// Mini Project: E-Commerce Order Management System in Java

import java.util.\*;

// Product class

class Product {

private String id;

private String name;

private double price;

private int stock;

public Product(String id, String name, double price, int stock) {

this.id = id;

this.name = name;

this.price = price;

this.stock = stock;

}

public String getId() { return id; }

public String getName() { return name; }

public double getPrice() { return price; }

public int getStock() { return stock; }

public void reduceStock(int quantity) {

if (quantity <= stock) {

stock -= quantity;

}

}

@Override

public String toString() {

return name + " ($" + price + ")";

}

}

// Cart class

class Cart {

private Map<Product, Integer> items = new HashMap<>();

public void addProduct(Product product, int quantity) {

items.put(product, items.getOrDefault(product, 0) + quantity);

}

public void removeProduct(Product product) {

items.remove(product);

}

public double calculateTotal() {

return items.entrySet().stream()

.mapToDouble(e -> e.getKey().getPrice() \* e.getValue())

.sum();

}

public Map<Product, Integer> getItems() {

return items;

}

public void clear() {

items.clear();

}

}

// Order class

class Order {

private String orderId;

private Map<Product, Integer> orderedItems;

private double totalAmount;

private String status;

public Order(String orderId, Map<Product, Integer> items, double total) {

this.orderId = orderId;

this.orderedItems = new HashMap<>(items);

this.totalAmount = total;

this.status = "Pending";

}

public void updateStatus(String newStatus) {

this.status = newStatus;

}

public String toString() {

return "Order ID: " + orderId + ", Total: $" + totalAmount + ", Status: " + status;

}

}

// PaymentService interface

interface PaymentService {

boolean processPayment(double amount);

}

// Dummy implementation

class DummyPaymentService implements PaymentService {

public boolean processPayment(double amount) {

return new Random().nextBoolean(); // randomly simulate payment success/failure

}

}

// User class

class User {

private String userId;

private String name;

private Cart cart = new Cart();

private List<Order> orderHistory = new ArrayList<>();

public User(String userId, String name) {

this.userId = userId;

this.name = name;

}

public Cart getCart() { return cart; }

public void placeOrder(PaymentService paymentService) {

double total = cart.calculateTotal();

if (paymentService.processPayment(total)) {

Order order = new Order(UUID.randomUUID().toString(), cart.getItems(), total);

orderHistory.add(order);

System.out.println("Payment successful. " + order);

cart.clear();

} else {

System.out.println("Payment failed. Please try again.");

}

}

}

// Main class

public class ECommerceApp {

public static void main(String[] args) {

// Sample products

Product p1 = new Product("P101", "Smartphone", 699.99, 10);

Product p2 = new Product("P102", "Laptop", 1299.99, 5);

// User

User user = new User("U001", "Alice");

// Add products to cart

user.getCart().addProduct(p1, 1);

user.getCart().addProduct(p2, 1);

// Show cart total

System.out.println("Cart Total: $" + user.getCart().calculateTotal());

// Place order

PaymentService paymentService = new DummyPaymentService();

user.placeOrder(paymentService);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Mini Project: E-Commerce Order Management System in Java with Admin Panel

import java.util.\*;

// Product class

class Product {

private String id;

private String name;

private double price;

private int stock;

public Product(String id, String name, double price, int stock) {

this.id = id;

this.name = name;

this.price = price;

this.stock = stock;

}

public String getId() { return id; }

public String getName() { return name; }

public double getPrice() { return price; }

public int getStock() { return stock; }

public void reduceStock(int quantity) {

if (quantity <= stock) {

stock -= quantity;

}

}

public void increaseStock(int quantity) {

stock += quantity;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public String toString() {

return "Product ID: " + id + ", Name: " + name + ", Price: $" + price + ", Stock: " + stock;

}

}

// Cart class

class Cart {

private Map<Product, Integer> items = new HashMap<>();

public void addProduct(Product product, int quantity) {

items.put(product, items.getOrDefault(product, 0) + quantity);

}

public void removeProduct(Product product) {

items.remove(product);

}

public double calculateTotal() {

return items.entrySet().stream()

.mapToDouble(e -> e.getKey().getPrice() \* e.getValue())

.sum();

}

public Map<Product, Integer> getItems() {

return items;

}

public void clear() {

items.clear();

}

}

// Order class

class Order {

private String orderId;

private Map<Product, Integer> orderedItems;

private double totalAmount;

private String status;

public Order(String orderId, Map<Product, Integer> items, double total) {

this.orderId = orderId;

this.orderedItems = new HashMap<>(items);

this.totalAmount = total;

this.status = "Pending";

}

public void updateStatus(String newStatus) {

this.status = newStatus;

}

public String toString() {

return "Order ID: " + orderId + ", Total: $" + totalAmount + ", Status: " + status;

}

}

// PaymentService interface

interface PaymentService {

boolean processPayment(double amount);

}

// Dummy implementation

class DummyPaymentService implements PaymentService {

public boolean processPayment(double amount) {

return new Random().nextBoolean(); // randomly simulate payment success/failure

}

}

// User class

class User {

private String userId;

private String name;

private Cart cart = new Cart();

private List<Order> orderHistory = new ArrayList<>();

public User(String userId, String name) {

this.userId = userId;

this.name = name;

}

public Cart getCart() { return cart; }

public void placeOrder(PaymentService paymentService) {

double total = cart.calculateTotal();

if (paymentService.processPayment(total)) {

Order order = new Order(UUID.randomUUID().toString(), cart.getItems(), total);

orderHistory.add(order);

System.out.println("Payment successful. " + order);

cart.clear();

} else {

System.out.println("Payment failed. Please try again.");

}

}

}

// Admin class to manage products

class Admin {

private List<Product> inventory;

public Admin(List<Product> inventory) {

this.inventory = inventory;

}

public void addProduct(String id, String name, double price, int stock) {

Product newProduct = new Product(id, name, price, stock);

inventory.add(newProduct);

System.out.println("Product added: " + newProduct);

}

public void removeProduct(String productId) {

inventory.removeIf(p -> p.getId().equals(productId));

System.out.println("Product with ID " + productId + " removed.");

}

public void updateProduct(String productId, double newPrice, int additionalStock) {

for (Product p : inventory) {

if (p.getId().equals(productId)) {

p.setPrice(newPrice);

p.increaseStock(additionalStock);

System.out.println("Product updated: " + p);

return;

}

}

System.out.println("Product with ID " + productId + " not found.");

}

public void listAllProducts() {

System.out.println("--- Product Catalog ---");

for (Product p : inventory) {

System.out.println(p);

}

}

}

// Main class

public class ECommerceApp {

public static void main(String[] args) {

// Sample inventory list

List<Product> inventory = new ArrayList<>();

Admin admin = new Admin(inventory);

// Admin adds products

admin.addProduct("P101", "Smartphone", 699.99, 10);

admin.addProduct("P102", "Laptop", 1299.99, 5);

admin.addProduct("P103", "Headphones", 99.99, 25);

admin.listAllProducts();

// User actions

User user = new User("U001", "Alice");

Product p1 = inventory.get(0);

Product p2 = inventory.get(1);

user.getCart().addProduct(p1, 1);

user.getCart().addProduct(p2, 1);

System.out.println("\nCart Total: $" + user.getCart().calculateTotal());

PaymentService paymentService = new DummyPaymentService();

user.placeOrder(paymentService);

}

}