Main class:

package J1.S.P0023;

/\*\*

\* This class represents the main class of the Fruit Shop System.

\* It provides options to create fruits, view orders, and perform shopping for buyers.

\*/

public class Main {

/\*\*

\* Initialize manager to use method in class ShoppingManagement

\*/

private static ShoppingMangament manager = new ShoppingMangament();

/\*\*

\* Initialize validation to use method in class FruitShopIO

\*/

private static FruitShopIO validation = new FruitShopIO();

/\*\*

\* The main method of the program. Displays a menu for the user to choose

\* options related to the Fruit Shop System.

\* **@param** args The command-line arguments (not used in this program).

\*/

public static void main(String[] args) {

// Loop until the user chooses to exit

manager.testData();

int choice;

while (true) {

System.out.println("FRUIT SHOP SYSTEM\n1. Create Fruit\n2. View orders\n3. Shopping (for buyer)\n4. Exit\nEnter your choice: ");

choice = validation.checkInputIntLimit(1, 4);

switch (choice) {

case 1:

manager.createFruit();

break;

case 2:

manager.viewOrder();

break;

case 3:

manager.shopping();

break;

case 4:

return;

}

}

}

}

Shopping Management Class:

package J1.S.P0023;

import java.util.ArrayList;

import java.util.Hashtable;

/\*\*

\* This class represents the management system of the fruit shop.

\* It allows users to create fruits, view orders, and perform shopping.

\*/

public class ShoppingMangament {

private static FruitShopIO validation = new FruitShopIO();

ArrayList<Fruit> lf = new ArrayList<>();

Hashtable<String, ArrayList<Order>> ht = new Hashtable<>();

/\*\*

\* Allows users to create new fruits and add them to the shop inventory.

\*/

public void createFruit() {

// Loop until the user decides not to create more fruits

while (true) {

System.out.print("Enter fruit id: ");

String fruitId = validation.checkInputString();

// Check if the ID already exists

if (!validation.checkIdExist(lf, fruitId)) {

System.err.println("Id exists");

return;

}

System.out.print("Enter fruit name: ");

String fruitName = validation.checkInputString();

System.out.print("Enter price: ");

double price = validation.checkInputDouble();

System.out.print("Enter quantity: ");

int quantity = validation.checkInputInt();

System.out.print("Enter origin: ");

String origin = validation.checkInputString();

lf.add(new Fruit(fruitId, fruitName, price, quantity, origin));

// Check if the user wants to continue adding fruits

if (!validation.checkInputYN()) {

return;

}

}

}

/\*\*

\* Displays the orders placed by customers.

\*/

public void viewOrder() {

for (String name : ht.keySet()) {

System.out.println("Customer: " + name);

ArrayList<Order> lo = ht.get(name);

displayListOrder(lo);

}

}

/\*\*

\* Allows customers to buy items from the shop.

\*/

public void shopping() {

// Check if the shop inventory is empty

if (lf.isEmpty()) {

System.err.println("No items available.");

return;

}

// Loop until the customer decides to stop shopping

ArrayList<Order> lo = new ArrayList<>();

while (true) {

displayListFruit();

System.out.print("Enter item: ");

int item = validation.checkInputIntLimit(1, lf.size());

Fruit fruit = getFruitByItem(item);

System.out.print("Enter quantity: ");

int quantity = validation.checkInputIntLimit(1, fruit.getQuantity());

fruit.setQuantity(fruit.getQuantity() - quantity);

// Check if the item is already in the customer's order

if (!validation.checkItemExist(lo, fruit.getFruitId())) {

updateOrder(lo, fruit.getFruitId(), quantity);

} else {

lo.add(new Order(fruit.getFruitId(), fruit.getFruitName(),

quantity, fruit.getPrice()));

}

// Check if the customer wants to continue shopping

if (!validation.checkInputYN()) {

break;

}

}

displayListOrder(lo);

System.out.print("Enter name: ");

String name = validation.checkInputString();

ht.put(name, lo);

System.err.println("Order placed successfully.");

}

/\*\*

\* Displays the list of available fruits in the shop.

\*/

public void displayListFruit() {

int countItem = 1;

System.out.printf("%-10s%-20s%-20s%-15s\n", "Item", "Fruit name", "Origin", "Price");

for (Fruit fruit : lf) {

// Check if the shop has the item in stock

if (fruit.getQuantity() != 0) {

System.out.printf("%-10d%-20s%-20s$%-15.0f\n", countItem++,

fruit.getFruitName(), fruit.getOrigin(), fruit.getPrice());

}

}

}

/\*\*

\* Retrieves the fruit selected by the customer based on the item number.

\*

\* @param item The item number selected by the customer.

\* @return The Fruit object corresponding to the selected item number.

\*/

public Fruit getFruitByItem(int item) {

int countItem = 1;

for (Fruit fruit : lf) {

// Check if the shop has the item in stock

if (fruit.getQuantity() != 0) {

countItem++;

}

if (countItem - 1 == item) {

return fruit;

}

}

return null;

}

/\*\*

\* Displays the list of orders and calculates the total amount.

\*

\* @param lo The list of orders placed by the customer.

\*/

public void displayListOrder(ArrayList<Order> lo) {

double total = 0;

System.out.printf("%-15s%-15s%-16s%-15s\n", "Product", "Quantity", "Price", "Amount");

for (Order order : lo) {

System.out.printf("%-15s%-15d$%-15.0f$%-15.0f\n", order.getFruitName(),

order.getQuantity(), order.getPrice(),

order.getPrice() \* order.getQuantity());

total += order.getPrice() \* order.getQuantity();

}

System.out.println("Total: " + total);

}

/\*\*

\* Updates the order if the same item is ordered again.

\*

\* @param lo The list of orders placed by the customer.

\* @param id The ID of the fruit being ordered.

\* @param quantity The quantity of the fruit being ordered.

\*/

public void updateOrder(ArrayList<Order> lo, String id, int quantity) {

for (Order order : lo) {

if (order.getFruitId().equalsIgnoreCase(id)) {

order.setQuantity(order.getQuantity() + quantity);

return;

}

}

}

/\*\*

\* Adds test data to the shop inventory and orders for demonstration purposes.

\*/

public void testData() {

lf.add(new Fruit("1", "Mango", 20, 5000, "VietNam"));

lf.add(new Fruit("2", "Apple", 12, 5000, "Korean"));

lf.add(new Fruit("3", "Dragon fruit", 15, 9999, "VietNam"));

lf.add(new Fruit("4", "Orange", 5, 9999, "Lao"));

lf.add(new Fruit("5", "Durian", 50, 200, "VietNam"));

ArrayList<Order> lo = new ArrayList<>();

lo.add(new Order("1", "Mango", 10, 20));

lo.add(new Order("2", "Apple", 20, 12));

ht.put("Marie", lo);

}

}

Fruit class:

package J1.S.P0023;

/\*\*

\* This class represents a fruit in the fruit shop inventory.

\*/

public class Fruit {

private String fruitId; // The unique identifier for the fruit

private String fruitName; // The name of the fruit

private double price; // The price of the fruit

private int quantity; // The quantity of the fruit in stock

private String origin; // The origin of the fruit

/\*\*

\* Default constructor for the Fruit class.

\*/

public Fruit() {

}

/\*\*

\* Parameterized constructor for the Fruit class.

\*

\* **@param** fruitId The unique identifier for the fruit.

\* **@param** fruitName The name of the fruit.

\* **@param** price The price of the fruit.

\* **@param** quantity The quantity of the fruit in stock.

\* **@param** origin The origin of the fruit.

\*/

public Fruit(String fruitId, String fruitName, double price, int quantity, String origin) {

this.fruitId = fruitId;

this.fruitName = fruitName;

this.price = price;

this.quantity = quantity;

this.origin = origin;

}

/\*\*

\* Gets the fruit's unique identifier.

\*

\* **@return** The fruit's unique identifier.

\*/

public String getFruitId() {

return fruitId;

}

/\*\*

\* Sets the fruit's unique identifier.

\*

\* **@param** fruitId The unique identifier to set.

\*/

public void setFruitId(String fruitId) {

this.fruitId = fruitId;

}

/\*\*

\* Gets the name of the fruit.

\*

\* **@return** The name of the fruit.

\*/

public String getFruitName() {

return fruitName;

}

/\*\*

\* Sets the name of the fruit.

\*

\* **@param** fruitName The name of the fruit to set.

\*/

public void setFruitName(String fruitName) {

this.fruitName = fruitName;

}

/\*\*

\* Gets the price of the fruit.

\*

\* **@return** The price of the fruit.

\*/

public double getPrice() {

return price;

}

/\*\*

\* Sets the price of the fruit.

\*

\* **@param** price The price to set.

\*/

public void setPrice(double price) {

this.price = price;

}

/\*\*

\* Gets the quantity of the fruit in stock.

\*

\* **@return** The quantity of the fruit in stock.

\*/

public int getQuantity() {

return quantity;

}

/\*\*

\* Sets the quantity of the fruit in stock.

\*

\* **@param** quantity The quantity to set.

\*/

public void setQuantity(int quantity) {

this.quantity = quantity;

}

/\*\*

\* Gets the origin of the fruit.

\*

\* **@return** The origin of the fruit.

\*/

public String getOrigin() {

return origin;

}

/\*\*

\* Sets the origin of the fruit.

\*

\* **@param** origin The origin to set.

\*/

public void setOrigin(String origin) {

this.origin = origin;

}

}

FruitIO Class:

package J1.S.P0023;

import java.util.ArrayList;

import java.util.Scanner;

/\*\*

\* This class handles input/output operations for the Fruit Shop System.

\*/

public class FruitShopIO {

private static final Scanner in = new Scanner(System.in);

/\*\*

\* Validates and retrieves an integer input within a specified range.

\*

\* **@param** min The minimum allowed value.

\* **@param** max The maximum allowed value.

\* **@return** The validated integer input.

\*/

public int checkInputIntLimit(int min, int max) {

// Loop until the user inputs a correct value

while (true) {

try {

int result = Integer.parseInt(in.nextLine().trim());

if (result < min || result > max) {

throw new NumberFormatException();

}

return result;

} catch (NumberFormatException e) {

System.err.println("Please input a number in the range [" + min + ", " + max + "]");

System.out.print("Enter again: ");

}

}

}

/\*\*

\* Validates and retrieves a non-empty string input.

\*

\* **@return** The validated non-empty string input.

\*/

public String checkInputString() {

// Loop until the user inputs a correct value

while (true) {

String result = in.nextLine().trim();

if (result.isEmpty()) {

System.err.println("Input cannot be empty");

System.out.print("Enter again: ");

} else {

return result;

}

}

}

/\*\*

\* Validates and retrieves an integer input.

\*

\* **@return** The validated integer input.

\*/

public int checkInputInt() {

// Loop until the user inputs a correct value

while (true) {

try {

int result = Integer.parseInt(in.nextLine().trim());

return result;

} catch (NumberFormatException e) {

System.err.println("Input must be an integer.");

System.out.print("Enter again: ");

}

}

}

/\*\*

\* Validates and retrieves a double input.

\*

\* **@return** The validated double input.

\*/

public double checkInputDouble() {

// Loop until the user inputs a correct value

while (true) {

try {

double result = Double.parseDouble(in.nextLine());

return result;

} catch (NumberFormatException e) {

System.err.println("Input must be a double");

System.out.print("Enter again: ");

}

}

}

/\*\*

\* Validates and retrieves a yes/no input.

\*

\* **@return** True if the user inputs 'Y' or 'y', false if the user inputs 'N' or 'n'.

\*/

public boolean checkInputYN() {

System.out.print("Do you want to continue (Y/N)? ");

// Loop until the user inputs a correct value

while (true) {

String result = checkInputString();

// Return true if the user inputs 'y' or 'Y'

if (result.equalsIgnoreCase("Y")) {

return true;

}

// Return false if the user inputs 'n' or 'N'

if (result.equalsIgnoreCase("N")) {

return false;

}

System.err.println("Please input 'y' or 'Y' for Yes, 'n' or 'N' for No.");

System.out.print("Enter again: ");

}

}

/\*\*

\* Checks if a given ID exists in the list of fruits.

\*

\* **@param** lf The list of fruits.

\* **@param** id The ID to check.

\* **@return** True if the ID does not exist in the list, false otherwise.

\*/

public boolean checkIdExist(ArrayList<Fruit> lf, String id) {

for (Fruit fruit : lf) {

if (id.equalsIgnoreCase(fruit.getFruitId())) {

return false;

}

}

return true;

}

/\*\*

\* Checks if a given item ID exists in the list of orders.

\*

\* **@param** lo The list of orders.

\* **@param** id The ID to check.

\* **@return** True if the item ID does not exist in the list, false otherwise.

\*/

public boolean checkItemExist(ArrayList<Order> lo, String id) {

for (Order order : lo) {

if (order.getFruitId().equalsIgnoreCase(id)) {

return false;

}

}

return true;

}

}

\

Order class:

package J1.S.P0023;

/\*\*

\* This class represents an order for a specific fruit in the Fruit Shop System.

\*/

public class Order {

private String fruitId; // The ID of the ordered fruit

private String fruitName; // The name of the ordered fruit

private int quantity; // The quantity of the ordered fruit

private double price; // The price of the ordered fruit

/\*\*

\* Default constructor for the Order class.

\*/

public Order() {

}

/\*\*

\* Parameterized constructor for the Order class.

\*

\* **@param** fruitId The ID of the ordered fruit.

\* **@param** fruitName The name of the ordered fruit.

\* **@param** quantity The quantity of the ordered fruit.

\* **@param** price The price of the ordered fruit.

\*/

public Order(String fruitId, String fruitName, int quantity, double price) {

this.fruitId = fruitId;

this.fruitName = fruitName;

this.quantity = quantity;

this.price = price;

}

/\*\*

\* Gets the ID of the ordered fruit.

\*

\* **@return** The ID of the ordered fruit.

\*/

public String getFruitId() {

return fruitId;

}

/\*\*

\* Sets the ID of the ordered fruit.

\*

\* **@param** fruitId The ID to set.

\*/

public void setFruitId(String fruitId) {

this.fruitId = fruitId;

}

/\*\*

\* Gets the name of the ordered fruit.

\*

\* **@return** The name of the ordered fruit.

\*/

public String getFruitName() {

return fruitName;

}

/\*\*

\* Sets the name of the ordered fruit.

\*

\* **@param** fruitName The name to set.

\*/

public void setFruitName(String fruitName) {

this.fruitName = fruitName;

}

/\*\*

\* Gets the quantity of the ordered fruit.

\*

\* **@return** The quantity of the ordered fruit.

\*/

public int getQuantity() {

return quantity;

}

/\*\*

\* Sets the quantity of the ordered fruit.

\*

\* **@param** quantity The quantity to set.

\*/

public void setQuantity(int quantity) {

this.quantity = quantity;

}

/\*\*

\* Gets the price of the ordered fruit.

\*

\* **@return** The price of the ordered fruit.

\*/

public double getPrice() {

return price;

}

/\*\*

\* Sets the price of the ordered fruit.

\*

\* **@param** price The price to set.

\*/

public void setPrice(double price) {

this.price = price;

}

}