new BufferedWriter(FW);

        PrintWriter PW = new PrintWriter(BW);

        File file = new File(FilePath);

        FileReader FR = new FileReader(file);

        BufferedReader BR = new BufferedReader(FR);

        String eachLine;

        while ((eachLine = BR.readLine()) != null) {

            String[] Data = eachLine.split(" ");

            if (Data[1].equals(editItem)) {

                Integer Stock = Integer.parseInt(Data[3]);

                Stock = Stock -1;

                PW.println(Data[0] + " " + Data[1] + " " + Data[2] + " " + Stock);
```

```
        } else {  
            PW.println(Data[0] + " " + Data[1] + " " + Data[2] + " " + Data[3]);  
        }  
    }  
  
    BR.close();  
  
    PW.flush();  
  
    PW.close();  
  
    if(currentFile.delete()){  
        File Dump = new File(FilePath);  
  
        replaceFile.renameTo(Dump);  
  
        JOptionPane.showMessageDialog(null, "The beverage is being dispensed, please  
wait.");  
    }else{  
        JOptionPane.showMessageDialog(null, "File is not deleted and replaced.");  
    }  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "Error");  
    }  
}
```

### **3.6 Editing Table Header**

```
tableView.getTableHeader().setFont(new Font("Teko Semibold", Font.BOLD, 18));
```

## **4.0 Assumptions**

Assumptions were made as the vending machine for the program does not exist.

First, it is assumed that this vending machine is owned by a company, and that they have a set of login credentials for every machine that they own.

Then, there are times where the input is done by the machine, not the users. For example, the machine is the one that selects the “Pay” button after receiving the exact amount of payment.

Not only that, the machine is assumed to only have 10 slots for products. Therefore, only 10 product details are present and are to be edited, not adding or deleting products.