COLLEGE OF
COMPUTING,
INFORMATICS AND
MATHEMATICS
KOLEJ PENGAJIAN PENGKOMPUTERAN,
INFORMATIK DAN MATEMATIK

UNIVERSITI TEKNOLOGI MARA PAHANG

27600 Raub, Pahang

PROGRAMMING PARADIGMS

(CSC305)

GROUP PROJECT ASSIGNMENT:

INVENTORY MANAGEMENT SYSTEM FOR A UCPh RETAIL STORE

LECTURER:

TS. MOHD NORAFIZAL ABD AZIZ

GROUP: CDCS1103G

PREPARED BY:

Student ID	Name
2022629194	KHADIJAH SAQIFAH BINTI ZAKIR HAMDI
2022823346	NOR KAMALIA BINTI JAMIL SYUKRAN
2022827578	NUR FARDINA BTE FAUZI

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA

CSC305: PROGRAMMING PARADIGM
SEMESTER OCTOBER 2023–February 2024

GROUP PROJECT
(30%) (CLO3, P3)

Please read the instructions as follows:

1. This is group assignment that consists of 2–3 members for each group.
2. Please complete the group details as required in Section A.
3. Please read the questions carefully and give your answer in the section provided in this template.
4. For any intention of plagiarism or any misconduct in this assignment, you will be penalized according to the rules, and a zero mark will be given due to the action. Further action will be considered according to the Academic UiTM rules and regulations in the Buku Peraturan Akademik UiTM (Pindaan 2017) and Akta Plagiarism UiTM (2017).
5. The rubric is also attached to this assignment for guidance and reference.
6. This assignment's due date and submission are before **January 12, 2024 (Friday).**
7. All submissions must be submitted in the **Google Classroom**.
8. Good Luck

Section A

Declaration (please read the following):

- I confirm that I have read and shall comply with all the terms and conditions of the UiTM plagiarism policy.
- I confirm that the online submission had also been sent accordingly with the attachment through the UFUTURE UiTM, and I had already checked the submission regularly before submission.
- We declare that this assignment is free from plagiarism and is our own properly derived work.
- We also reaffirm that anti-plagiarism software (if applicable) has checked the same work, as appropriate.

No	Student Name	Student ID	Class	Declaration Signature
1	KHADIJAH SAQIFAH BINTI ZAKIR HAMDI	20226292194	CDCS103G	
2	NUR FARDINA BTE FAUZI	2022827578	CDCS1103G	
3	NOR KAMALIA BINTI JAMIL SYUKRAN	2022823346	CDCS1103G	
Lecturer:		**Ts. MOHD NORAFIZAL ABD AZIZ**		

Case Study Title: Inventory Management System for a UCPh Retail Store

Objective: The objective of this case study is to design and implement an inventory management system using either object-oriented programming or imperative programming. The system should allow a retail store to keep track of its products, manage sales, and generate reports.

Scenario:

The end-user imagines having a small retail store that needs a computerized system to manage its inventory. The store sells various products, each with a unique product ID, name, price, and quantity in stock. The system should also handle customer sales and provide information about product availability.

Requirements:

1. *Product Class*: Create a product class that contains attributes such as product ID, name, price, and quantity in stock. Implement methods to add new products, update product information, and check product availability

2. *Customer Class*: Implement a customer class with attributes like customer ID, name, and purchase history. Implement methods for registering new customers and recording sales.

3. *Transaction Class*: Create a transaction class to handle sales transactions. It should record the products sold, quantities, and the total price for each transaction.

4. *Inventory Class*: Design an inventory class that manages the store's product inventory. It should include methods for adding and updating products, handling sales, and generating inventory reports.

5. *Menu System*: Develop a menu-based user interface that allows store employees to interact with the system. The menu should provide options for adding and updating products, recording sales, and generating reports.

6. *Data Storage*: Implement a simple file-based system to store product, customer, and transaction data persistently. You can use text files to store this data.

7. *Error Handling*: Implement error handling for cases such as invalid inputs, out-of-stock products, and incorrect pricing.

8. Reports: Create the ability to generate reports, such as a list of all products in stock, sales reports by date or customer, and revenue reports.

9. *Other Requirements*:

a. Optimize the system for performance, especially when dealing with a large number of products and transactions.
b. Allow for discounts, promotions, and loyalty programs for customers, and incorporate these into the sales and reporting mechanisms.

GROUP TASK

1. Consider the above requirements in developing the inventory management system as proposed.
2. Suggest your solution using the object-oriented and imperative programming languages. Therefore, your group must code in both languages.
3. Finally, discussed the differences and advantages of each language in providing the solution for the above case study.

END OF QUESTIONS

TABLE OF CONTENTS

1.0 - PROGRAM CODES

C PROGRAMMING

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Product {
    int product_ID;
    char name[50];
    double price;
    int quantity_in_stock;
};

struct Transaction {
    struct Product product;
    int quantity;
    double total_price;
};

struct Customer {
    char customer_ID[5];
    char cust_name[50];
    char loyal[3];
    struct Transaction purchase_history[700];
    int purchase_count;
```

```
};

struct Inventory {
    struct Product products[700];
    int product_count;
};

void add_product(struct Inventory *inventory);
void update_product(struct Inventory *inventory, int
product_ID);
void record_sale(struct Inventory *inventory, struct Customer
*customer, struct Transaction *transaction);
void generate_inv_report(struct Inventory *inventory);
void generate_cust_report(struct Customer *customer);
void menu(struct Inventory *inventory);
void load_data(struct Inventory *inventory, const char
*filename);
void save_data(struct Inventory *inventory, const char
*filename);

int main() {
    struct Inventory inventory;
    struct Customer tempCustomer;
    inventory.product_count = 0;

    load_data(&inventory, "data.txt");

    int option;
    do {
```

```
        menu(&inventory);
        printf("ENTER OPTION: ");
        scanf("%d", &option);

        switch (option) {
            case 1:
                add_product(&inventory);
                break;

            case 2:
                {
                    int product_ID_update;
                    printf("ENTER PRODUCT ID TO UPDATE: ");
                    scanf("%d", &product_ID_update);
                    if (product_ID_update >= 1 &&
product_ID_update <= inventory.product_count) {
                        update_product(&inventory,
product_ID_update);
                    } else {
                        printf("INVALID INPUT!\n");
                    }
                }
                break;

            case 3:
                {
                    printf("Enter customer ID: ");
                    scanf("%s", tempCustomer.customer_ID);
                    printf("Enter customer name: ");
```

```
                    scanf("%s", tempCustomer.cust_name);
                    printf("Eligible for membership? (y/n): ");
                    scanf("%s", tempCustomer.loyal);
                    printf("\n+--------------------------------
--------+\n");
                    struct Transaction transaction;
                    printf("\nENTER ID OF PRODUCT SOLD: ");
                    scanf("%d",
&transaction.product.product_ID);
                    printf("ENTER QUANTITY SOLD: ");
                    scanf("%d", &transaction.quantity);


                    record_sale(&inventory, &tempCustomer,
&transaction);
                }
                break;


            case 4:
                {
                    int report;
                    printf("+-----------------------------+\n");
                    printf("|       GENERATE REPORT       |\n");
                    printf("+-----------------------------+\n");
                    printf(" 1. Generate Inventory Report\n");
                    printf(" 2. Generate Customer Report\n");
                    printf("+-----------------------------+\n");
                    printf("ENTER OPTION: ");
                    scanf("%d", &report);
                    printf(" \n");
```

```
                    if (report == 1) {
                        generate_inv_report(&inventory);
                    } else if (report == 2) {
                        generate_cust_report(&tempCustomer);
                    } else {
                        printf("INVALID INPUT!\n");
                    }
                }
                break;

            case 5:
                printf("QUITTING PROGRAM...\n");
                break;

            default:
                printf("INVALID INPUT!\n");
        }
    } while (option != 5);

    save_data(&inventory, "data.txt");

    return 0;
}

void add_product(struct Inventory *inventory) {
    inventory->product_count++;
    struct Product *new_product = &(inventory-
>products[inventory->product_count - 1]);
```

```
    new_product->product_ID = inventory->product_count;
    printf("Enter product name: ");
    scanf("%s", new_product->name);
    printf("Enter product price: RM");
    scanf("%lf", &new_product->price);
    printf("Enter quantity in stock: ");
    scanf("%d", &new_product->quantity_in_stock);
    printf("[ PRODUCT ADDED! ]\n");
}

void update_product(struct Inventory *inventory, int product_ID)
{
    if (product_ID < 1 || product_ID > inventory->product_count)
{
        printf("INVALID PRODUCT ID!\n");
        return;
    }

    int choice = 0;
    printf("1 - UPDATE NAME\n2 - UPDATE PRICE\n3 - UPDATE
QUANTITY\nENTER A NUMBER (1/2/3): ");
    scanf("%d", &choice);
    if (choice == 1) {
        printf("Enter new product name: ");
        scanf("%s", inventory->products[product_ID - 1].name);
    } else if (choice == 2) {
        printf("Enter new product price: RM");
        scanf("%lf", &inventory->products[product_ID -
1].price);
```

```
    } else if (choice == 3) {
        printf("Enter new product quantity: ");
        scanf("%d", &inventory->products[product_ID -
1].quantity_in_stock);
    }
    printf("[ PRODUCT UPDATED! ]\n");
}


void record_sale(struct Inventory *inventory, struct Customer
*customer, struct Transaction *transaction) {
    if (transaction->product.product_ID < 1 || transaction-
>product.product_ID > inventory->product_count) {
        printf("INVALID PRODUCT ID!\n");
        return;
    }


    struct Product *sold_product = &(inventory-
>products[transaction->product.product_ID - 1]);
    if (sold_product->quantity_in_stock < transaction->quantity)
{
        printf("OUT OF STOCK!\n");
        return;
    }


    sold_product->quantity_in_stock -= transaction->quantity;


    // loyal discount
    double discount_price = (customer->loyal[0] == 'y' ||
customer->loyal[0] == 'Y') ?
        sold_product->price * 0.9 : sold_product->price;
```

```
    transaction->total_price = transaction->quantity *
discount_price;


    printf("TOTAL PRICE: RM%.2f\n", transaction->total_price);

    printf("[ SALE RECORDED! ]\n");


    customer->purchase_history[customer->purchase_count] =
*transaction;

    customer->purchase_count++;

}


void generate_inv_report(struct Inventory *inventory) {

    printf("+-----------------------------+\n");

    printf("|       INVENTORY REPORT      |\n");

    printf("+-----------------------------+\n");

    for (int i = 0; i < inventory->product_count; ++i) {

        struct Product *product = &(inventory->products[i]);

        printf("Product ID: %d\n", product->product_ID);

        printf("Name: %s\n", product->name);

        printf("Price: RM %.2lf\n", product->price);

        printf("Quantity in Stock: %d\n", product-
>quantity_in_stock);

        printf("+-----------------------------+\n");

    }

}


void generate_cust_report(struct Customer *customer) {

    printf("+-----------------------------+\n");
```

```
    printf("|       CUSTOMER REPORT       |\n");
    printf("+-----------------------------+\n");
    printf("Customer ID: %s\n", customer->customer_ID);
    printf("Name: %s\n", customer->cust_name);
    printf("Membership Eligibility: %s\n", customer->loyal);
    printf("Purchase History:\n");
    for (int i = 0; i < customer->purchase_count; ++i) {
        struct Transaction *transaction = &(customer-
>purchase_history[i]);
        printf("  Product ID: %d\n", transaction-
>product.product_ID);
        printf("  Quantity: %d\n", transaction->quantity);
        printf("+-----------------------------+\n");
    }
}

void menu(struct Inventory *inventory) {
    printf("+------------------------+\n");
    printf("| MENU                   |\n");
    printf("+------------------------+\n");
    printf("| 1. ADD PRODUCT         |\n");
    printf("+------------------------+\n");
    printf("| 2. UPDATE PRODUCT      |\n");
    printf("+------------------------+\n");
    printf("| 3. RECORD SALE         |\n");
    printf("+------------------------+\n");
    printf("| 4. GENERATE REPORT     |\n");
    printf("+------------------------+\n");
    printf("| 5. EXIT                |\n");
```

```
    printf("+-------------------------+\n");
}


void load_data(struct Inventory *inventory, const char
*filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("ERROR OPENING FILE!\n");
        return;
    }


    int count = 0;
    while (fscanf(file, "%d %s %lf %d", &inventory-
>products[count].product_ID,
           inventory->products[count].name, &inventory-
>products[count].price,
           &inventory->products[count].quantity_in_stock) == 4)
{
        count++;


        if (count >= 700) {
            printf("LIMIT OF PRODUCTS REACHED\n");
            fclose(file);
            return;
        }
    }


    inventory->product_count = count;
    fclose(file);
```

```
}


void save_data(struct Inventory *inventory, const char
*filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("ERROR OPENING FILE!\n");
        return;
    }

    for (int i = 0; i < inventory->product_count; ++i) {
        fprintf(file, "%d %s %.2lf %d\n", inventory-
>products[i].product_ID, inventory->products[i].name,
                inventory->products[i].price, inventory-
>products[i].quantity_in_stock);
    }

    fclose(file);
}
```

JAVA PROGRAMMING

Product Class - Java

```
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.File;

public class Product {
    private int productID;
```

```
    private String name;
    private double price;
    private int quantityInStock;

    public Product(){}

    public Product(int id, String n, double p, int q) {
        this.productID = id;
        this.name = n;
        this.price = p;
        this.quantityInStock = q;
    }

    //mutator
    public void setProductID(int id){this.productID = id;}
    public void setName(String n) {this.name = n;}
    public void setPrice(double p){this.price = p;}
    public void setQuantityInStock(int q){this.quantityInStock =
q;}

    //accessor
    public int getProductID () {return this.productID;}
    public String getName (){return this.name;}
    public double getPrice (){return this.price;}
    public int getQuantityInStock() { return
this.quantityInStock;}

    public void updateProduct(int newProductID, String newName,
double newPrice, int newQuantityInStock) {
```

```
        if (newProductID != -1) {
            this.setProductID(newProductID);
        }
        if (newName != null) {
            this.setName(newName);
        }
        if (newPrice != -1) {
            this.setPrice(newPrice);
        }
        if (newQuantityInStock != -1){
            this.setQuantityInStock(newQuantityInStock);
        }
    }

    public void addProduct(boolean append) {
        try{
            PrintWriter pw = new PrintWriter(new FileWriter(new
File("Products.txt"), append));
            String input = this.getProductID() + "-" +
this.getName() + "-" + this.getPrice() + "-" +
getQuantityInStock();
            pw.println(input);
            pw.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }
```

```
    public void checkAvailability() {
        if (quantityInStock>0) {
            System.out.println(name + "\tAVAILABLE " +
quantityInStock + " UNITS IN STOCK");
        } else {
            System.out.println(name + " OUT OF STOCK");
        }
     }

    public String toString(){
        return "ID: " + this.getProductID() +
               "\nName: " + this.getName() +
               "\nPrice: RM" + this.getPrice() +
               "\nQuantity: " + this.getQuantityInStock();
    }
}
```

Customer Class - Java

```
import java.util.ArrayList;
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.File;

public class Customer
{
    //attributes
    private String custID;
    private String name;
    private ArrayList<String> purchaseHistory;
```

```
//constructors
public Customer(){}
public Customer (String id, String n, ArrayList<String> ph)
{
    custID = id;
    name = n;
    purchaseHistory = ph;
}

//mutators
public void setCustID(String id)
{ custID = id;}
public void setName(String n)
{ name = n;}
public void setPurchaseHistory(ArrayList<String> ph)
{ purchaseHistory = ph;}

//accessors
public String getCustID()
{ return custID;}
public String getName()
{ return name;}
public ArrayList<String> getPurchaseHistory()
{ return purchaseHistory;}

//printer
public String toString()
```

```
    {
        return ("\nCustomer ID: " + custID + "\nName: " + name
);
    }

    public void recordSales(Transaction tran){
        try{
            PrintWriter pw = new PrintWriter(new FileWriter(new
File("Sales.txt."), true));
            String input = "CUSTOMER ID: " + this.getCustID() +
tran.displayReceipt();
            pw.println(input);
            pw.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }

    public void registerNewCustomer() {
        try{
            PrintWriter pw = new PrintWriter(new FileWriter(new
File("Customer.txt"), true));
            String input = this.getCustID() + "-" +
this.getName();
            pw.println(input);
            pw.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
```

```
        }
    }

    public boolean isLoyaltyProgram(){
        boolean loyal = false;
        if(this.getPurchaseHistory().size() >= 5)
            loyal = true;

        return loyal;
    }
}
```

Inventory Class - Java

```
import java.util.ArrayList;
import java.util.Scanner;

public class Inventory {
    private ArrayList<Product> products;

    public Inventory() {}

    public Inventory(ArrayList<Product> p){
        products = p;
    }

    //mutators
    public void setProducts(ArrayList <Product> p)
    { products = p;}
```

```
    //accessors
    public ArrayList <Product> getProducts()
    { return products;}

    public void addProduct(Product product) {
        product.addProduct(true); //
        ArrayList<Product> prodList = this.getProducts();
        prodList.add(product);
        this.setProducts(prodList);
    }

    public void sellProduct(Customer cust, Transaction tran) {
        Scanner in = new Scanner(System.in);
        System.out.println("\nYour total is RM" +
String.format("%.2f", tran.totalPrice()));
        double payment = 0;
        boolean invalid = true;
        while(invalid){
            System.out.println("\nEnter amount of payment: RM");
            payment = in.nextDouble();
            if(payment>=tran.totalPrice())
                invalid = false;
            if(invalid)
                System.out.println("\nSorry payment amount is
insufficient, please make new payment");
        }
        tran.setPayment(payment);
        System.out.println(tran.displayReceipt());
```

```
        cust.recordSales(tran);
        for(int i=0; i<tran.getProductSold().size(); i++){
            Product p = tran.getProductSold().get(i);
            Integer q = tran.getQuantitiesSold().get(i);
            int newQty = p.getQuantityInStock()-q;
            this.updateProduct(p.getProductID(), -1, null, -1,
newQty);
        }
    }

    public void updateProduct(int keyID, int productID, String
productName, double price, int quantity){
        ArrayList<Product> prodList = this.getProducts();

        for(int i = 0; i < prodList.size(); i++){
            Product p = prodList.get(i);
            if (p.getProductID()== keyID)
                p.updateProduct(productID, productName, price,
quantity);
            if (i == 0){ //read file - override file use the
first product
                p.addProduct(false);
            }
            else{
                p.addProduct(true);
            }
        }
        this.setProducts(prodList);
    }
```

```
    public String isLowOnStock(Product p){
        String stock = "";
        if (p.getQuantityInStock() < 10 ){
            stock = "Yes";
        }
        else{
            stock = "No";
        }
        return stock;
    }


    public String generateInventoryReport(){
System.out.println("Inventory Report:");
        String report = String.format("%-15s %-40s %-5s %-12s %-
17s %-12s %n", "\nProduct ID", "Product Name", "Qty",
"Price(RM)", "Stock Value(RM)", "Low On Stock?");
        report +=
"================================================================
=============================================\n";
        for(Product p: this.getProducts()){
            String price = String.format("%.2f", p.getPrice());
            String stockValue = String.format("%.2f",
p.getPrice()*p.getQuantityInStock());
            report += String.format("%-15s %-40s %-5d %-12s %-
17s %-12s %n", p.getProductID(), p.getName(),
p.getQuantityInStock(), price, stockValue,
this.isLowOnStock(p));
        }
        return report;
    }
```

```
    public void saleReportCustomer(ArrayList <Customer>
custList, String keyID){
        System.out.println("\nSales Report by Customer:\n");
        for(Customer c : custList){
            if(c.getCustID().equalsIgnoreCase(keyID)){
                for(String ph : c.getPurchaseHistory()){
                    System.out.println(ph);
                }
            }
        }
    }

    public void saleReportDate(ArrayList <String> sale, String
keyDate){
        System.out.println("\nSales Report by Date:");
        for(int i=0; i<sale.size(); i++){
            String s = sale.get(i); //read index ,customerID
            if(s.length() < 15){

            }
            else if (s.substring(6).equalsIgnoreCase(keyDate)){
                i--;
                s = sale.get(i);
                System.out.println("\n" + s);
                while
(!s.substring(0,7).equalsIgnoreCase("Payment")){
                i++;
                s = sale.get(i);
```

```
                System.out.println(s);
                }
            }
        }
    }


    public void revenueReport(ArrayList <String> sale, String
keyMonth){
        int quantity = 0;
        double totalPrice = 0.0;
        System.out.println("\nRevenue Report:");
        for(int i=0; i<sale.size(); i++){
            String s = sale.get(i);
            if(s.length() < 15){
                //
            } else
if(s.substring(9,11).equalsIgnoreCase(keyMonth)){
                i = i + 5;
                s = sale.get(i);
                String num = s.substring(31);
                char[] a = num.toCharArray();
                String qty = "";
                for(char b : a){
                    if(b == ' '){
                        break;
                    }
                    qty += b;
                }
                quantity += Integer.parseInt(qty);
```

```
                i = i + 2;
                s = sale.get(i);

while(!s.substring(0,5).equalsIgnoreCase("Total")){
                    quantity +=
Integer.parseInt(s.substring(31,32));
                    i = i + 2;
                    s = sale.get(i);
                }
                i++;
                s = sale.get(i);
                totalPrice +=
Double.parseDouble(s.substring(15));
            }
        }
        System.out.println("Number of units sold: " + quantity);
        System.out.println("Sales Revenue: RM" + totalPrice);
    }
}
```

Transaction class - Java

```
import java.util.ArrayList;
public class Transaction {
    private ArrayList <Product> productsSold;
    private ArrayList <Integer> quantitiesSold;
    private String date;
    private double payment;
    private double discount;
```

```
    public Transaction() {}


    public Transaction (ArrayList<Product> p, ArrayList<Integer>
q, double pt, String d, double dis)
    {
        productsSold = p;
        quantitiesSold = q;
        payment = pt;
        date = d;
        discount = dis;
    }
    //mutators
    public void setProductSold(ArrayList <Product> p)
    { productsSold = p;}
    public void setQuantitiesSold(ArrayList <Integer> q)
    { quantitiesSold = q;}
    public void setPayment(double pt)
    { payment = pt;}
    public void setDate(String d)
    { date = d;}
    public void setDiscount(double dis)
    { discount = dis;}

    //accessors
    public ArrayList <Product> getProductSold()
    { return productsSold;}
    public ArrayList <Integer> getQuantitiesSold()
    { return quantitiesSold;}
```

```
    public double getPayment()
    { return payment;}
    public String getDate()
    { return date;}
    public double getDiscount()
    { return discount;}

    public void addProduct(Product product, int quantity) {
//display the product name, quantity, and update the total
price.
        ArrayList<Product> prodList = this.getProductSold();
        prodList.add(product);
        this.setProductSold(prodList);
        ArrayList<Integer> qtyList = this.getQuantitiesSold();
        qtyList.add(quantity);
        this.setQuantitiesSold(qtyList);
    }

    public double totalPrice(){
        double price = 0.0;
        for(int i = 0; i <this.getProductSold().size(); i++ ){
            Product p = this.getProductSold().get(i);
            Integer q = this.getQuantitiesSold().get(i);
            price += (p.getPrice() * q);
        }
        price = price * (1 - (this.getDiscount()/100));
        return price;
    }
```

```
    public double calcBalance(){
        return this.getPayment() - this.totalPrice();
    }

    public String displayReceipt() {
        String receipt = "\nDate: " + this.getDate();
        receipt += "\nReceipt:";
        receipt += "\n+----------------------------------------
-------+";
        receipt += String.format("%-30s %-10s %n", "\nProduct
Name", "Quantity");
        receipt += "+----------------------------------------
-----+\n";
        for (int i = 0; i < this.getProductSold().size(); i++) {
//getName - product's name from product class
            receipt += String.format("%-30s %-10d %n",
this.getProductSold().get(i).getName(),
this.getQuantitiesSold().get(i));
            receipt += "+--------------------------------------
---------+\n";
        }
        receipt += "Discount: " + this.getDiscount() + "%";
        receipt += "\nTotal Price: RM" + this.totalPrice();
        receipt += "\nBalance: RM" + this.calcBalance();
        receipt += "\nPayment: RM" + this.getPayment() + "\n\n";
        return receipt;
    }
}
```

MenuSystem (Application)

```
import java.util.Scanner;
```

```
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.StringTokenizer;

public class MenuSystem {
    private static final Scanner in = new Scanner(System.in);
    private static final Scanner inputText = new
Scanner(System.in);
    static String wrongOutput = "";

    public static void main(String[] args) throws Exception {
        char choice = ' ';
        Inventory inventory = new Inventory(readFileProduct());

        while (choice != 'E' && choice != 'e') {
            System.out.println("\f" + wrongOutput + "\n========
Store Management System =========");
            wrongOutput = "";
            System.out.println("A - Manage Product");
            System.out.println("B - Manage Customer");
            System.out.println("C - Manage Transaction");
            System.out.println("D - Generate Report");
            System.out.println("E - Exit");

System.out.println("==========================================")
;
            System.out.print("Enter your choice [A/B/C/D/E]: ");
            choice = in.next().charAt(0);
```

```
            boolean repeat = true;
            if (choice == 'A' || choice == 'a') {
                while(repeat){
                    repeat = manageProduct(inventory);
                }
            } else if (choice == 'B' || choice == 'b') {
                while(repeat){
                    repeat = manageCustomer();
                }
            } else if (choice == 'C' || choice == 'c') {
                while(repeat){
                    repeat = manageTransaction(inventory);
                }
            } else if (choice == 'D' || choice == 'd') {
                while(repeat){
                    repeat = generateReport(inventory);
                }
            } else if (choice == 'E' || choice == 'e') {
                //
            }
            else{
                wrongOutput = "Invalid choice. Please enter a
valid option.\n";
            }
        }
        System.out.println("Exiting the system..... Goodbye!");
    }
```

```
    public static boolean manageProduct(Inventory inventory){
            char choice = ' ';
            System.out.println("\f" + wrongOutput +
"\n================= PRODUCT ===================");
            System.out.println("A - Add Product");
            wrongOutput = "";
            System.out.println("B - Update Product");
            System.out.println("C - Check Availability of the
Product");
            System.out.println("D - Exit");

System.out.println("============================================
=");
            System.out.print("Enter your choice [A/B/C/D]: ");
            choice = in.next().charAt(0);
            Product product = new Product();

            if (choice == 'A' || choice == 'a') {
                System.out.print("\nEnter product ID: ");
                int productId = in.nextInt();

                System.out.print("Enter product name: ");
                String productName = inputText.nextLine();

                System.out.print("Enter product price: ");
                double price = in.nextDouble();

                System.out.print("Enter quantity in stock: ");
```

```
                int quantityInStock = in.nextInt();
                product = new Product(productId, productName,
price, quantityInStock);
                inventory.addProduct(product);


            } else if (choice == 'B' || choice == 'b') {
                int keyID = prodIDErrorHandling(inventory,
false);

System.out.println("=====================================");
                System.out.println("A - Product ID");
                System.out.println("B - Product Name");
                System.out.println("C - Price of the Product");
                System.out.println("D - Quantity in stock:");

System.out.println("=====================================");
                System.out.print("Which details to be updated?
[A/B/C/D] : ");
                char answer = in.next().charAt(0);


                if (answer == 'A' || answer == 'a'){
                    System.out.println("\nProduct ID: ");
                    int productId = in.nextInt();
                    inventory.updateProduct(keyID,productId,
null, -1, -1);
                }
                else if(answer == 'B' || answer == 'b'){
                    System.out.println("Product Name: ");
                    String productName = inputText.nextLine();
```

```
                    inventory.updateProduct(keyID , -1,
productName, -1, -1);
                }
                else if(answer == 'C' || answer == 'c'){
                    System.out.print("Enter product price: ");
                    double price = in.nextDouble();
                    inventory.updateProduct(keyID, -1, null,
price, -1);
                }
                else if(answer == 'D' || answer == 'd'){
                    System.out.print("Enter quantity in stock:
");
                    int quantityInStock = in.nextInt();
                    inventory.updateProduct(keyID, -1, null, -1,
quantityInStock);
                }
                else {
                    System.out.println("\nInvalid choice. Please
enter a valid option.");
                }
            }


             else if (choice == 'C' || choice == 'c') {
                int input = prodIDErrorHandling(inventory,
false);
                for(int i = 0; i <
inventory.getProducts().size(); i++){
                    Product p = inventory.getProducts().get(i);
                    if (p.getProductID()== input){
                        p.checkAvailability();
```

```
                    break;
                }
            }

        } else if (choice == 'D' || choice == 'd') {
            return false;
        } else{
            wrongOutput = "Invalid input. Please try
again.\n";
            if(manageProduct(inventory))
            return true;
            else
            return false;

        }
        return repeatProcess();
    }

    public static ArrayList<Product> readFileProduct() throws
Exception{
        ArrayList<Product> list = new ArrayList<Product>();
            try{
            BufferedReader br = new BufferedReader(new
FileReader("Products.txt"));
            String line = br.readLine();
            while(line != null){
                StringTokenizer st = new StringTokenizer(line,
"-");
                int prodID = Integer.parseInt(st.nextToken());
```

```
                String prodName = st.nextToken();
                double price =
Double.parseDouble(st.nextToken());
                int qty = Integer.parseInt(st.nextToken());
                Product p = new Product(prodID, prodName, price,
qty);
                list.add(p);
                line = br.readLine();
            }
            br.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
        return list;
    }



    public static boolean manageCustomer() throws Exception {
            char choice = ' ';
            System.out.println("\f" + wrongOutput +
"\n============= CUSTOMER ============");
            wrongOutput = "";
            System.out.println("A - Register New Customer");
            System.out.println("B - View registered customer");
            System.out.println("C - Search registered
customer");
            System.out.println("D - Exit");

System.out.println("====================================");
```

```
            System.out.print("Enter your choice [A/B/C/D]: ");
            choice = in.next().charAt(0);
            Customer cust = new Customer();

            if (choice == 'A' || choice == 'a') {
                System.out.print("Enter customer ID: ");
                String custId = inputText.nextLine();

                System.out.print("Enter customer name: ");
                String custName = inputText.nextLine();

                cust = new Customer(custId, custName,new
ArrayList<String>());
                cust.registerNewCustomer();
            } else if (choice == 'B' || choice == 'b') {
                System.out.println("Registered Customer: ");
                ArrayList <Customer> custList =
readFileCustomer();
                for(Customer c : custList){
                    System.out.println(c.toString());
                }
            } else if(choice == 'C' || choice == 'c'){
                ArrayList <Customer> custList =
readFileCustomer();
                System.out.print("Enter customer name to be
searched: ");
                String n = inputText.nextLine();
                boolean found = false;
                for(Customer c : custList){
```

```
                    if(c.getName().equalsIgnoreCase(n)){
                        System.out.println(c.toString());


                        if(c.isLoyaltyProgram()){
                            System.out.println("\nThis customer
is eligible for the membership programme.");
                        }
                        found = true;
                    }
                }
                if(!found){
                    System.out.println("\nSorry there is no
customer with such name found.");
                }
            }
            else if (choice == 'D' || choice == 'd'){
                return false;
            }
            else {
                wrongOutput = "Invalid input. Please try
again.\n";
                if(manageCustomer())
                return true;
                else
                return false;


            }
            return repeatProcess();
    }
```

```
        public static ArrayList<Customer> readFileCustomer()
throws Exception{
        ArrayList<Customer> list = new ArrayList<Customer>();
            try{
            BufferedReader br = new BufferedReader(new
FileReader("Customer.txt"));
            String line = br.readLine();
            while(line != null){
                StringTokenizer st = new StringTokenizer(line,
"-");
                String custId= st.nextToken();
                String custName= st.nextToken();
                ArrayList <String> saleList = readFileSale();
                ArrayList <String> purchaseHistory = new
ArrayList <String>();
                for(int i=0; i<saleList.size(); i++){
                    String ph = "";
                    String s = saleList.get(i);
                     if(s.length() < 18){
                        //to avoid line less than 18
                    }
                    else if
(s.substring(13).equalsIgnoreCase(custId)){
                        ph += s + "\n";
                        while
(!s.substring(0,7).equalsIgnoreCase("Payment")){
                            i++;
                            s = saleList.get(i);
                            ph += s + "\n";
```

```
                        }
                        purchaseHistory.add(ph);
                    }
                }
                Customer c = new Customer(custId, custName,
purchaseHistory);
                list.add(c);
                line = br.readLine();
            }
            br.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
        return list;
    }


    public static boolean manageTransaction(Inventory inventory)
throws Exception {
            char choice = ' ';
            System.out.println( "\f" + wrongOutput +
"\n========== TRANSACTION ===========");
            wrongOutput = "";
            System.out.println("A. Make new Transaction");
            System.out.println("B. Exit");

System.out.println("=====================================");
            System.out.print("Enter your choice [A/B]: ");
            choice = in.next().charAt(0);
```

```
            if (choice == 'A' || choice == 'a') {
                ArrayList <Product> productsSold = new ArrayList
<Product>();
                ArrayList <Integer> quantitiesSold = new
ArrayList <Integer>();
                ArrayList <Customer> customer =
readFileCustomer();
                System.out.println("\nEnter date [DD/MM/YEAR]:
");
                String date = inputText.nextLine();
                ArrayList<Customer> custList =
readFileCustomer();
                String input = custIDErrorHandling(custList);
                Customer cust = new Customer();

                for(int i = 0; i<customer.size(); i++){
                    Customer c = customer.get(i);
                    if(c.getCustID().equalsIgnoreCase(input)){
                        cust = c;
                        break;
                    }
                }
                System.out.println("\nEnter Product ID and
Quantity of the Product to be sold: ");

                int cont = 0;
                Product prod = new Product();
                double discount = 0.0;
                if(cust.isLoyaltyProgram()){
```

```
                    discount = 10;
                }
                Transaction tran = new Transaction(productsSold,
quantitiesSold, 0, date, discount);


                while (cont != -1){
                    int productID =
prodIDErrorHandling(inventory, true);


                    for(int i = 0; i <
inventory.getProducts().size(); i++){
                        Product p =
inventory.getProducts().get(i);
                        if (p.getProductID()== productID){
                            prod = p;
                            break;
                        }
                    }


                    boolean invalid = true;
                    int quantity = 0;
                    while(invalid){
                        System.out.println("Quantity: ");
                        quantity = in.nextInt();
                        if(quantity<=prod.getQuantityInStock())
                            invalid = false;
                        if(invalid)
                            System.out.println("\nSorry, the
quantity exceeded current stock. Please input another
quantity.");
```

```
                    }
                    tran.addProduct(prod, quantity);

                    System.out.println("Continue? [enter -1 to
stop/ 0 to continue]: ");
                    cont = in.nextInt();
                }
                inventory.sellProduct(cust, tran);
            }
            else if (choice == 'B' || choice == 'b') {
                return false;
            }
            else{
                wrongOutput = "Invalid input. Please try
again.\n";
                if(manageTransaction(inventory))
                return true;
                else
                return false;

            }

            return repeatProcess();
    }

    public static boolean generateReport(Inventory inventory)
throws Exception {
            char choice = ' ';
```

```
            System.out.println("\f" + wrongOutput +
"\n============= Inventory ===============");
            wrongOutput = "";
            System.out.println("A - Inventory Report");
            System.out.println("B - Sales Report (by
customer)");
            System.out.println("C - Sales Report (by date)");
            System.out.println("D - Revenue report");
            System.out.println("E - Exit");

System.out.println("========================================");
            System.out.print("Enter your choice [A/B/C/D/E]: ");
            choice = in.next().charAt(0);



            if (choice == 'A' || choice == 'a') {

System.out.println(inventory.generateInventoryReport());
            } else if (choice == 'B' || choice == 'b') {
                ArrayList<Customer> custList =
readFileCustomer();
                String keyID = custIDErrorHandling(custList);
                inventory.saleReportCustomer(custList, keyID);
            } else if (choice == 'C' || choice == 'c') {
                System.out.println("Enter date [DD/MM/YEAR]: ");
                String keyDate = inputText.nextLine();
                ArrayList <String> sale = readFileSale();
                inventory.saleReportDate(sale, keyDate);
            }
             else if (choice == 'D' || choice == 'd') {
```

```
                System.out.println("Enter month: ");
                String keyMonth = inputText.nextLine();
                ArrayList <String> sale = readFileSale();
                inventory.revenueReport(sale, keyMonth);
            }
            else if (choice == 'E' || choice == 'e') {
                return false;

            }
            else{
                wrongOutput = "\nInvalid input. Please try
again.\n";
                if(generateReport(inventory))
                return true;
                else
                return false;

            }
            return repeatProcess();
    }

    public static ArrayList<String> readFileSale() throws
Exception{
        ArrayList<String> list = new ArrayList<String>();
        try{
            BufferedReader br = new BufferedReader(new
FileReader("Sales.txt"));
            String line = br.readLine();
            while(line != null){
```

```
                if(!line.equals(""))
                    list.add(line);
                line = br.readLine();
            }
            br.close();
        }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
        return list;
    }

    public static String custIDErrorHandling(ArrayList<Customer>
custList){
        boolean invalid = true;
        String keyID = "";
        while(invalid){
            System.out.print("\nPlease enter the ID for the
customer to be searched: ");
            keyID = inputText.nextLine();
            for(Customer c: custList){
                if(c.getCustID().equals(keyID)){
                    invalid = false;
                    break;
                }
            }
            if(invalid)
                System.out.println("\nCustomer ID is not found.
Please enter again.");
```

```
        }
        return keyID;
    }



    /*
     *
     */
    public static int prodIDErrorHandling(Inventory inventory,
boolean outOfStock){
        boolean invalid = true;
        int keyID = 0;
        while(invalid || outOfStock){
            boolean notFound = true;
            System.out.print("\nPlease enter the ID for the
product to be searched: ");
            keyID = in.nextInt();
            for(Product p: inventory.getProducts()){
                if(p.getProductID() ==keyID){
                    if(!outOfStock){
                        invalid = false;
                        notFound = false;
                    }
                    else{
                        if(p.getQuantityInStock() == 0){
                            System.out.println("\nThe product is
out of stock.");
                            notFound = false;
                        }
```

```
                        else{
                            invalid = false;
                            outOfStock = false;
                            notFound = false;
                        }
                    }
                    break;
                }
            }
            if(notFound)
                System.out.println("\nThe product ID does not
exists. Please try again.");
        }
        return keyID;
    }

    public static boolean repeatProcess(){
        char input;
        System.out.print("\nRepeat?: ");
        System.out.println("\n[enter X to return to menu,
otherwise enter Y to repeat this process]");
        input = in.next().charAt(0);
        if(input == 'X' || input == 'x')
            return false;
        else
            return true;
    }
}
```

2.0 - INPUT AND OUTPUT FOR EACH LANGUAGE

C PROGRAMMING

```
+-----------------------+
| MENU                  |
+-----------------------+
| 1. ADD PRODUCT        |
+-----------------------+
| 2. UPDATE PRODUCT     |
+-----------------------+
| 3. RECORD SALE        |
+-----------------------+
| 4. GENERATE REPORT    |
+-----------------------+
| 5. EXIT               |
+-----------------------+
ENTER OPTION:
```

```
+-----------------------+
| MENU                  |
+-----------------------+
| 1. ADD PRODUCT        |
+-----------------------+
| 2. UPDATE PRODUCT     |
+-----------------------+
| 3. RECORD SALE        |
+-----------------------+
| 4. GENERATE REPORT    |
+-----------------------+
| 5. EXIT               |
+-----------------------+
ENTER OPTION: 2
ENTER PRODUCT ID TO UPDATE: 1
1 - UPDATE NAME
2 - UPDATE PRICE
3 - UPDATE QUANTITY
ENTER A NUMBER (1/2/3): 2
Enter new product price: 15
[ PRODUCT UPDATED! ]
```

```
ENTER OPTION: 3
Enter customer ID: PD1
Enter customer name: JANE
Eligible for membership? (y/n): y

+----------------------------------------+

ENTER ID OF PRODUCT SOLD: 2
ENTER QUANTITY SOLD: 2
TOTAL PRICE: RM54.00
[ SALE RECORDED! ]
```

```
ENTER OPTION: 4
+----------------------------+
|       GENERATE REPORT      |
+----------------------------+
 1. Generate Inventory Report
 2. Generate Customer Report
+----------------------------+
ENTER OPTION: 1

+----------------------------+
|       INVENTORY REPORT     |
+----------------------------+
Product ID: 1
Name: HAT
Price: RM 10.00
Quantity in Stock: 87
+----------------------------+
Product ID: 2
Name: SHOES
Price: RM 30.00
Quantity in Stock: 78
+----------------------------+
Product ID: 3
Name: PANTS
Price: RM 10.00
Quantity in Stock: 58
+----------------------------+
Product ID: 4
Name: TROUSERS
Price: RM 20.00
Quantity in Stock: 40
+----------------------------+
```

```
+-------------------------+
ENTER OPTION: 4
+----------------------------+
|       GENERATE REPORT      |
+----------------------------+
 1. Generate Inventory Report
 2. Generate Customer Report
+----------------------------+
ENTER OPTION: 2

+----------------------------+
|       CUSTOMER REPORT      |
+----------------------------+
Customer ID: PD1
Name: JANE
Membership Eligibility: y
Purchase History:
  Product ID: 2
  Quantity: 2
+----------------------------+
```

```
+-----------------------+
| MENU                  |
+-----------------------+
| 1. ADD PRODUCT        |
+-----------------------+
| 2. UPDATE PRODUCT     |
+-----------------------+
| 3. RECORD SALE        |
+-----------------------+
| 4. GENERATE REPORT    |
+-----------------------+
| 5. EXIT               |
+-----------------------+
ENTER OPTION: 1
Enter product name: TROUSERS
Enter product price: RM20
Enter quantity in stock: 40
[ PRODUCT ADDED! ]
```

```
+-----------------------+
| MENU                  |
+-----------------------+
| 1. ADD PRODUCT        |
+-----------------------+
| 2. UPDATE PRODUCT     |
+-----------------------+
| 3. RECORD SALE        |
+-----------------------+
| 4. GENERATE REPORT    |
+-----------------------+
| 5. EXIT               |
+-----------------------+
ENTER OPTION: 5
QUITTING PROGRAM...
Program ended with exit code: 0
```

JAVA PROGRAMMING

```
======== Store Management System =========
A - Manage Product
B - Manage Customer
C - Manage Transaction
D - Generate Report
E - Exit
==========================================
Enter your choice [A/B/C/D/E]:
```

```
================ PRODUCT ==================
A - Add Product
B - Update Product
C - Check Availability of the Product
D - Exit
==========================================
Enter your choice [A/B/C/D]: a

Enter product ID: 3819388
Enter product name: Satin Shawl
Enter product price: 15
Enter quantity in stock: 2

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
================= PRODUCT ===================
A - Add Product
B - Update Product
C - Check Availability of the Product
D - Exit
=============================================
Enter your choice [A/B/C/D]: b

Please enter the ID for the product to be searched: 2094841
=====================================
A - Product ID
B - Product Name
C - Price of the Product
D - Quantity in stock:
=====================================
Which details to be updated? [A/B/C/D] : d
Enter quantity in stock: 40

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
================= PRODUCT ===================
A - Add Product
B - Update Product
C - Check Availability of the Product
D - Exit
=============================================
Enter your choice [A/B/C/D]: c

Please enter the ID for the product to be searched: 2023112
Bika    AVAILABLE 19 UNITS IN STOCK

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
============= CUSTOMER =============
A - Register New Customer
B - View registered customer
C - Search registered customer
D - Exit
=====================================
Enter your choice [A/B/C/D]: a
Enter customer ID: CD7898
Enter customer name: Hani Suraia

[enter X
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
============= CUSTOMER ============
A - Register New Customer
B - View registered customer
C - Search registered customer
D - Exit
===================================
Enter your choice [A/B/C/D]: b
Registered Customer:

Customer ID: CD5674
Name: Hazeeq

Customer ID: CD2020
Name: Khadijah

Customer ID: CD3465
Name: Fardina

Customer ID: CD6789
Name: Kamalia

Customer ID: CD3231
Name: Ilyana

Customer ID: CD4652
Name: Amirul
```

```
============= CUSTOMER ============
A - Register New Customer
B - View registered customer
C - Search registered customer
D - Exit
===================================
Enter your choice [A/B/C/D]: c
Enter customer name to be searched: Fardina

Customer ID: CD3465
Name: Fardina

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
========== TRANSACTION ===========
A. Make new Transaction
B. Exit
=======================================
Enter your choice [A/B]: a

Enter date [DD/MM/YEAR]:
15/01/2024

Please enter the ID for the customer to be searched: CD3465

Enter Product ID and Quantity of the Product to be sold:
2349809
Quantity:
2
Continue? [enter -1 to stop/ 0 to continue]:
```

```
============ Inventory ==============
A - Inventory Report
B - Sales Report (by customer)
C - Sales Report (by date)
D - Revenue report
E - Exit
=======================================
Enter your choice [1/2/3/4]: a
Inventory Report:
```

Product ID	Product Name	Qty	Price(RM)	Stock Value(RM)	Low On Stock?
2023290	Aik Cheong White Coffee	6	14.20	85.20	Yes
2022345	Wonda White Coffee	7	3.50	24.50	Yes
2022346	Kimball Chili Sauce	27	4.00	108.00	No
2028881	Kopiko	9	3.00	27.00	Yes
2022344	Serbuk Kari Adabi	10	1.60	16.00	No
2021231	Garam	6	2.00	12.00	Yes
2021121	Shokubutsu Body Wash	22	8.90	195.80	No
2094841	Ice Cream PaddlePop	40	1.00	40.00	No
2023112	Bika	19	3.00	57.00	No
2093475	Glo Dish Wash	0	14.60	0.00	Yes
8479281	Cucur Ikan Bilis	12	4.50	54.00	No
6536178	Mrs.Potato	12	2.50	30.00	No
2349809	Mineral Water	29	1.20	34.80	No
8471891	Chocolate Muffin	13	4.50	58.50	No
13123	Kopi	1000	12.50	12500.00	No
3819388	Satin Shawl	2	15.00	30.00	Yes

```
============ Inventory ==============
A - Inventory Report
B - Sales Report (by customer)
C - Sales Report (by date)
D - Revenue report
E - Exit
=====================================
Enter your choice [1/2/3/4]: b

Please enter the ID for the customer to be searched: CD2020

Sales Report by Customer:

CUSTOMER ID: CD2020
Date: 11/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             1
+-----------------------------------------------+
Garam                          1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM5.5
Balance: RM4.5
Payment: RM10.0
```

```
CUSTOMER ID: CD2020
Date: 02/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             2
+-----------------------------------------------+
Chocolate Muffin               1
+-----------------------------------------------+
Mineral Water                  1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM12.7
Balance: RM2.3000000000000007
Payment: RM15.0

CUSTOMER ID: CD2020
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             1
+-----------------------------------------------+
Aik Cheong White Coffee        1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM17.7
```

```
CUSTOMER ID: CD2020
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             1
+-----------------------------------------------+
Aik Cheong White Coffee        1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM17.7
Balance: RM2.3000000000000007
Payment: RM20.0

CUSTOMER ID: CD2020
Date: 04/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Kopiko                         1
+-----------------------------------------------+
Discount: 0.0%
Total Price: RM3.0
Balance: RM2.0
Payment: RM5.0

CUSTOMER ID: CD2020
```

```
CUSTOMER ID: CD2020
Date: 12/12/2023
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Aik Cheong White Coffee        34
+-----------------------------------------------+
Discount: 10.0%
Total Price: RM434.52
Balance: RM65.48000000000002
Payment: RM500.0

CUSTOMER ID: CD2020
Date: 12/12/2023
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Glo Dish Wash                   2
+-----------------------------------------------+
Discount: 10.0%
Total Price: RM26.28
Balance: RM3.719999999999999
Payment: RM30.0

CUSTOMER ID: CD2020
Date: 13/12/2023
Receipt:
```

```
CUSTOMER ID: CD2020
Date: 13/12/2023
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Garam                          2
+-----------------------------------------------+
Discount: 10.0%
Total Price: RM3.6
Balance: RM1.4
Payment: RM5.0

CUSTOMER ID: CD2020
Date: 02/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Kimball Chili Sauce            1
+-----------------------------------------------+
Discount: 10.0%
Total Price: RM3.6
Balance: RM0.0
Payment: RM3.6


Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
CUSTOMER ID: CD2020
Date: 7/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Glo Dish Wash                   2
+-----------------------------------------------+
Discount: 0.0%
Total Price: RM29.2
Balance: RM0.8000000000000007
Payment: RM30.0

CUSTOMER ID: CD2020
Date: 05/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Glo Dish Wash                   1
+-----------------------------------------------+
Discount: 10.0%
Total Price: RM13.14
Balance: RM6.859999999999999
Payment: RM20.0

CUSTOMER ID: CD2020
Date: 12/12/2023
Receipt:
```

```
============= Inventory ===============
A - Inventory Report
B - Sales Report (by customer)
C - Sales Report (by date)
D - Revenue report
E - Exit
=======================================
Enter your choice [A/B/C/D/E]: c
Enter date [DD/MM/YEAR]:
01/01/2024

Sales Report by Date:

CUSTOMER ID: CD3465
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Ice Cream PaddlePop            1
+-----------------------------------------------+
Chocolate Muffin               1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM5.5
Balance: RM0.5
Payment: RM6.0
```

```
CUSTOMER ID: CD2020
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             1
+-----------------------------------------------+
Aik Cheong White Coffee        1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM17.7
Balance: RM2.3000000000000007
Payment: RM20.0

CUSTOMER ID: CD0938
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Garam                          1
+-----------------------------------------------+
Discount: 0.0%
Total Price: RM2.0
Balance: RM3.0
Payment: RM5.0
```

```
CUSTOMER ID: CD2020
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Wonda White Coffee             1
+-----------------------------------------------+
Aik Cheong White Coffee        1
+-----------------------------------------------+
Discount:0.0%
Total Price: RM17.7
Balance: RM2.3000000000000007
Payment: RM20.0

CUSTOMER ID: CD0938
Date: 01/01/2024
Receipt:
+-----------------------------------------------+
Product Name                  Quantity
+-----------------------------------------------+
Garam                          1
+-----------------------------------------------+
Discount: 0.0%
Total Price: RM2.0
Balance: RM3.0
Payment: RM5.0

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

```
============= Inventory ===============
A - Inventory Report
B - Sales Report (by customer)
C - Sales Report (by date)
D - Revenue report
E - Exit
=======================================
Enter your choice [A/B/C/D/E]: d
Enter month [01/02/03/.../12]:
02

Revenue Report:
Number of units sold: 0
Sales Revenue: RM0.0

Repeat?:
[enter X to return to menu, otherwise enter Y to repeat this process]
```

3.0 - DIFFERENCES AND ADVANTAGES OF JAVA AND C

LANGUAGE	JAVA	C
DIFFERENCES	When compared to C, Java's syntax is complex.	Due to its short syntax, C might produce code that is less structured.
	Java is meant to be simpler to read and write, and its syntax is more like English.	C can be challenging to read and write because of its grammar, which is similar to that of machine language.
	Java is a high-level language that enables platform independence, automated memory management, and a huge standard library.	Low-level programming languages such as C offer direct hardware resource access.
	Java only supports a call by value.	Call by value and call by reference is supported in C.
	Java is an object-oriented programming language, it places more emphasis on data and objects than just logic.	C is a Procedural Oriented language.
ADVANTAGES	Can reuse existing classes and increase their functionality, minimizing repetition in the code base.	With the C programming language, programs compile more quickly.
	Java enables the creation of reusable code and standard programs.	Writing code in one system and using it in another makes the C language incredibly portable.
	Platform independence and high reliability are features of Java.	Because the C language doesn't involve the execution of any complicated commands, debugging in it is simple and useful.
	Java has the ability to construct several threads simultaneously thanks to its multithreading functionality.	C provides dynamic memory allocation it means programmers are free to allocate memory at run time.
	Because Java has good memory management, it is a robust programming language.	C is extendibility. It can continuously add or extend our own functions to C library.

4.0 REFERENCES

Advantages and Disadvantages of C Language - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/advantages-and-disadvantages-of-c-language

Corbo, A. (2022, December 29). *What is java?* Built In. https://builtin.com/software-engineering-perspectives/java#:~:text=Java%20is%20a%20general%2Dpurpose,programming%20languages%20in%20the%20world.

GeeksforGeeks. (2023, February 21). *Difference between Java and C language*. https://www.geeksforgeeks.org/difference-between-java-and-c-language/

Sharma, A. (2023, April 20). *Difference between C and Java Language*. PrepBytes Blog. https://www.prepbytes.com/blog/c-programming/difference-between-c-and-java/#:~:text=The%20main%20difference%20between%20c%20language%20and%20java%20is%20that,and%20a%20large%20standard%20library.

Testbook. (2023, August 8). *Learn what is the Difference between C and Java*. Testbook. https://testbook.com/key-differences/difference-between-c-and-java