Objective:

The main goals of the provided code are to create a system for managing electronic devices in a store. This involves handling device information like name, price, date, quantity, brand, and model.

Problem:

The challenge to address is how to effectively manage the information related to electronic devices in a store. This includes keeping track of device details, prices, quantities, and other attributes.

Prerequisites:

Before diving into the code, it's helpful to have a basic understanding of Java programming concepts, such as classes, objects, constructors, and methods. Familiarity with graphical user interfaces (GUIs) and event handling in Java Swing can also be beneficial.

Learning Outcomes:

By working with this code, you can expect to gain insights into how to create a basic application for handling electronic device information. You'll learn about constructing and managing objects, using graphical components for user interaction, and implementing functionality to edit and delete device entries.

Problem Analysis:

The code tackles the issue of efficiently managing electronic device information within a store context. It addresses how to handle tasks like adding, editing, and deleting device records while keeping the interface user-friendly.

Background Theory:

The code leverages Java programming concepts like classes and objects. It also employs Java Swing for creating the graphical user interface (GUI) components, such as buttons, text fields, and tables. Additionally, it showcases event handling to respond to user interactions effectively.

Algorithm Design:

The algorithm design involves creating a Java class named retail to represent individual electronic devices. This class stores information like device name, price, date, quantity, brand, and model. It includes a constructor for initializing device attributes and a method for displaying these attributes. The code also features functions to handle adding, editing, and deleting device records within a graphical interface. This requires selecting rows, updating values, and managing the underlying data model.

Code:

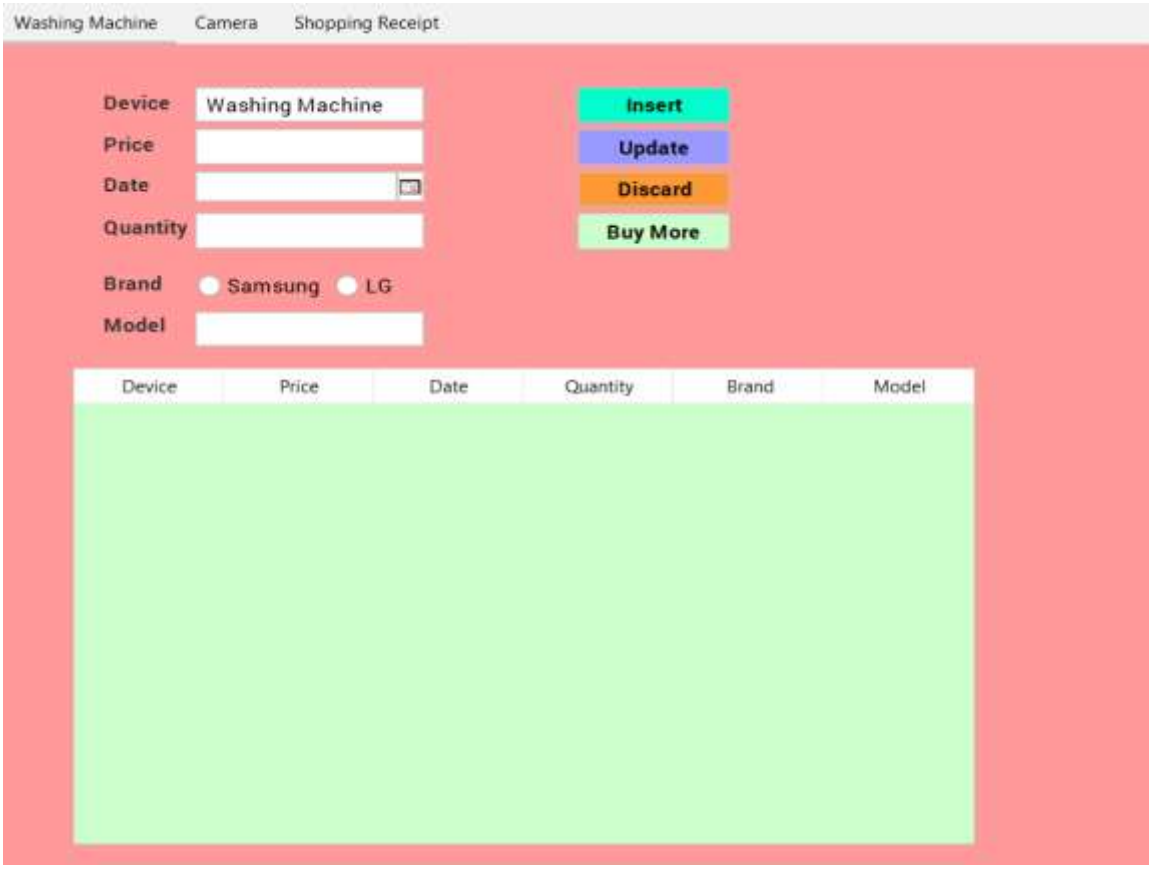
```
package Electronics;

public class retail {
    String device;
    int price;
    String date;
    int quantity;
    String brand;
    String model;
    String radioValue;
    retail(String device,int price,String date, int quantity, String brand,String model)
    {
        this.device=device;
        this.price=price;
        this.date=date;
        this.quantity = quantity;
        this.brand=brand;
        this.model=model;

    }
   void display() {
        System.out.println(device + " " + price + " " + date + " " + quantity+" "+brand+" "+ model);
    }
}
```

This code defines a Java class named "retail" in the "Electronics" package, representing electronic devices with attributes like device name, price, date, quantity, brand, and model. It has a constructor to initialize these attributes and a method to display the device's information.

```
package Electronics;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class Store extends javax.swing.JFrame {

    ArrayList<retail> lb;
DefaultTableModel dtm;
    public Store() {
        lb=new ArrayList<>();
         initComponents();
    }
```

This code defines a graphical interface (GUI) for an electronics store using Java Swing, displaying a table to manage retail items, and allows interaction with the store's inventory.

Washing Machine Panel

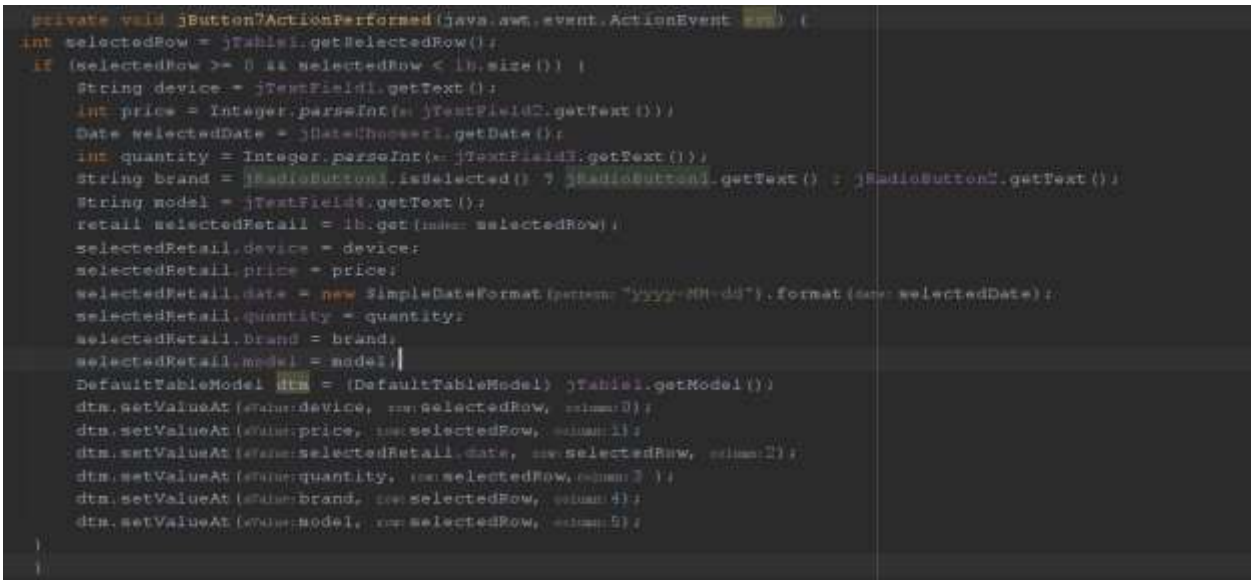


I used JFrame to create a user interface for a Washing Machine store. It enables easy data insertion, jTable data updates, data deletion, and includes a "Buy More" option to transition to another panel for purchasing additional devices.

Insert Button

When the button is clicked, this code adds a new retail item to a list based on user inputs, updates two jTable views with the item details, and clears input fields and selections for adding more items. It uses swing components and manages data for devices, prices, dates, quantities, brands, and models.

Update Button



When jButton7 button is clicked, this code updates the selected row in a table. It takes input for retails details like device, price, date, quantity, brand, and model, then modifies the corresponding row values with these inputs in the jTable being displayed.

Discard Button

It checks if the user wants to delete a row from a jTable1, and if confirmed, it removes the selected row's data from a list lb and updates the table's display accordingly.

Buy More Button

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
jTabbedPane1.setSelectedIndex(index: 1);
}
```

When the jButton5 is clicked, this code switches the tabbed view to the second panel.

After Click in Buy More Button

Camera Panel

After clicking the "Buy More" button (jButton5), a new panel named "Camera" opens. This panel contains buttons for inserting, deleting, and discarding items, similar to the washing machine panel. Additionally, the Camera panel has two distinct buttons: "Back" to return to the previous Washing Machine panel and "Buy Now" to proceed to the Shopping Receipt panel for the final receipt.

The "Insert," "Delete," and "Discard" buttons function similarly to the Washing Machine panel. However, the "Back" button returns to the previous panel, and the "Buy Now" button leads to a new panel named "Shopping Receipt" for the final receipt.

Back Button

```
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
jTabbedPane1.setSelectedIndex(index: 0);
}
```

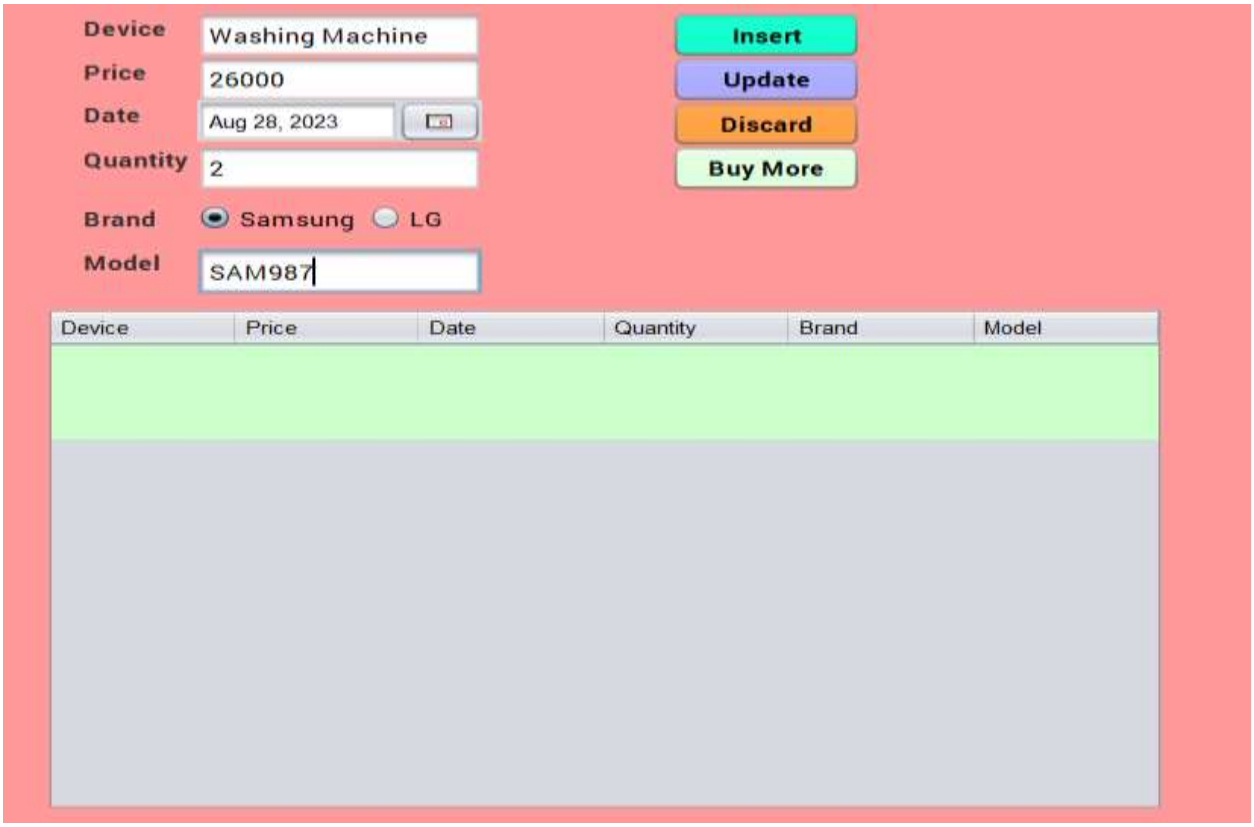
This code switches the Camena panel to the first Washing Machine panel when jButton9 is clicked.

After Click Buy Now Button

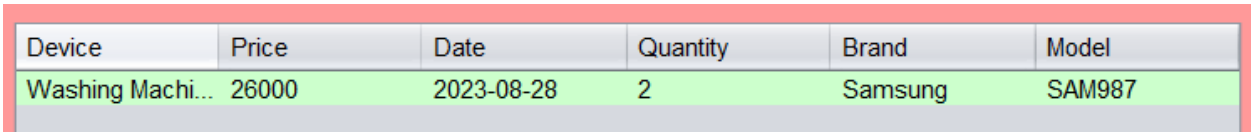
```
    private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
StringBuilder dataText = new StringBuilder();
int totalPrice = 0;
int totalQuantity = 0;

for (retail r : lb) {
    dataText.append(str: "Device: ").append(str: r.device).append(str: "\n");
    dataText.append(str: "Price: ").append(i: r.price).append(str: "\n");
    dataText.append(str: "Date: ").append(str: r.date).append(str: "\n");
    dataText.append(str: "Quantity: ").append(i: r.quantity).append(str: "\n");
    dataText.append(str: "Brand: ").append(str: r.brand).append(str: "\n");
    dataText.append(str: "Model: ").append(str: r.model).append(str: "\n");
    dataText.append(str: "Radio Value: ").append(str: r.radioValue).append(str: "\n");
    dataText.append(str: "--------------------\n");

    totalPrice += r.price * r.quantity; // Calculate total price for this item
    totalQuantity += r.quantity; // Accumulate total quantity
}

dataText.append(str: "Total Price: ").append(i: totalPrice).append(str: "\n");
dataText.append(str: "Total Quantity: ").append(i: totalQuantity);

jTextArea1.setText(t: dataText.toString());
jTabbedPane1.setSelectedIndex(index: 2);
    }
```

When the jbutton13 is clicked, it gathers information about retail items from a list called 'lb'. It then creates a formatted text containing details like device, price, date, quantity, brand, model, and radio value for each item. The code also calculates the total price by multiplying each item's price with its quantity and adds up the quantities. Finally, it displays this information, including the total price and total quantity, in a JTextArea component of the third Shopping Receipt panel. This helps users see a summary of the retail items and their costs.

Output:

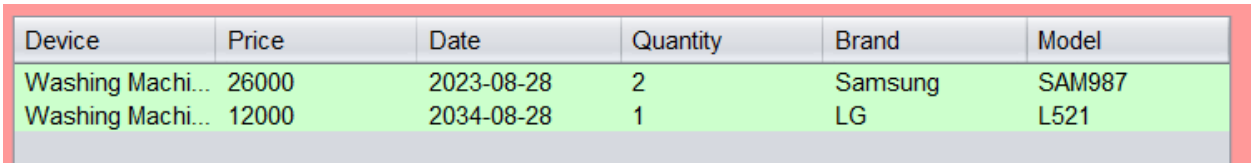


After Click on Insert, this button enter data in Table:



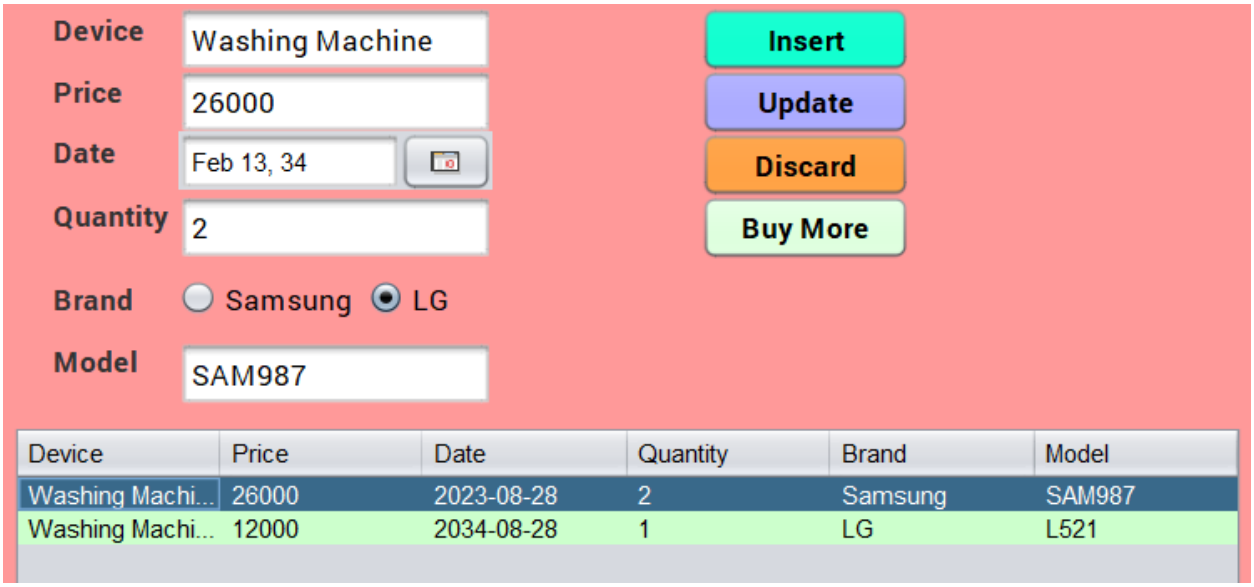
Device	Price	Date	Quantity	Brand	Model
Washing Machi...	26000	2023-08-28	2	Samsung	SAM987

Then I add one more data:

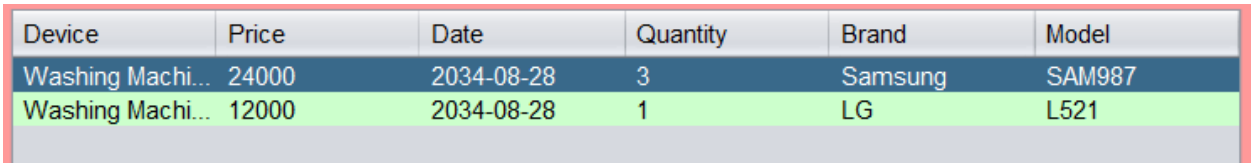


Device	Price	Date	Quantity	Brand	Model
Washing Machi...	26000	2023-08-28	2	Samsung	SAM987
Washing Machi...	12000	2034-08-28	1	LG	L521

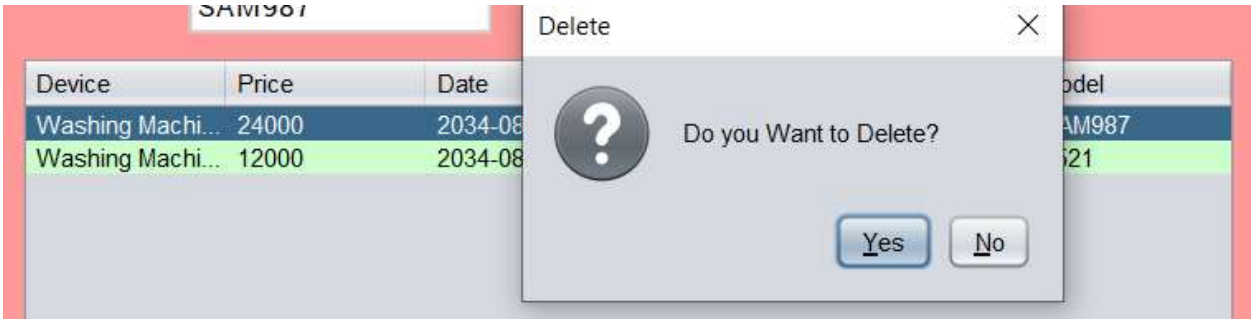
After Click in Table row one, then all data again display on Text Filed, then I can easily modified those data and update it by click Update Button:



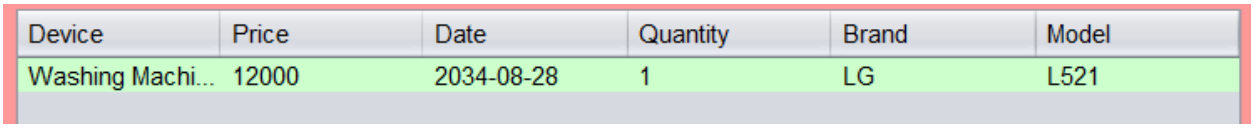
After Update:



Then I select 1st row on table and click Discard. That data was removed:

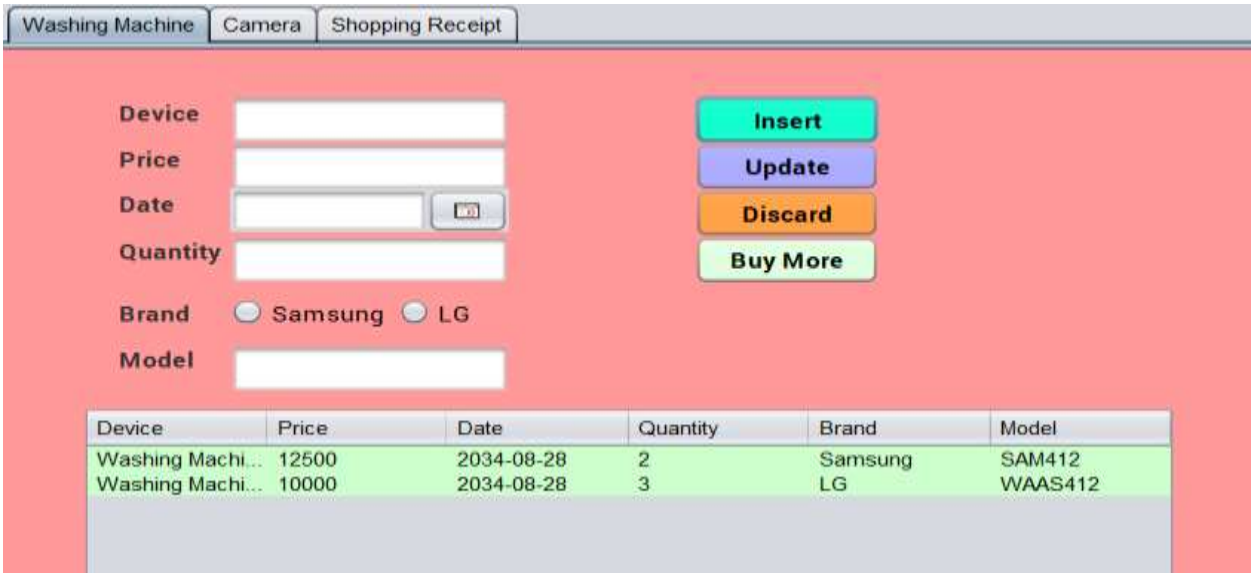


1st row deleted:

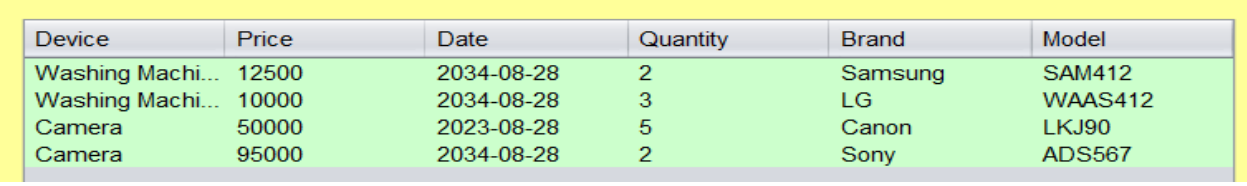


Then I click Buy more button for open Camera Panel:

Insert, Update and Discard work same like as Washing Machine Panel. So, I don't show them there work again. After click back then open Washing Machine panel again for buy more, that's why I go back that panel and add another data:



Came back Camera Panel and add more data for final Shopping Receipt calculation:



After click Buy Now Button, then **Final Output** showing in Shopping Receipt Panel:

In this final output we can see, Total Price and Total Quantity for selective items.

Conclution:

The code you've shown is about an "Electronics" package that contains a class called "retail." This class is used to store information about electronic devices in a store. Each device has a price, date, quantity, brand, and model. The class has a constructor to set these values when creating a new device object. There's also a method to display these device details.

In conclusion, the code defines a way to represent and manage electronic devices in a store, keeping track of their important information like device, price,date,quantity, brand and model.