import java.util.ArrayList;

import java.util.Scanner;

class Product {

private String productName;

private String productId;

private double price;

private int stock;

private double costPrice;

private double totalProfit;

// Constructor

public Product(String productName, String productId, double price, int stock, double

costPrice) {

this.productName = productName;

this.productId = productId;

this.price = validatePrice(price);

this.stock = validateStock(stock);

this.costPrice = validatePrice(costPrice);

this.totalProfit = 0;

}

// Validate price

private double validatePrice(double price) {

if (price &lt; 0) {

throw new IllegalArgumentException(&quot;Price cannot be negative!&quot;);

}

return price;

}

// Validate stock

private int validateStock(int stock) {

if (stock &lt; 0) {

throw new IllegalArgumentException(&quot;Stock cannot be negative!&quot;);

}

return stock;

}

// Display product information

public void displayProductInfo() {

System.out.println(&quot;Product Name: &quot; + productName);

System.out.println(&quot;Product ID: &quot; + productId);

System.out.println(&quot;Price: &quot; + price);

System.out.println(&quot;Stock: &quot; + stock);

System.out.println(&quot;Cost Price: &quot; + costPrice);

}

// Getters

public String getProductId() {

return productId;

}

public int getStock() {

return stock;

}

public double getPrice() {

return price;

}

public double getCostPrice() {

return costPrice;

}

// Update stock after a purchase

public void purchase(int quantity) {

if (quantity &lt;= 0) {

System.out.println(&quot;Invalid quantity! Purchase quantity should be greater than

zero.&quot;);

} else if (quantity &gt; stock) {

System.out.println(&quot;Not enough stock available!&quot;);

} else {

stock -= quantity;

double profit = (price - costPrice) \* quantity;

totalProfit += profit;

System.out.println(&quot;Purchase successful! Remaining stock: &quot; + stock);

System.out.println(&quot;Profit from this transaction: &quot; + profit);

}

}

// Shipping (adding stock)

public void ship(int quantity) {

if (quantity &lt;= 0) {

System.out.println(&quot;Invalid shipping quantity! It must be greater than zero.&quot;);

} else {

stock += quantity;

System.out.println(&quot;Stock updated! New stock: &quot; + stock);

}

}

// Display total profit for this product

public double getTotalProfit() {

return totalProfit;

}

}

public class InventoryManagementSystem {

private static ArrayList&lt;Product&gt; productList = new ArrayList&lt;&gt;();

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

public static void main(String[] args) {

// Adding some products to the inventory

try {

productList.add(new Product(&quot;Laptop&quot;, &quot;P1001&quot;, 1500.00, 50, 1200.00));

productList.add(new Product(&quot;Smartphone&quot;, &quot;P1002&quot;, 800.00, 100, 600.00));

productList.add(new Product(&quot;Headphones&quot;, &quot;P1003&quot;, 150.00, 200, 100.00));

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

while (true) {

System.out.println(&quot;\nInventory Management Menu:&quot;);

System.out.println(&quot;1. List of all products&quot;);

System.out.println(&quot;2. Display individual product information&quot;);

System.out.println(&quot;3. Purchase&quot;);

System.out.println(&quot;4. Shipping&quot;);

System.out.println(&quot;5. Check stock balance&quot;);

System.out.println(&quot;6. Profit and Loss calculation&quot;);

System.out.println(&quot;7. Exit&quot;);

System.out.print(&quot;Choose an option: &quot;);

int option = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (option) {

case 1:

listAllProducts();

break;

case 2:

displayProductInfo();

break;

case 3:

makePurchase();

break;

case 4:

handleShipping();

break;

case 5:

checkStockBalance();

break;

case 6:

calculateProfitAndLoss();

break;

case 7:

System.out.println(&quot;Exiting the system. Goodbye!&quot;);

return;

default:

System.out.println(&quot;Invalid option! Please choose a valid option.&quot;);

}

}

}

// List all products in the inventory

private static void listAllProducts() {

System.out.println(&quot;\nProduct List:&quot;);

for (Product product : productList) {

System.out.println(&quot;Product Name: &quot; + product.getProductId() + &quot; - &quot; +

product.getProductId());

}

}

// Display individual product information

private static void displayProductInfo() {

System.out.print(&quot;Enter Product ID: &quot;);

String productId = scanner.nextLine();

Product product = findProductById(productId);

if (product != null) {

product.displayProductInfo();

} else {

System.out.println(&quot;Product not found!&quot;);

}

}

// Handle purchase

private static void makePurchase() {

System.out.print(&quot;Enter Product ID for purchase: &quot;);

String productId = scanner.nextLine();

Product product = findProductById(productId);

if (product != null) {

System.out.print(&quot;Enter quantity to purchase: &quot;);

int quantity = scanner.nextInt();

if (quantity &gt; 0) {

product.purchase(quantity);

} else {

System.out.println(&quot;Invalid quantity! Quantity should be greater than zero.&quot;);

}

} else {

System.out.println(&quot;Product not found!&quot;);

}

}

// Handle shipping (add stock)

private static void handleShipping() {

System.out.print(&quot;Enter Product ID for shipping: &quot;);

String productId = scanner.nextLine();

Product product = findProductById(productId);

if (product != null) {

System.out.print(&quot;Enter quantity to add to stock: &quot;);

int quantity = scanner.nextInt();

if (quantity &gt; 0) {

product.ship(quantity);

} else {

System.out.println(&quot;Invalid quantity! Quantity should be greater than zero.&quot;);

}

} else {

System.out.println(&quot;Product not found!&quot;);

}

}

// Check stock balance

private static void checkStockBalance() {

System.out.print(&quot;Enter Product ID to check stock: &quot;);

String productId = scanner.nextLine();

Product product = findProductById(productId);

if (product != null) {

System.out.println(&quot;Current stock for product &quot; + productId + &quot; is: &quot; +

product.getStock());

} else {

System.out.println(&quot;Product not found!&quot;);

}

}

// Calculate total profit and loss

private static void calculateProfitAndLoss() {

double totalProfit = 0;

for (Product product : productList) {

totalProfit += product.getTotalProfit();

}

if (totalProfit &gt;= 0) {

System.out.println(&quot;Total profit: $&quot; + totalProfit);

} else {

System.out.println(&quot;Total loss: $&quot; + Math.abs(totalProfit));

}

}

// Find product by ID

private static Product findProductById(String productId) {

for (Product product : productList) {

if (product.getProductId().equalsIgnoreCase(productId)) {

return product;

}

}

return null;

}

}