Department of Computer Science & Engineering
UNIVERSITY OF LIBERAL ARTS
BANGLADESH

University of Liberal Arts Bangladesh

Course Code: CSE 413

Course Title: Programming with JAVA Lab

Final Open Ended Lab Report

"Fruit Shop Software"

Submitted To

Dr. Mohammed Ashikur Rahman

Assistant Professor

Department of Computer Science and Engineering

University of Liberal Arts Bangladesh

Submitted By

Farjahan Akter Boby

ID:201014007

Submission Date: December 27, 2022

Table of Contents

“Fruit Shop Software”

1. Problem Statement

Implement a Java program that simulates the fruit shop where user can buy sell and display the product in GUI. Use the concept of abstraction and polymorphism.

2. Objectives

The main objective of this lab is to

1. store minimum 5 fruits in program for sell
2. Add fruit as per requirement
3. Sell fruit as per requirement
4. Display fruit status report
5. Use the graphical user interface

3. Motivation

There are several reasons behind developing, designing and analysing this project. Such that

1. To keep tracking using this software how many fruits are being adding and selling.
2. To solve the problem of data losing.
3. The user does not need to roam in the fruit store to see which fruits are available. Rather than that using this software they can easily see the list of fruits in display option which will be efficient.

4. Requirement Analysis

A. Methodology:

OOP Concept:

The whole programme could be implemented using Encapsulation, abstraction, polymorphism, Inheritance and array object.

1. Encapsulation: Encapsulation is nothing but a mechanism that binds code and function in a single unit which can not be access outside of the class. It will protect the information of the user. We can use private or protected access modifier for that. In my programme I used private access modifier as a result it will be accessed only in the current class.

As in this system, user will use the system, for protection their information needs to be protected. That's why encapsulation is essential here. The syntax for encapsulation is given below here.

```
class classs_Name{
private datatype variable;
public void setvariable(datatype variable){ this.variable=variable;}
public datatype getvariable(){ return variable;}}
```

2. Inheritance: It is a process to access one class properties to another class. Through it we can inherit object, method, attributes of a class. But we cannot inherit the constructor of a class but we can get access of a constructor by the this or super keyword. So, for code reusability this is used where a child class can inherit properties from the parent class. There are different kinds of inheritance such that

- Single-level inheritance
- Multi-level inheritance
- Hierarchical inheritance
- Multiple Level inheritance

In my project a single level inheritance has been used.

The syntax for inheritance is given below here.

```
class parent_class {
<statements>
}
public class child_class extends parent_class {
<inherit parent class properties>
}
```

In this programme, users name and ID will be inherited from the child class so that their information can be displayed or can be used.

3. Abstraction: Abstraction is the process of hiding the implementation but its functionality is visible to the users. This is used with class and methods. When a class is initialized as an abstract class that must have to be inherited at least by one regular class. Also, if contain any abstract method that must be overridden.

But this abstract method does not have any body. If all the method of that abstract class is abstract then that class must have to be inherited by another abstract class or within that abstract class, we have to create at least a regular method to solve this issue. In my lab experiment an abstract class admin was created. Within it an abstract method was deployed which was later overridden in the child class. The common syntax of abstract class is given below here

```
public abstract class a{
  public abstract void c();
  public void d(){};
}
public class b extends a{
  public void c(){
 //<statement>
}
}
```

4. Polymorphism: Polymorphism means a different form of class or method. The first condition of polymorphism is that it must fulfil the condition of inheritance first. There are 4 ways to achieve polymorphism. Such that

1. Through method overloading
2. Method overriding
3. Constructor overloading
4. Polymorphic behavior of objects

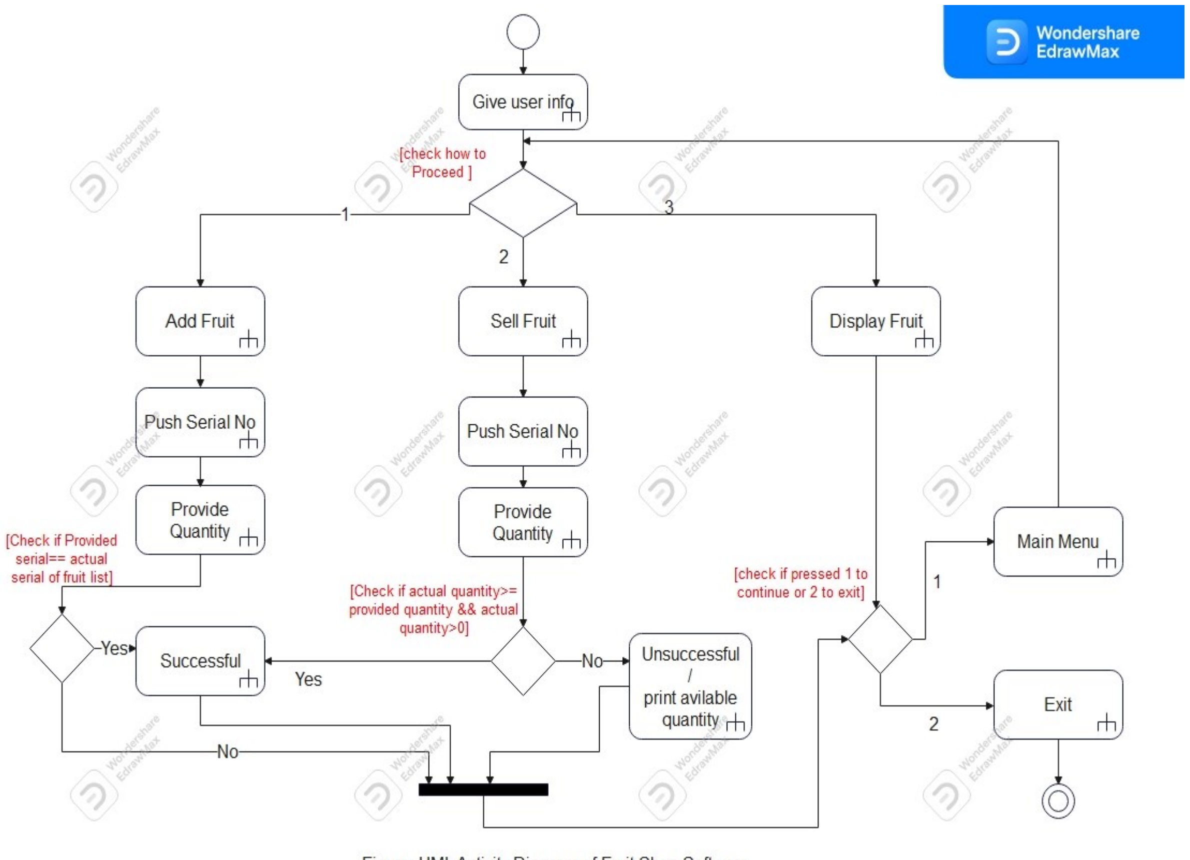
In this lab I used polymorphic behavior of objects and constructor overloading to have polymorphic behavior.

5. Array Object: An array object was created here, as multiple fruits of same type and quantity of same type needs to be pushed.

B. Library Function:

Function	Used For
append()	Append in Java is a StringBuilder and StringBuffer class method used to append a value to the current sequence. In here it is used to append serial, quantity and fruit name sequentially.
parseInt()	It is used just to convert another type of data to integer type. In this experience it is used to convert string type integer type.
equals()	This is used to check if two strings are same or not.
setText()	It receives a string from user.
getText()	It returns a string that was received from the user.
setVisible()	It is used to define if a frame will be visible or not. If it is set to true then the frame will be visible. If it is set to false it will not be visible.
JComponents	
1. JButton	The JButton class is used to create a labelled button that has platform independent implementation. In this experiment I used it for different purpose. For example, when the user press 'Enter' button, they will get access to the software. When they will press "Display" button list of available fruit will be displayed. And the 'Add' and 'Sell' button to add or sell fruits.
2. JTextField	The object of a JTextField class is a text component that allows the editing of a single line text.
3. JTextArea	The class JTextArea is a multi-line area to display plain text. In this programme, whenever user will interact in this system, all the activities like display fruits list after adding or selling or if gives any invalid information that will be shown in the textArea field.
4. JLabel	JLabel is used to display a short string or an image icon. In here it is used just to display string. It is inactive to input events such a mouse focus or keyboard focus.

C. Graphical Representation of the programme



Wondershare EdrawMax
Give user info
[check how to Proceed]
1
2
3
Add Fruit
Sell Fruit
Display Fruit
Push Serial No
Push Serial No
Provide Quantity
Provide Quantity
[Check if Provided serial== actual serial of fruit list]
[Check if actual quantity>= provided quantity && actual quantity>0]
[check if pressed 1 to continue or 2 to exit]
Yes
Successful
Yes
No
Unsuccessful / print avilable quantity
No
Main Menu
1
Exit
2

Figure: UML Activity Diagram of Fruit Shop Software

D. Explanation of the Problem:

At first to enter in the system the user will have to provide their name and code. If the code match, they will get access of the software. Then at the welcome page their information will be displayed. Along with that they will get an option to add, sell or display the fruits. Here a if_else statement was created. So, if they push 1, will be able to add fruits, if push 2 will be able to sell, else they can display the list.

```
if(choice==1)
<statement>// access to add function
else if (choice==2)
<statement>// access to sell function
else
<statement> // access to display function
```

When they push 1, they will get access to the add function, where the class object variable, fruits quantity, parent class object and the serial number will be parameterised for that function. Here they have to choose the serial number and also have to give input that how many fruits they want to add. When they will push serial number it will travers the whole array and check if that serial is available or not. If finds the serial then it will add fruits. The syntax for it is given below here

```
a.add(object_class arr[], quantity, serial, parent_class obj){
for(initial; ending of array; increment){
if(finds serial)
//increase the fruits in the list  }  }
```

When they push 2, they will get access to the sell function, where the class object variable, fruits quantity and the serial number will be parameterised to sell function. Here it will travers the whole array to find the serial. if finds then it will again check if the quantities are greater or equal to the requires fruits. If it finds that the fruits that are available is greater than the required fruits then the fruits will be sold. Otherwise, it will show that quantity and sold will not be possible. The system for it as follows

```
a.sell(object_class arr[], quantity, serial){
for(initial; ending of array; increment){
if(finds serial)
if(0<arr[i].quantity && arr[i].quantity>=required_quantity)
//decrease the quantity of fruits from the list  }  }
```

When pushed 3, it will travers the whole array using the for loop. And will print updated fruits, quantity one by one.

Again, after each step, they will be asked if wants to continue or exit the program. It will check if entered 1 or 2. If 1, then will go to the choose function to continue. Otherwise, will exit the program.

```
if(choice==1)
//terminate to the choose function
else
// exit the program
```

5. Feature of the Software

User Information

1. The user will be asked to provide his/her name and ID, to enter the system.
2. Once entered all the information, his/her information will be shown in the welcome page.

Add Fruits

1. The user will be asked of which serial of the fruit list, the user wants to increase quantities of the fruit.
2. Fruits can be added in the list as many as the user wants to add. For that, the user will be asked how many fruit, she\she wants to add.
3. If the fruits are added it will show the massage of successfully updating.

Sell Fruits

1. The user will be asked of which serial of the fruit list, the user wants to sell the fruit.
2. Again, fruits can be sold from the list as many as the user wants to sell. For that, the user will be asked how many fruit, she\she wants to sell.
3. If the fruits are sold it will show the massage of successfully selling.
4. If there are not available quantities to sell, it will show how many quantities of that fruit is available or can sell.

Display Fruits

1. In this interface the user can see the fruit list to add or sell.
2. Also, after adding or selling the fruits, the user can display the updated fruit list.

The user can add fruits, sell fruits or display the fruit list in this system as many times as he/she wants. Also, can exit at any time.

6. Result and Analysis:

Fruit_detail class:

This is an object class. which has two attributes called quantity and name. Also have a parameterised constructor. The attributes of this constructor have been accessed in the fruit_shop class. Again, this constructor has been overloaded here which has no parameters. Polymorphism has been done here by the constructor overloading.

Admin class:

This is an abstract class which later has been inherited by the fruit_shop class.

Attributes: It has two attributes such that name and ID. Through encapsulation name and ID of the user has been protected here.

Methods: The methods I used within this class is given below here

setID(): It receives the code of the user from the user.

getID(): It returns the code of the user from the setID method. The return type of it is integer.

setname1(): It receives the name of the user from the user.

getname1(): It returns the name of the user from the setname1 method. The return type of it is string.

design(): This is an abstract method. It has been overridden within the fruit_shop class.

Fruit_Shop class:
This is a regular class which inherited the abstract class attributes and methods.

Methods: The methods I used within this class is given below here

design(): This method was overridden from the abstract class cause this was abstract method of the Admin class.

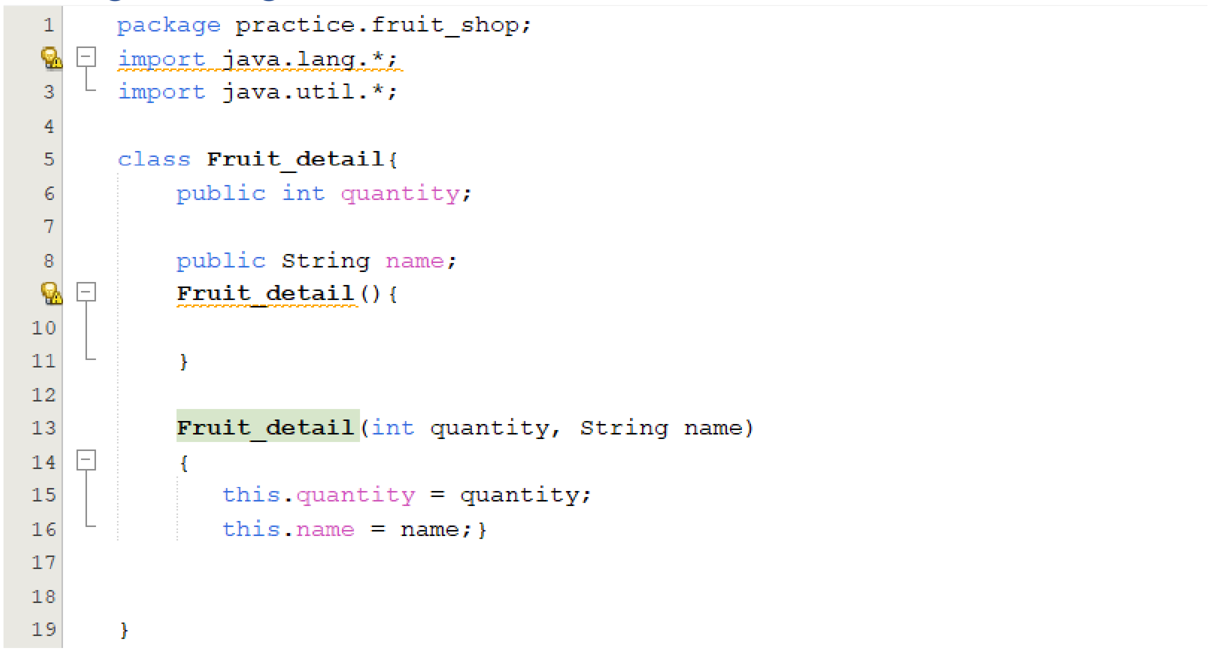
choice(): This method was parameterised with the 'arr[]' objects of fruit_details and object 'a' of fruit_shop. Through this method user can choose how they want to proceed like want to add fruit or delete fruit or display fruit. If they choose 1 then it will ask for quantity and serial. Then will pass these variable to the add_fruit() method. If they choose 2 then it will ask for quantity and serial. Then will pass these variable to the sell_fruit() method. If they choose 3 then refer the display_fruit() method.

add_fruit(): This method was parameterised with quantity, serial number, 'arr[]' objects of fruit_details and object 'a' of fruit_shop. Then within the for-loop check whether the serial number exists here or not. If exists then fruits will be added.

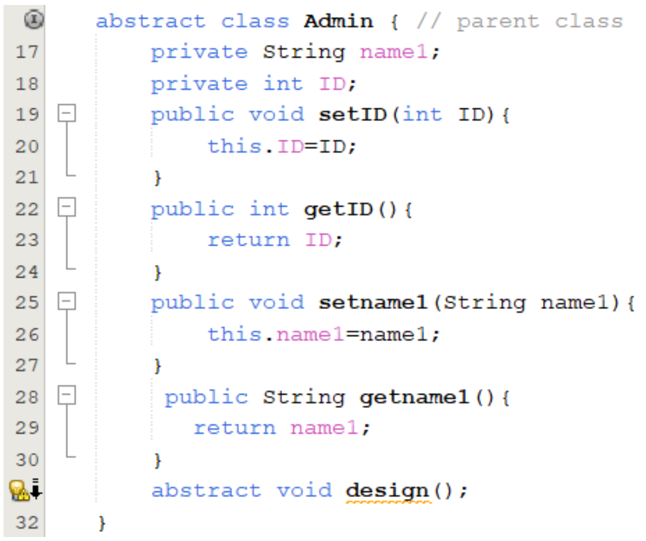
sell_fruit(): This method was parameterised with quantity, serial number, 'arr[]' objects of fruit_details and object 'a' of fruit_shop. Then within the for-loop check whether the serial number exists here or not. If exists then fruits will be deleted. At the same time, it will check whether the quantity is less than the fruits have. if it is then selling will be successful.

display_fruit(): This method was parameterised with the ‘arr[]’ objects of fruit_details and object ‘a’ of fruit_shop. In here it will simply display the list of available or updated fruits.

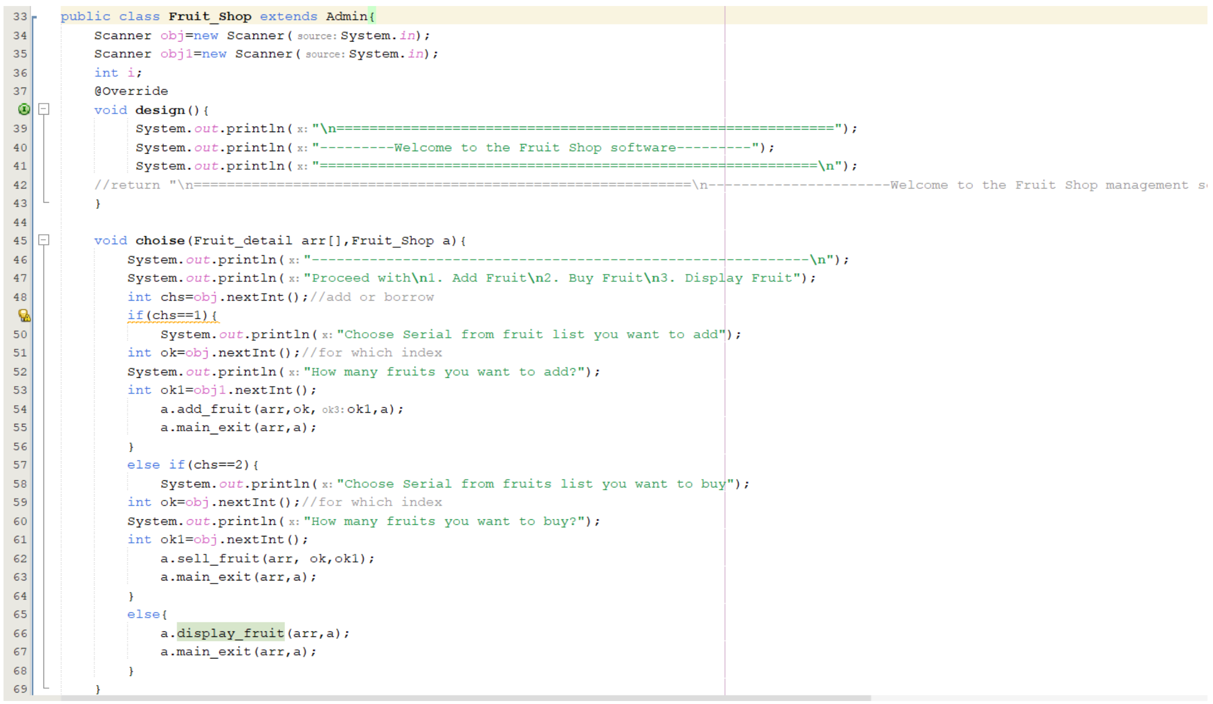
7. Programming Code:



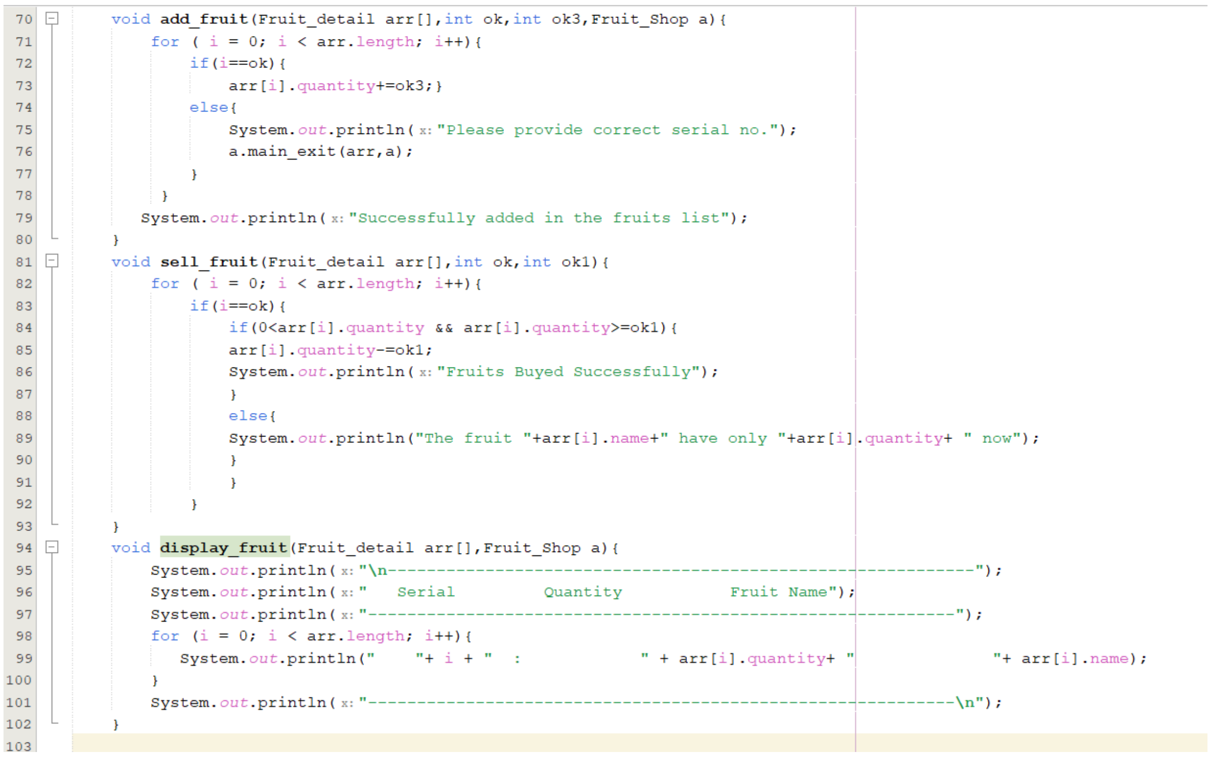
```java
package practice.fruit_shop;
import java.lang.*;
import java.util.*;

class Fruit_detail{
    public int quantity;

    public String name;
    Fruit_detail(){

    }

    Fruit_detail(int quantity, String name)
    {
       this.quantity = quantity;
       this.name = name;}


}
```



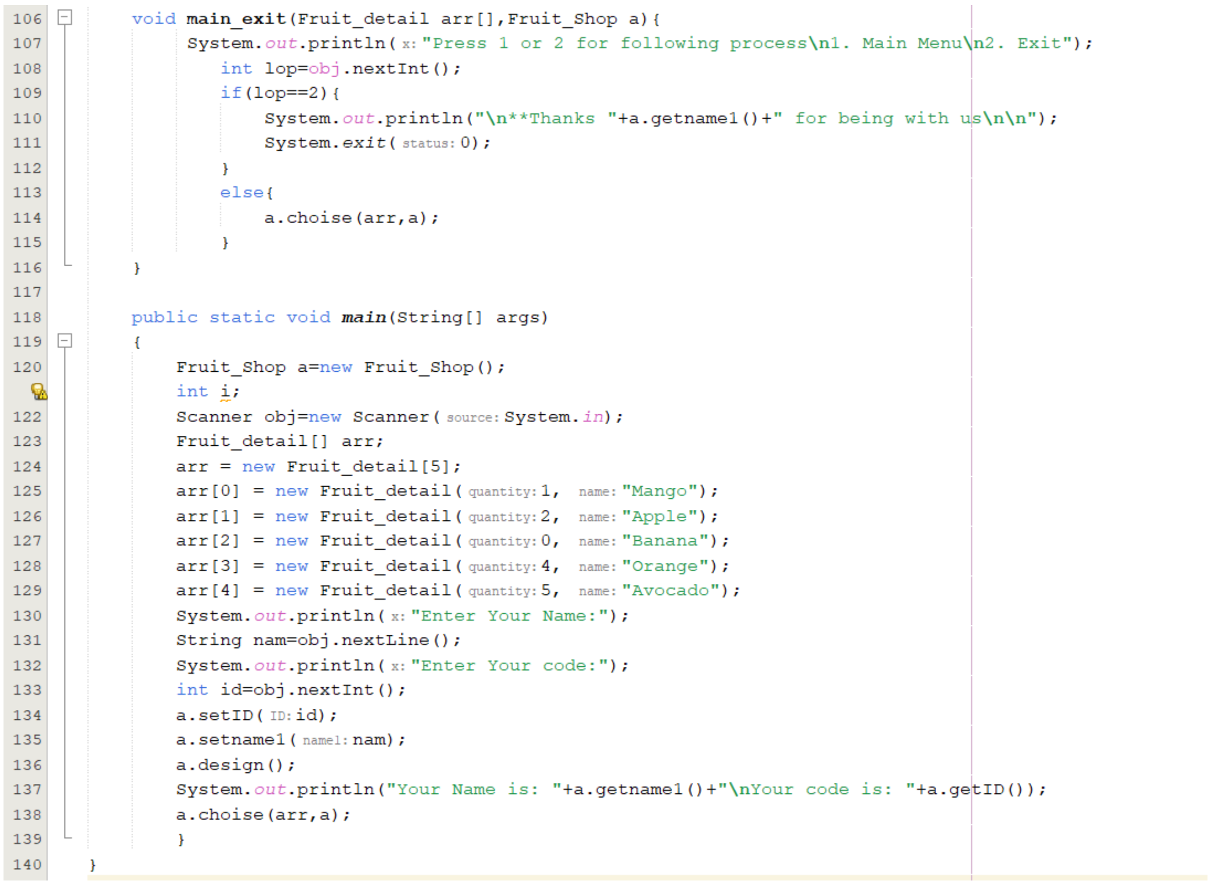
```java
abstract class Admin { // parent class
    private String name1;
    private int ID;
    public void setID(int ID){
        this.ID=ID;
    }
    public int getID(){
        return ID;
    }
    public void setname1(String name1){
        this.name1=name1;
    }
     public String getname1(){
       return name1;
    }
    abstract void design();
}
```



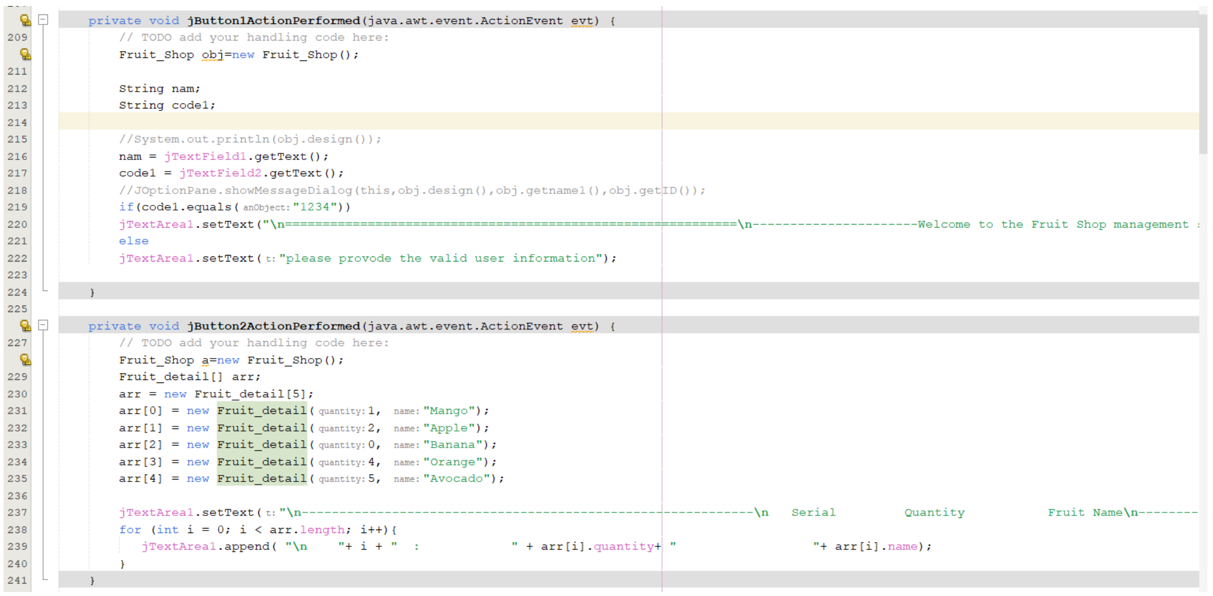
```java
public class Fruit_Shop extends Admin{
    Scanner obj=new Scanner( source: System.in);
    Scanner obj1=new Scanner( source: System.in);
    int i;
    @Override
    void design(){
         System.out.println( x: "\n==============================================================");
         System.out.println( x: "---------Welcome to the Fruit Shop software---------");
         System.out.println( x: "==============================================================\n");
    //return "\n==============================================================\n---------------------Welcome to the Fruit Shop management s
    }

    void choise(Fruit_detail arr[],Fruit_Shop a){
        System.out.println( x: "-------------------------------------------------------------\n");
        System.out.println( x: "Proceed with\n1. Add Fruit\n2. Buy Fruit\n3. Display Fruit");
        int chs=obj.nextInt();//add or borrow
        if(chs==1){
            System.out.println( x: "Choose Serial from fruit list you want to add");
        int ok=obj.nextInt();//for which index
        System.out.println( x: "How many fruits you want to add?");
        int ok1=obj1.nextInt();
            a.add_fruit(arr,ok, ok3: ok1,a);
            a.main_exit(arr,a);
        }
        else if(chs==2){
            System.out.println( x: "Choose Serial from fruits list you want to buy");
        int ok=obj.nextInt();//for which index
        System.out.println( x: "How many fruits you want to buy?");
        int ok1=obj.nextInt();
            a.sell_fruit(arr, ok,ok1);
            a.main_exit(arr,a);
        }
        else{
            a.display_fruit(arr,a);
            a.main_exit(arr,a);
        }
    }
```



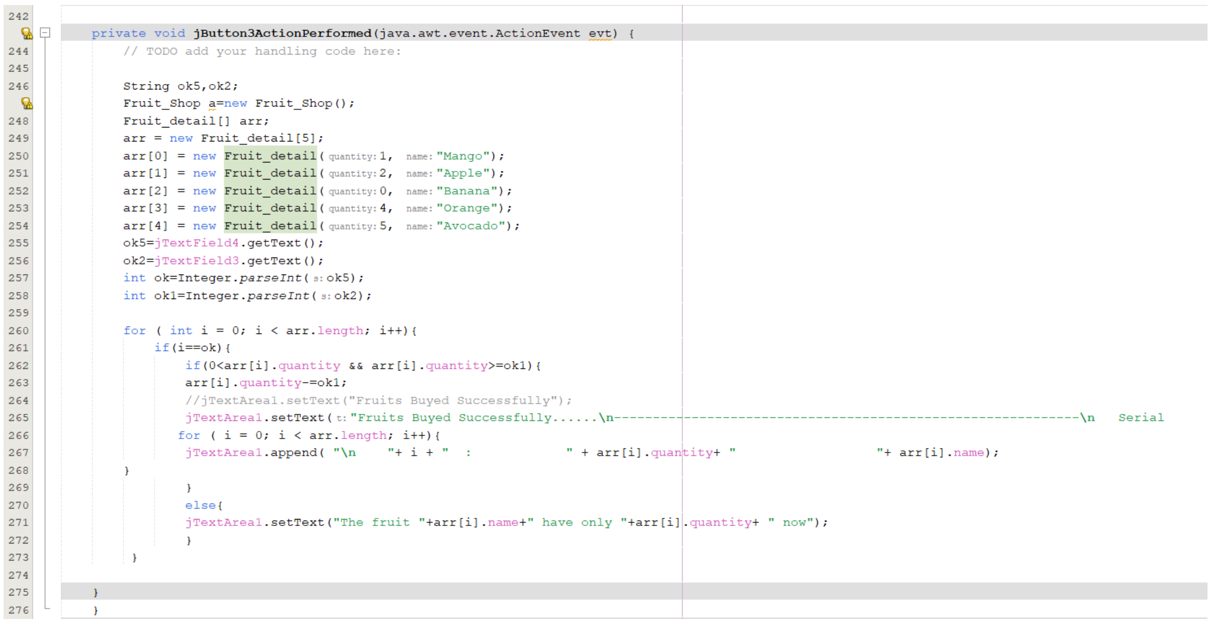
```java
    void add_fruit(Fruit_detail arr[],int ok,int ok3,Fruit_Shop a){
        for ( i = 0; i < arr.length; i++){
            if(i==ok){
                arr[i].quantity+=ok3;}
            else{
                System.out.println( x: "Please provide correct serial no.");
                a.main_exit(arr,a);
            }
        }
      System.out.println( x: "Successfully added in the fruits list");
    }
    void sell_fruit(Fruit_detail arr[],int ok,int ok1){
        for ( i = 0; i < arr.length; i++){
            if(i==ok){
                if(0<arr[i].quantity && arr[i].quantity>=ok1){
                arr[i].quantity-=ok1;
                System.out.println( x: "Fruits Buyed Successfully");
                }
                else{
                System.out.println("The fruit "+arr[i].name+" have only "+arr[i].quantity+ " now");
                }
            }
    }
    void display_fruit(Fruit_detail arr[],Fruit_Shop a){
        System.out.println( x: "\n---------------------------------------------------------------");
        System.out.println( x: "  Serial         Quantity             Fruit Name");
        System.out.println( x: "---------------------------------------------------------------");
        for (i = 0; i < arr.length; i++){
            System.out.println("    "+ i + "  :              " + arr[i].quantity+ "                 "+ arr[i].name);
        }
        System.out.println( x: "------------------------------------------------------------\n");
    }
```



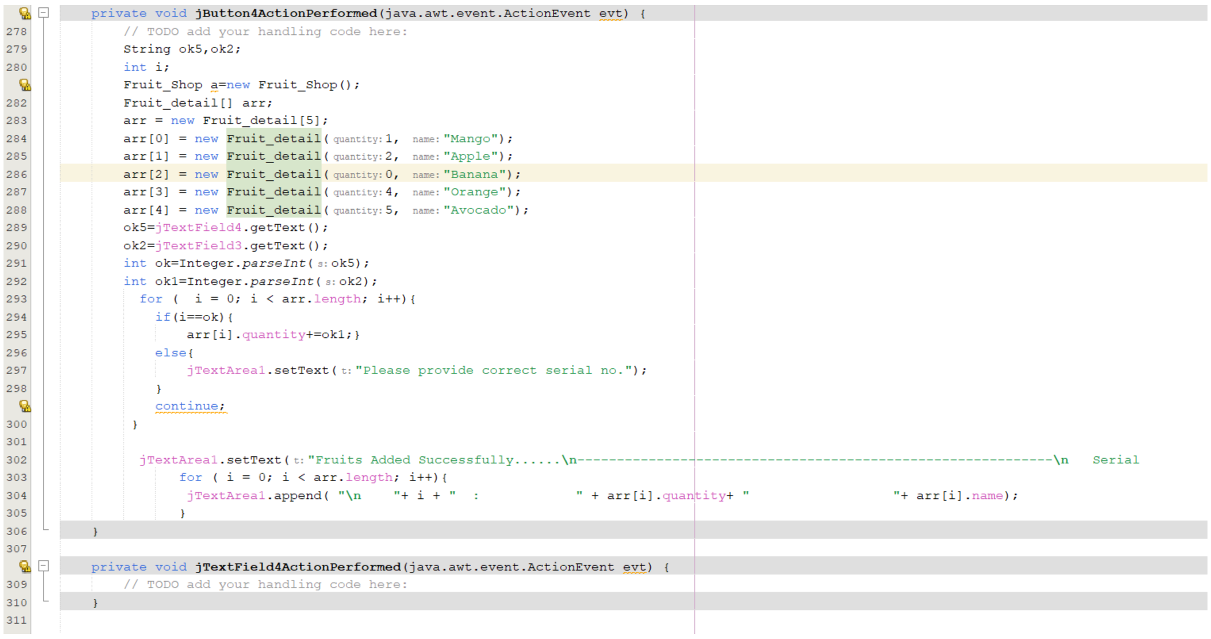
```java
    void main_exit(Fruit_detail arr[],Fruit_Shop a){
        System.out.println( x: "Press 1 or 2 for following process\n1. Main Menu\n2. Exit");
            int lop=obj.nextInt();
            if(lop==2){
                System.out.println("\n**Thanks "+a.getname1()+" for being with us\n\n");
                System.exit( status: 0);
            }
            else{
                a.choise(arr,a);
            }
    }

    public static void main(String[] args)
    {
        Fruit_Shop a=new Fruit_Shop();
        int i;
        Scanner obj=new Scanner( source: System.in);
        Fruit_detail[] arr;
        arr = new Fruit_detail[5];
        arr[0] = new Fruit_detail( quantity: 1,  name: "Mango");
        arr[1] = new Fruit_detail( quantity: 2,  name: "Apple");
        arr[2] = new Fruit_detail( quantity: 0,  name: "Banana");
        arr[3] = new Fruit_detail( quantity: 4,  name: "Orange");
        arr[4] = new Fruit_detail( quantity: 5,  name: "Avocado");
        System.out.println( x: "Enter Your Name:");
        String nam=obj.nextLine();
        System.out.println( x: "Enter Your code:");
        int id=obj.nextInt();
        a.setID( ID: id);
        a.setname1( name1: nam);
        a.design();
        System.out.println("Your Name is: "+a.getname1()+"\nYour code is: "+a.getID());
        a.choise(arr,a);
        }
}
```



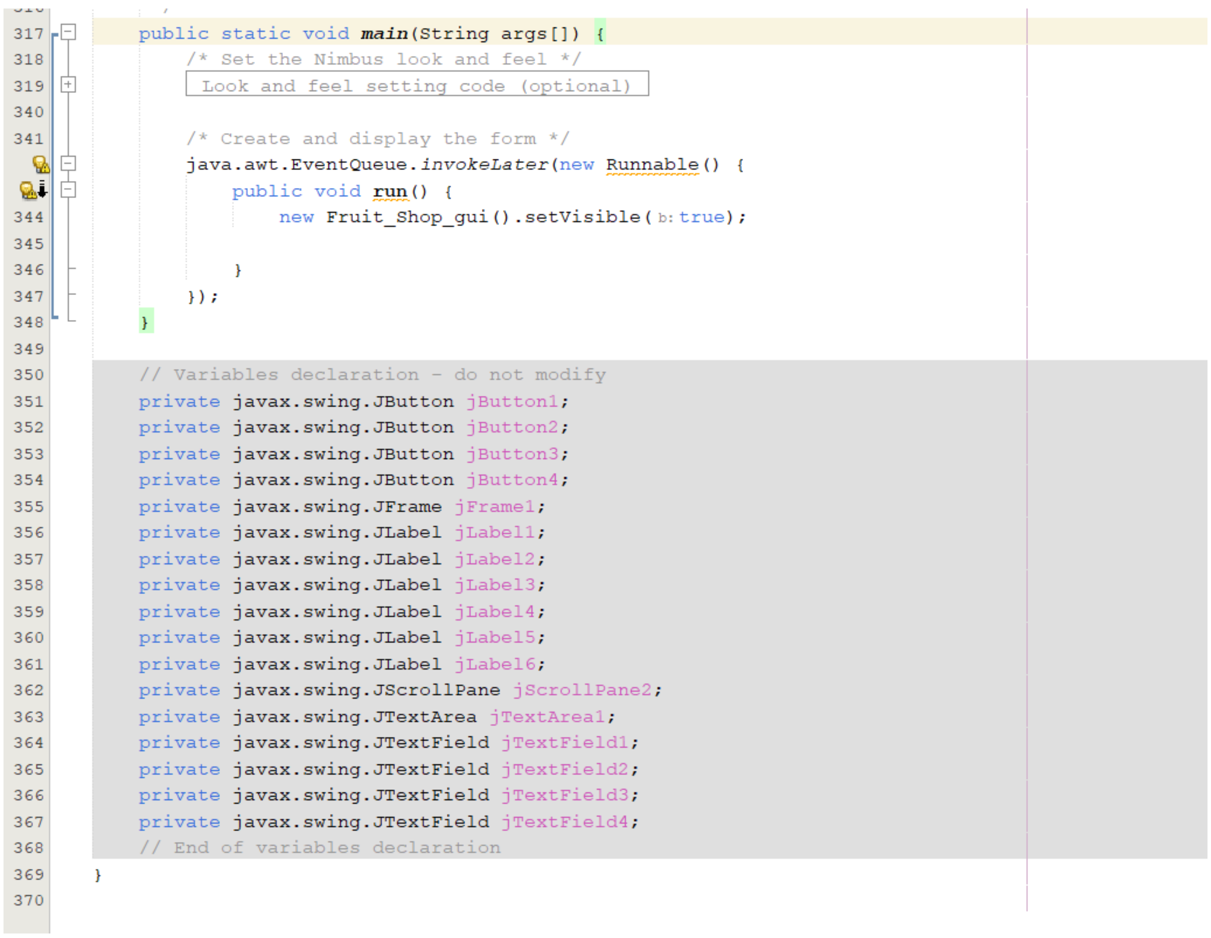
```java
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Fruit_Shop obj=new Fruit_Shop();

        String nam;
        String code1;

        //System.out.println(obj.design());
        nam = jTextField1.getText();
        code1 = jTextField2.getText();
        //JOptionPane.showMessageDialog(this,obj.design(),obj.getname1(),obj.getID());
        if(code1.equals( anObject: "1234"))
        jTextArea1.setText("\n============================================================\n----------------------Welcome to the Fruit Shop management :
        else
        jTextArea1.setText( t: "please provode the valid user information");

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Fruit_Shop a=new Fruit_Shop();
        Fruit_detail[] arr;
        arr = new Fruit_detail[5];
        arr[0] = new Fruit_detail( quantity: 1,  name: "Mango");
        arr[1] = new Fruit_detail( quantity: 2,  name: "Apple");
        arr[2] = new Fruit_detail( quantity: 0,  name: "Banana");
        arr[3] = new Fruit_detail( quantity: 4,  name: "Orange");
        arr[4] = new Fruit_detail( quantity: 5,  name: "Avocado");

        jTextArea1.setText( t: "\n------------------------------------------------------------\n   Serial         Quantity            Fruit Name\n--------
        for (int i = 0; i < arr.length; i++){
           jTextArea1.append( "\n    "+ i + "  :            " + arr[i].quantity+ "                  "+ arr[i].name);
        }
    }
```



```java

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String ok5,ok2;
        Fruit_Shop a=new Fruit_Shop();
        Fruit_detail[] arr;
        arr = new Fruit_detail[5];
        arr[0] = new Fruit_detail( quantity: 1,  name: "Mango");
        arr[1] = new Fruit_detail( quantity: 2,  name: "Apple");
        arr[2] = new Fruit_detail( quantity: 0,  name: "Banana");
        arr[3] = new Fruit_detail( quantity: 4,  name: "Orange");
        arr[4] = new Fruit_detail( quantity: 5,  name: "Avocado");
        ok5=jTextField4.getText();
        ok2=jTextField3.getText();
        int ok=Integer.parseInt( s: ok5);
        int ok1=Integer.parseInt( s: ok2);

        for ( int i = 0; i < arr.length; i++){
            if(i==ok){
                if(0<arr[i].quantity && arr[i].quantity>=ok1){
                arr[i].quantity-=ok1;
                //jTextArea1.setText("Fruits Buyed Successfully");
                jTextArea1.setText( t: "Fruits Buyed Successfully......\n--------------------------------------------------------------\n   Serial
               for ( i = 0; i < arr.length; i++){
                jTextArea1.append( "\n    "+ i + "  :              " + arr[i].quantity+ "                  "+ arr[i].name);
        }

                }
                else{
                jTextArea1.setText("The fruit "+arr[i].name+" have only "+arr[i].quantity+ " now");
                }
         }

    }
    }
```



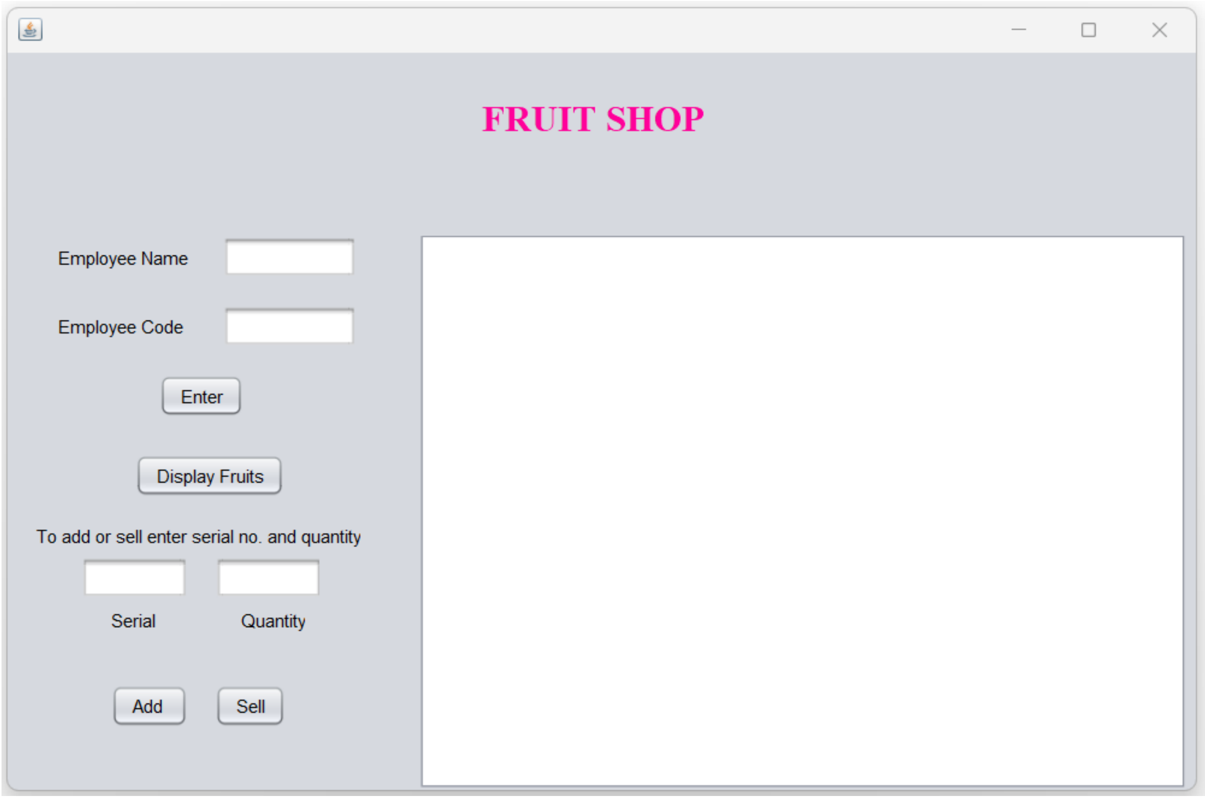
```java
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String ok5,ok2;
        int i;
        Fruit_Shop a=new Fruit_Shop();
        Fruit_detail[] arr;
        arr = new Fruit_detail[5];
        arr[0] = new Fruit_detail( quantity: 1,  name: "Mango");
        arr[1] = new Fruit_detail( quantity: 2,  name: "Apple");
        arr[2] = new Fruit_detail( quantity: 0,  name: "Banana");
        arr[3] = new Fruit_detail( quantity: 4,  name: "Orange");
        arr[4] = new Fruit_detail( quantity: 5,  name: "Avocado");
        ok5=jTextField4.getText();
        ok2=jTextField3.getText();
        int ok=Integer.parseInt( s: ok5);
        int ok1=Integer.parseInt( s: ok2);
          for (  i = 0; i < arr.length; i++){
            if(i==ok){
                arr[i].quantity+=ok1;}
            else{
                jTextArea1.setText( t: "Please provide correct serial no.");
            }
            continue;
         }

          jTextArea1.setText( t: "Fruits Added Successfully......\n--------------------------------------------------------------\n   Serial
               for ( i = 0; i < arr.length; i++){
                jTextArea1.append( "\n    "+ i + "  :              " + arr[i].quantity+ "                  "+ arr[i].name);
                }
    }

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

```



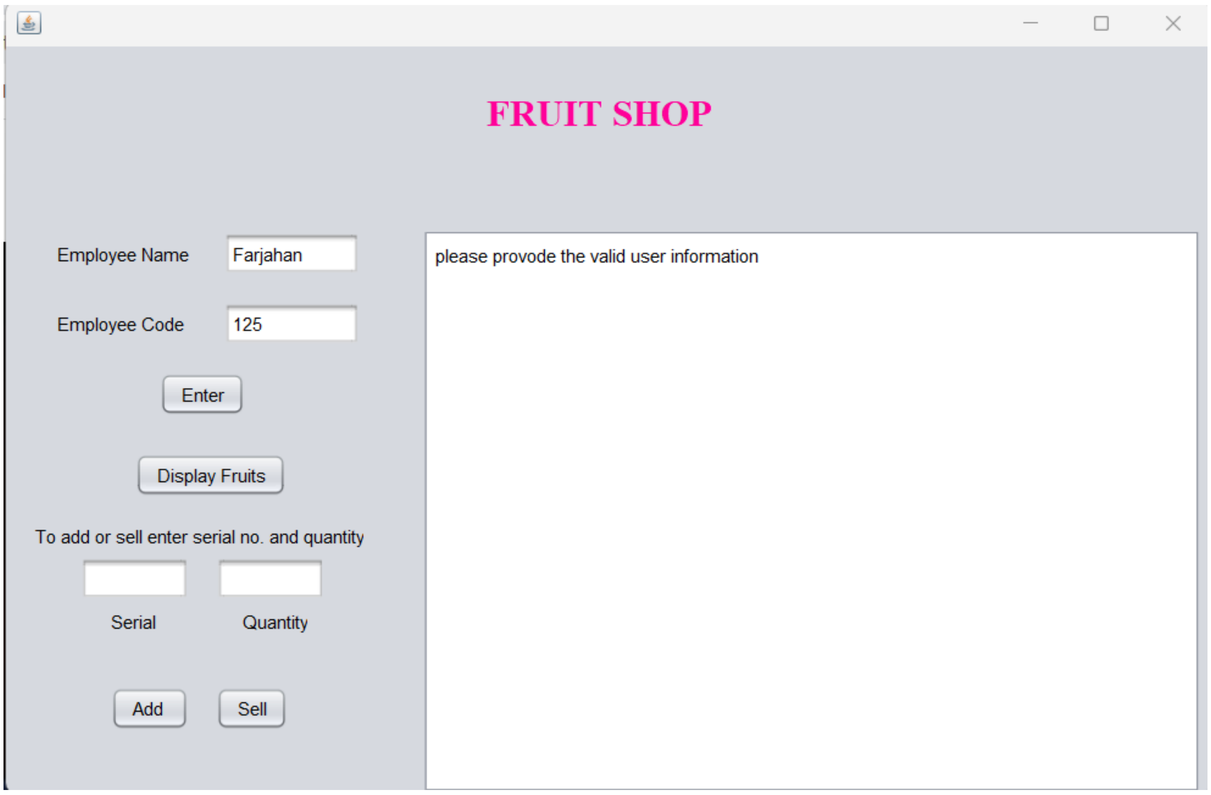
```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

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

            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JFrame jFrame1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    // End of variables declaration
}
```

8. Output:

Welcome Interface



To use this software, at first when the user will provide her information the screen will be as following. If the user does not provide correct information than it will show valid user information needs to provide like the following interface.

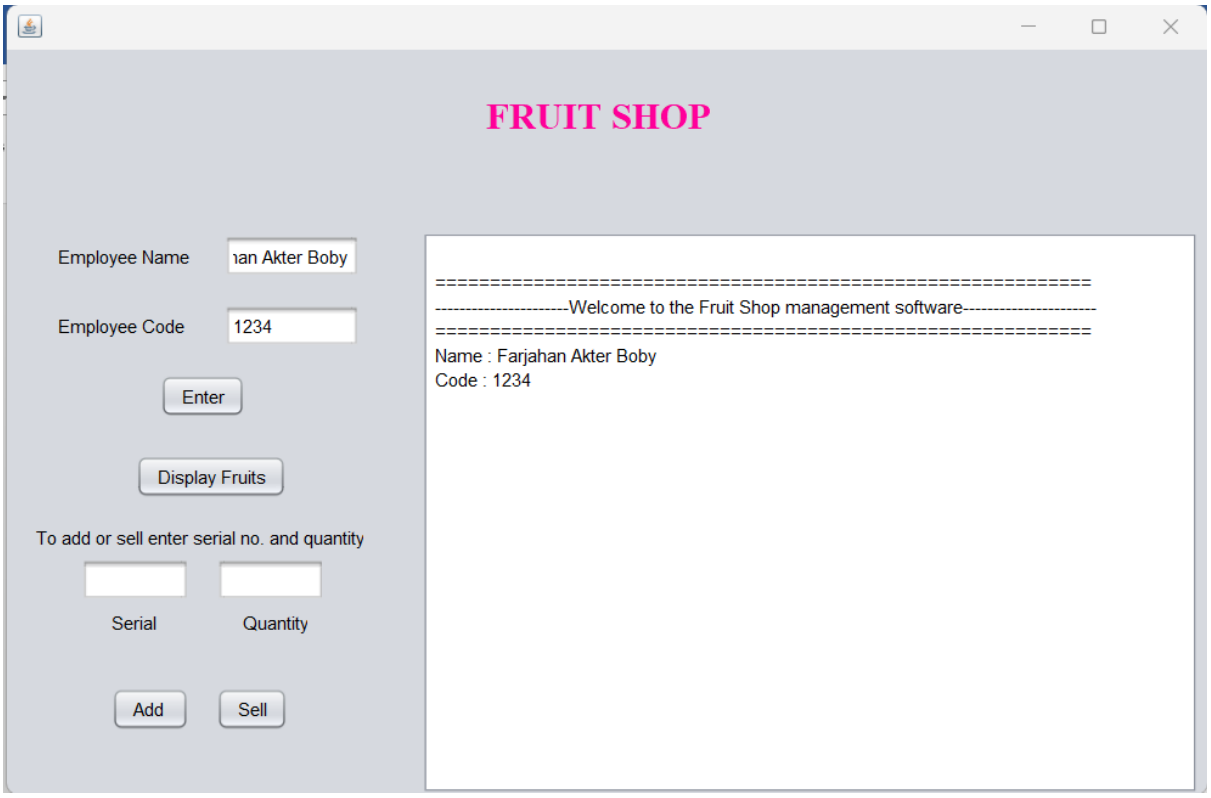

FRUIT SHOP

Employee Name: Farjahan

Employee Code: 125

please provode the valid user information

When they provided correct code, they were able to enter the interface and in the welcome interface the information was showed.

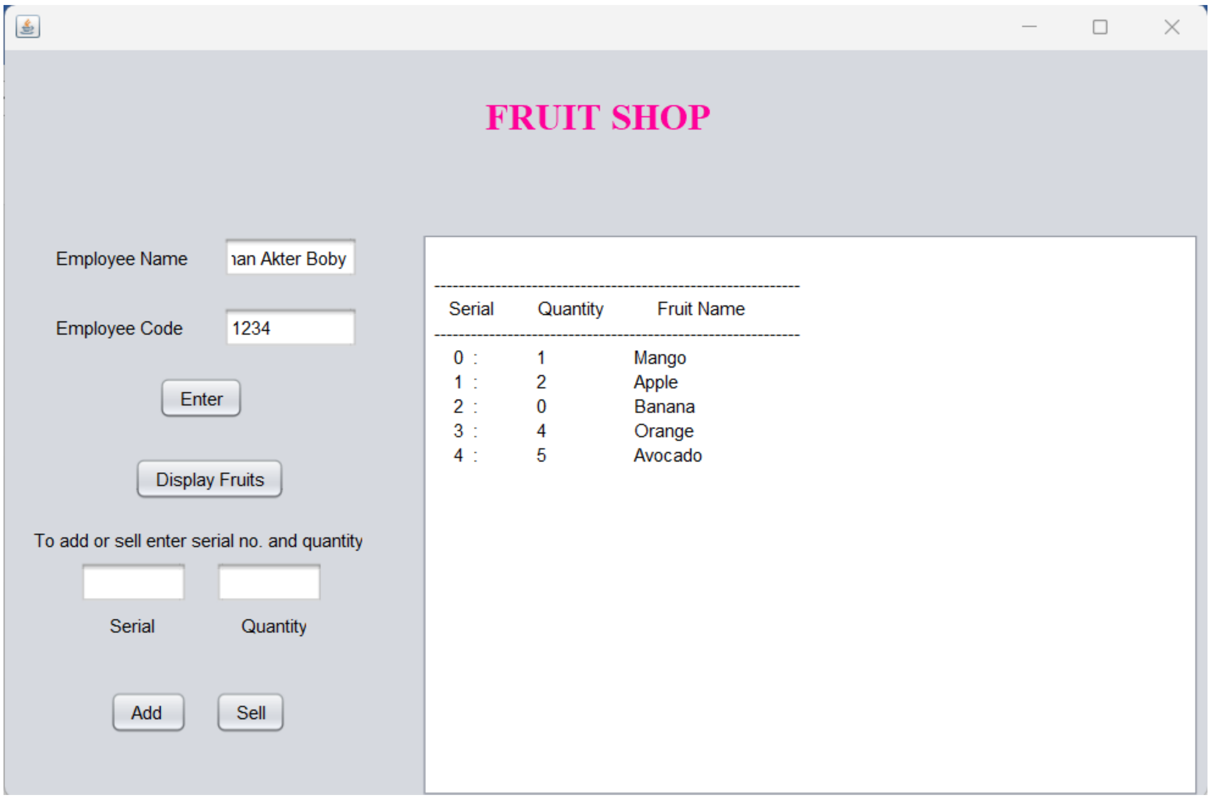


Employee Name: ıan Akter Boby
Employee Code: 1234

```
==============================================================
---------------------Welcome to the Fruit Shop management software---------------------
==============================================================
Name : Farjahan Akter Boby
Code : 1234
```

Display Interface

To see the list of fruit list when the user will press display button, the list will be displayed in the jTextArea like the following interface.

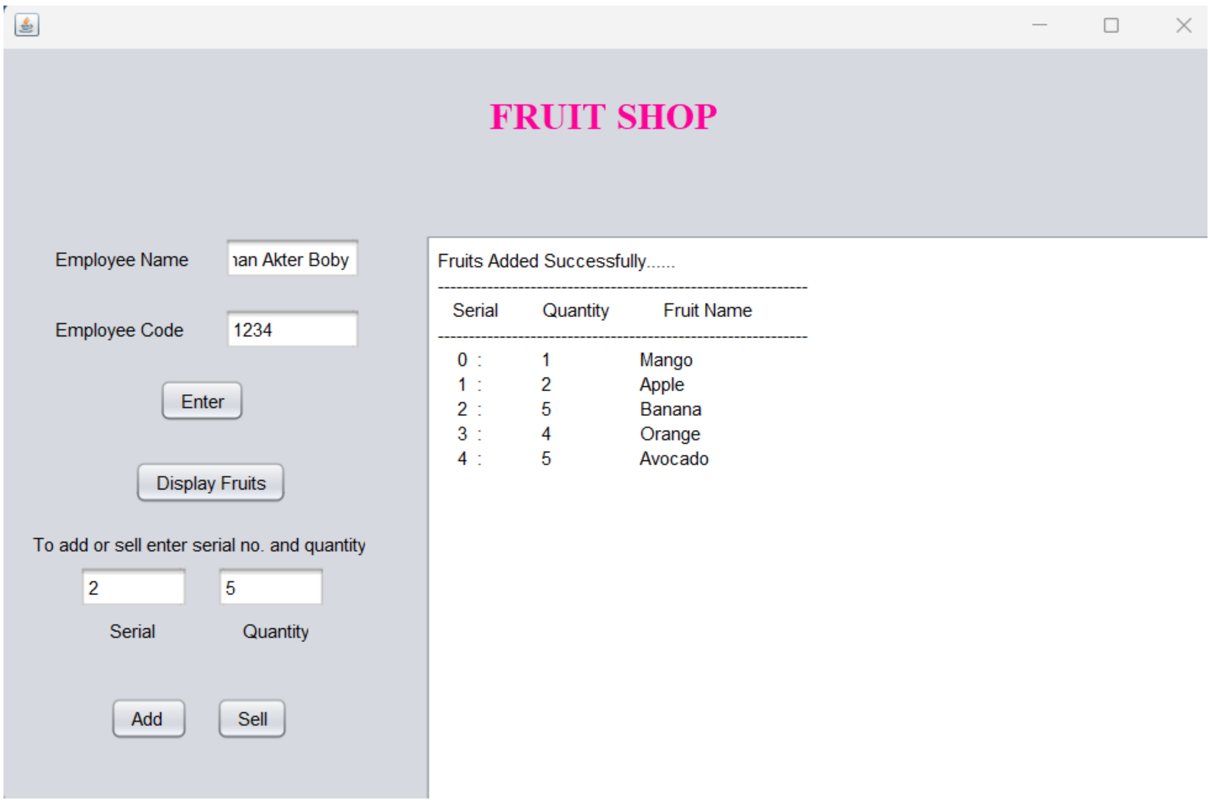


Employee Name: ıan Akter Boby
Employee Code: 1234

```
-----------------------------------------------------------
  Serial      Quantity        Fruit Name
-----------------------------------------------------------
   0 :          1            Mango
   1 :          2            Apple
   2 :          0            Banana
   3 :          4            Orange
   4 :          5            Avocado
```

Adding Fruit Interface

To add fruits at first when the user pushed serial number and the quantity and presses Add button, it showed that in serial 2 which means 5 Banana has been added. Though at the initial stage there was 0 Banana.

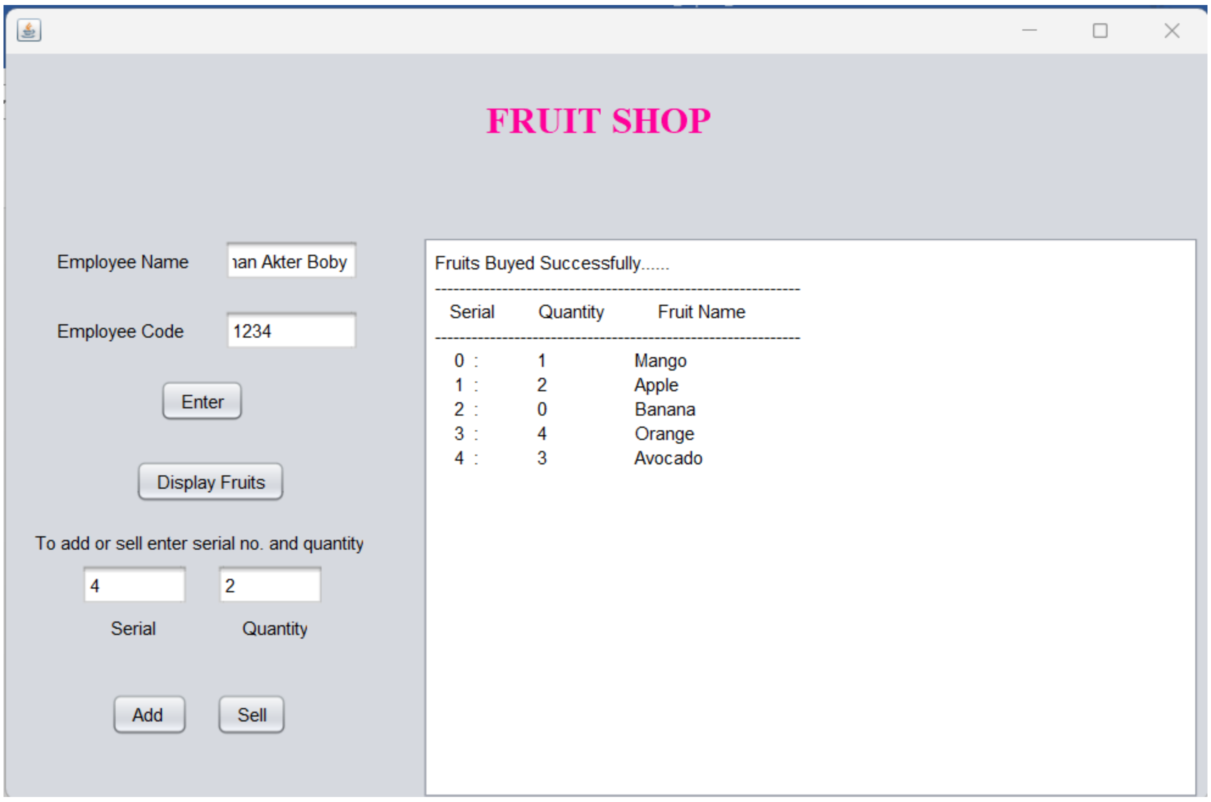


Employee Name	ıan Akter Boby
Employee Code	1234
Serial	2
Quantity	5

```
Fruits Added Successfully......
--------------------------------------------------------------
   Serial      Quantity        Fruit Name
--------------------------------------------------------------
   0 :          1              Mango
   1 :          2              Apple
   2 :          5              Banana
   3 :          4              Orange
   4 :          5              Avocado
```

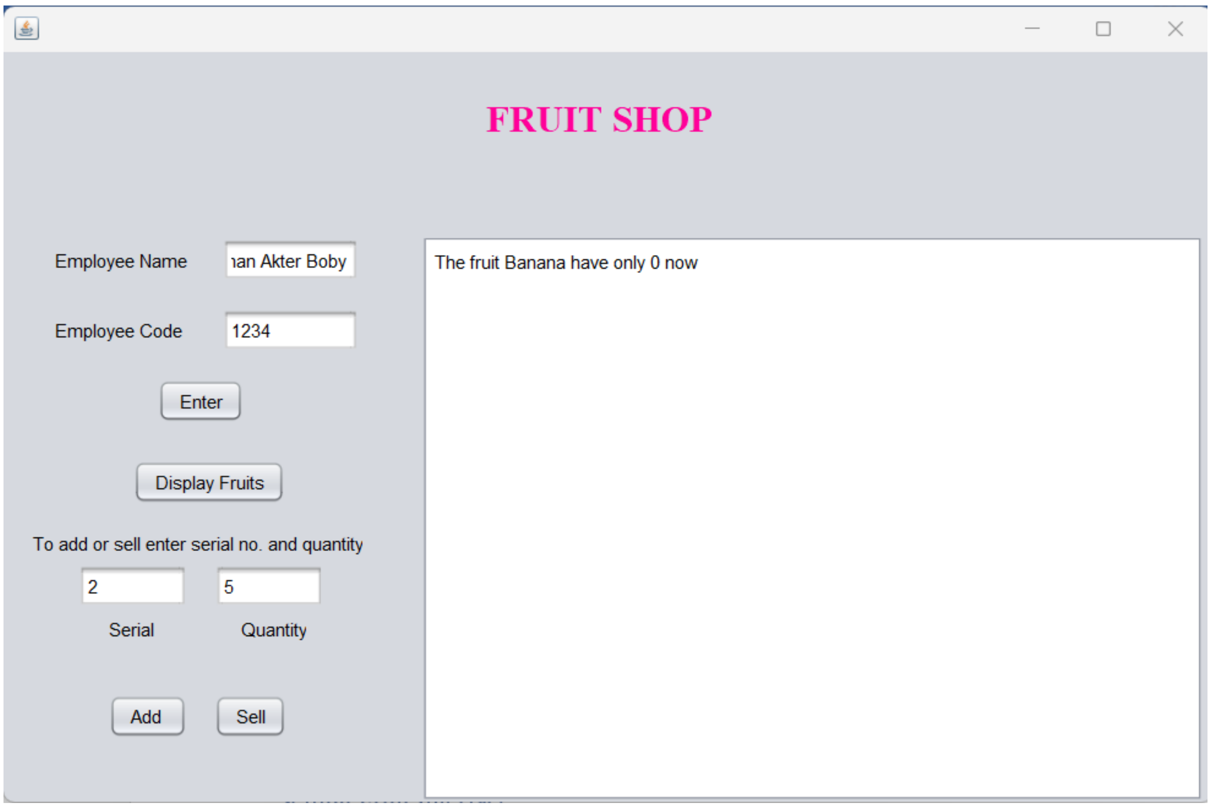
Selling Fruit Interface

Again, to sell fruits when user pushed serial 4, which means Avocado and pushed quantity as 2, it was sold successfully. And the updated list is the following figure. There at first was only 5 Avocado. But after selling it have only 3 Avocado.



```
Fruits Buyed Successfully......
------------------------------------------------------------
  Serial      Quantity       Fruit Name
------------------------------------------------------------
   0  :          1            Mango
   1  :          2            Apple
   2  :          0            Banana
   3  :          4            Orange
   4  :          3            Avocado
```

But when the acquired quantity is greater than the fruits that are available in the list, selling fruits will be unsuccessful. For example, in the list there was no Banana in serial 2, but user wanted to sell 5 bananas, so it showed the quantity in the display like the following interface.



```
The fruit Banana have only 0 now
```

9. Conclusion

Using Apache NetBeans, the whole code was built following the problem statements and problem theory. Some properties were inherited, some were kept protected. Thus, the code was reusable.

Again, outcome section it is clearly predicted that, the user can add or sell fruits whenever they want to do it while using the software. Also, can see the updated list at any time. The whole programme was successfully built using the Abstraction, polymorphism, inheritance and encapsulation.