



New Era University
College of Computer Studies
Computer Science Department



Point-of-Sale System for Small Business

In Partial Fulfillment of the Requirements in
CCS222-18/CSL222-18 Object-Oriented Programming

Submitted by:

Soriano, Jannah R.
Torres, Rosh Hashana S.

Class Schedule:

Saturday, 8:00 AM - 1:00 PM

Submitted to:

Edilberto L. Simbulan Jr.
Instructor

Table of Contents

I. Introduction	1
II. Scope and Limitation.....	2
III. System Layout	3-6
IV. Object-Oriented Programming Concepts	7-10

I. Introduction

A Point-of-Sale (POS) is a place where a customer executes the payment for goods or services and where sales taxes may become payable.

Over time, technology is growing so fast. Likewise, the technology in running trade or sale. Errors in recording, the calculation is less precise, speed in service is often constrained in the Micro, Small and Medium Enterprises and even in small businesses particularly in trade or sale.

A lot of small businesses still use the manual recording in their sales transactions and sales reports. This main problem can be solved by providing training in terms of management and partnership, in addition to all aspects of marketing and production even is utilize by relying on technology penetration.

Using a Point of Sales application, you can gain some benefit by the added value that can be administered in the form of improved quality of service, improved business image, competitive advantage, and late controlling and decision-making processes.

The use of technology can be a solution in the recording and processing of transactions so it is done by a computerized system, as it will speed up the transaction process and gives accurate results so that a more effective and efficient. The computerized transaction provides convenience in work and a definitive result for the buyer or custom.

II. Scope and Limitation

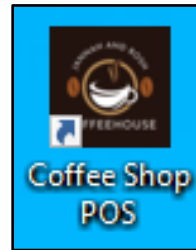
Scope:

The Point-of-Sale System is intended for small business enterprises as it only has limited functionalities. This system shows the price list and allows the printing of order lists and the creation of receipts based on the transactions performed. It calculates simple sales transactions showing the total amount of sale, customer's mode of payment, and change.

Limitation:

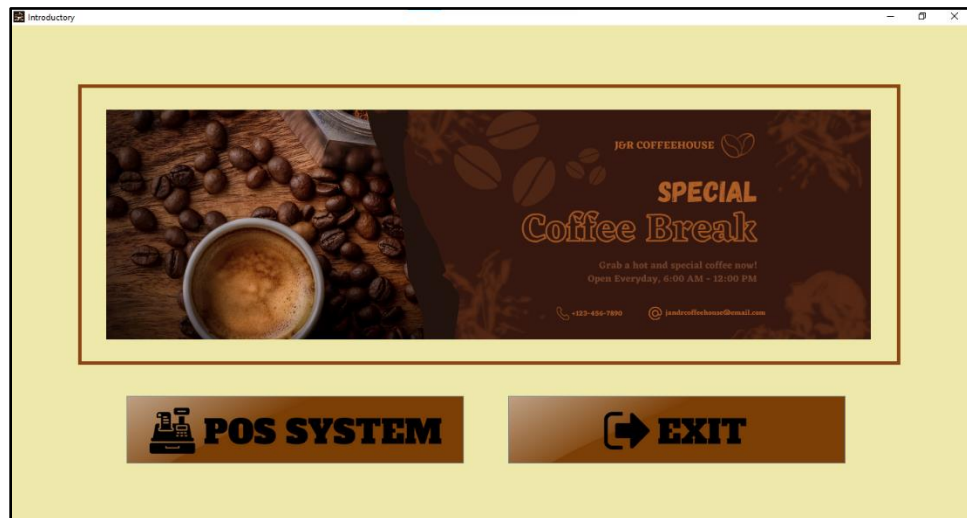
The Point-of-Sale System does not cover the backtracking of the transaction history. As it only provides simple calculations and limited functionalities, it cannot cover large business enterprises. Furthermore, the inventory of the products is not recorded in the system.

III. System Layout



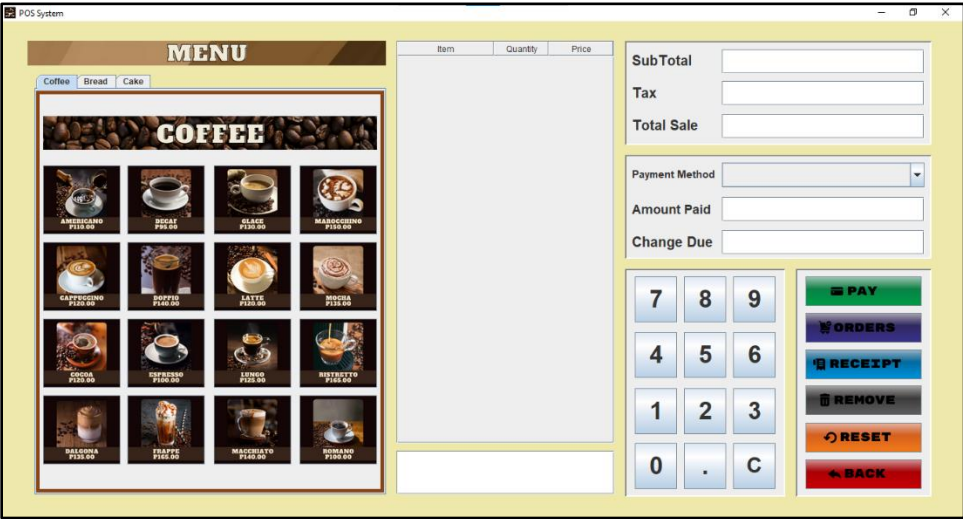
Coffee Shop POS

Introductory Window:



The Point-of-Sale System has an appealing user-friendly interface. An introductory window will be popped up once you opened the application. There were two buttons on this window, the POS System and Exit. When you click the POS System button, it will direct you to the main window while for the Exit button, it will close the application.

Main Window - POS System:



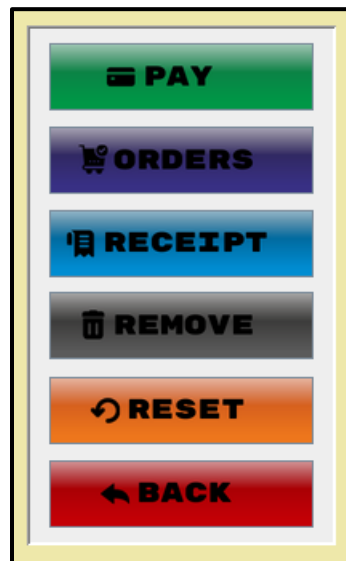
The main window contains Point-of-Sale System which is divided into three portions. On the left screen display, the system consists of menu buttons where products are listed.



In the middle portion, a summary of customer orders will be shown after transactions have been made and there are barcodes at the bottom screen for easy tracking of the products.



Lastly, the right screen display shows the calculations of the sales transaction. On the upper part, it shows the subtotal, tax, and total sale which are auto-calculated values. In the middle, displays the customer's mode of payment, the amount paid, and change. The values for the amount paid are entered into the system using a numeric keypad.



Beside the numeric keypad, there are six functional buttons. These are the following:

- **Pay** - Use this button to auto-calculate the customer's change. It will reflect in the 'Change Due' field.
- **Orders** - Use this button to print the customer's order list.

Printed Order List:

J&R Coffeehouse - Order List		
Item	Quantity	Price
Cocoa	1	120.0
Frappe	1	165.0
Marocchino	1	150.0
Butter Croissant	1	100.0
Chocolate Brownies	1	95.0
Glazed Doughnut	1	90.0
Blueberry Cheesecake	1	1045.0
Choco Midnight Cake	1	630.0
Red Velvet Cake	1	685.0


- **Receipt** - Use this button to view the receipt in the receipt window.


Receipt Window:

Receipt

RECEIPT

Item	Quantity	Price
Cocoa	1	120.0
Frappe	1	165.0
Marocchino	1	150.0
Butter Croissant	1	100.0
Chocolate Bro.	1	95.0
Glazed Dough.	1	90.0
Blueberry Cha	1	1045.0
Choco Midnight	1	630.0
Red Velvet Cake	1	685.0





Time: 8:50:22
Date: 5-27-2022

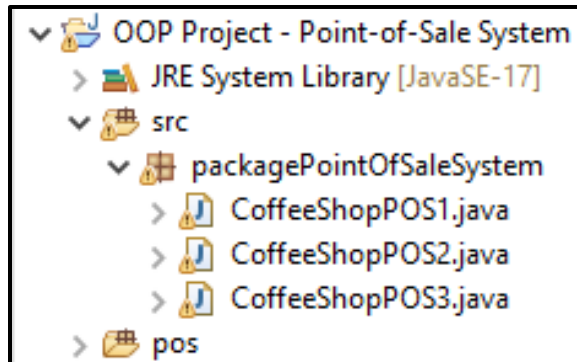
SubTotal: ₱ 3080.00
Tax: ₱ 7.70
Total Sale: ₱ 3087.70
Amount Paid: 3100.00
Change Due: ₱ 12.30

[← BACK](#)

- **Remove** - Use this button to remove a customer's order.
- **Reset** - Use this button to reset the point-of-sale system.
- **Back** - Use this button to go back to the introductory window.

IV. Object-Oriented Programming Concepts

Classes and Objects:



```
CoffeeShopPOS1 window = new CoffeeShopPOS1();
```

```
CoffeeShopPOS2 nw = new CoffeeShopPOS2();
```

```
CoffeeShopPOS3 window = new CoffeeShopPOS3();
```

A class describes how a specific sort of object will appear. A declaration and a definition make up a class description. Objects are instances of a class created with specifically defined data. It can correspond to real-world objects or an abstract entity. As seen in the system, three objects were created for the windows that will store each variable to access a shared member through the object's variable.

Methods:

```
public void Change()
{
    double sum = 0;
    double tax = 0.25;
    double cash = Double.parseDouble(jtxtAmountPaid.getText());

    for(int i = 0; i < table_OrderList.getRowCount(); i++)
    {
        sum = sum + Double.parseDouble(table_OrderList.getValueAt(i, 2).toString());
    }

    double cTax = (tax * sum)/100;
    double cChange = (cash - (sum + cTax));
    String ChangeGiven = String.format("P %.2f", cChange);
    jtxtChangeDue.setText(ChangeGiven);
}
```

A method is a collection of statements that perform some specific tasks and return the result to the caller. It can perform some specific tasks without returning anything. As seen in the system, methods were used to

provide information about, and access to, a single method on a class or interface.

Encapsulation:

```
public void actionPerformed(ActionEvent e) {  
    TableModel table = table_OrderList.getModel();  
    String outputBarcode = jtxtBarcode.getText();  
    String outputSubTotal = jtxtSubTotal.getText();  
    String outputTax = jtxtTax.getText();  
    String outputTotalSale = jtxtTotalSale.getText();  
    String inputAmountPaid = jtxtAmountPaid.getText();  
    String outputChangeDue = jtxtChangeDue.getText();  
  
    CoffeeShopPOS3 window = new CoffeeShopPOS3();  
    window.table_OutputOrderList.setModel(table);  
    window.txtOutputBarcode.setText(outputBarcode);  
    window.lblOutputSubtotal.setText(outputSubTotal);  
    window.lblOutputTax.setText(outputTax);  
    window.lblOutputTotalSale.setText(outputTotalSale);  
    window.lblInputAmountPaid.setText(inputAmountPaid);  
    window.lblOutputChangeDue.setText(outputChangeDue);  
  
    window.frmReceipt.setVisible(true);  
    frmPOS.dispose();  
}
```

```
private JLabel Time;  
private JLabel Date;
```

Encapsulation works by combining the scripts and manipulating the data in one location while keeping them safe from outside interaction and misuse. It also refers to allowing the user to access and modify the state of an object through their actions. Also, it is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.

The Encapsulation concept was utilized in the Point-of-Sale System, which is described as the technique of keeping fields within a class private and then making them accessible via public methods. It's a safeguard that keeps data and code secure inside the class. Thus, objects may be reused objects such as code components or variables without providing system-wide data access. This concept is significantly used throughout the system, as inputs taken from the program are assigned to different classes. For example, after the user clicks either the POS System or Exit button, the

variables inside their respective class will be assigned according to their respective input data.

Inheritance:

```
public void actionPerformed(ActionEvent e) {  
    frmHome = new JFrame("Exit");  
  
    if(JOptionPane.showConfirmDialog(frmHome, "Confirm if you want to exit", "Point of Sale",  
        JOptionPane.YES_NO_OPTION) == JOptionPane.YES_NO_OPTION)  
    {  
        System.exit(0);  
    }  
}
```

```
public void actionPerformed(ActionEvent e) {  
    //-----POS System Window-----  
    CoffeeShopPOS2 nw = new CoffeeShopPOS2();  
    nw.frmPOS.setVisible(true);  
    frmHome.dispose();  
  
}
```

Inheritance is a concept that acquires the properties from one class to other classes. In Java, a class can inherit attributes and methods from another class. The class that inherits the properties is known as the sub-class or the child class. The class from which the properties are inherited is known as the superclass or the parent class. The properties of the base class are acquired by the derived classes.

As seen in the system, inheritance uses the “actionPerformed” keyword, a simple technique used as a pattern if there is a need to make a callback to the parent class – the child class must be extended with some methods to hold a reference to the parent. The parent must implement a new interface the child will use to signal the parent.

Polymorphism:

```
table_OutputOrderList.setModel(new DefaultTableModel(  
    new Object[][] {  
    },  
    new String[] {  
        "Item", "Quantity", "Price"  
    }  
));  
  
public void actionPerformed(ActionEvent e)  
{  
    double priceOfItem = 110;  
    DefaultTableModel model = (DefaultTableModel) table_OrderList.getModel();  
    model.addRow(new Object[] {"Americano", "1", priceOfItem});  
    ItemCost();  
}
```

Polymorphism is a crucial notion. It merely indicates that there are multiple forms. That is, in different contexts, the same entity like method, operator, or object might perform multiple activities. It allows you to use these inherited properties to perform different tasks. Thus, allowing you to achieve the same action in many ways.

A superclass named "table_OutputOrderList.setModel" has a method "DefaultTableModel()". The subclasses of table_OutputOrderList.setModel can be Item, Quantity, and Price, each subclass has its way of calculating area. Using Inheritance and Polymorphism means, that the subclasses can use the DefaultTableModel() method to find the area's formula for that table_OutputOrderList.setModel.