ACADEMIA DE STUDII ECONOMICE BUCUREŞTI FACULTATEA DE CIBERNETICĂ STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ



Gestiunea unui magazin online

**Profesor Coordonator:Ciurea Cristian**   **Student: Spiridon Gabriela-Elena**

## **Cuprins**

[**1. INTRODUCERE**](#) **2**

[1.1 Scopul Proiectului](#) 2

[1.2 Arhitectura POO pe Straturi](#) 2

[**2. DETALIEREA CLASELOR ȘI A PRINCIPIILOR POO**](#) **3**

[2.1 Stratul Model (model package)](#) 3

[2.2 Stratul Persistenței (repository package)](#) 4

[2.3 Stratul Logicii de Business (service package)](#) 5

[**3. IMPLEMENTAREA CERINȚELOR SPECIFICE**](#) **6**

[3.1 Cerința 1 & 2: Persistența Datelor (Fișiere Text)](#) 6

[3.2 Cerința 5: Gestiunea Excepțiilor și Validarea](#) 6

[3.3 Cerința 6 & 7: Interfața Grafică și Fluxul Logic](#_6zpbw6k4jfip) 7

# 

## **1. INTRODUCERE**

### **1.1 Scopul Proiectului**

Scopul proiectului este realizarea unei aplicații Java care să simuleze sistemul de gestiune al unui magazin online (e-commerce). Aplicația implementează operațiile fundamentale **CRUD** și logica de tranzacție (plasare comandă, actualizare stoc), fără a utiliza o bază de date, ci bazându-se pe **persistența datelor în fișiere text**.

### **1.2 Arhitectura POO pe Straturi**

Aplicația este structurată conform principiilor **Programării Orientate pe Obiect (POO)** într-o Arhitectură pe Trei Straturi (Model-Service-Repository), asigurând o separare clară a responsabilităților:

| **Pachet** | **Strat** | **Responsabilitate POO** |
| --- | --- | --- |
| **model** | Entități | Datele aplicatiei (Încapsulare, Moștenire). |
| **repository** | Persistență | Stocarea/Restaurarea datelor (Abstractizare, Polimorfism). |
| **service** | Logică Business | Regulile, Validările, Tranzacțiile (Cerinta 5). |
| **ui** | Prezentare | Interfața Grafică (GUI) și interacțiunea cu utilizatorul (Cerinta 7). |
| **app** | Inițializare | Punctul de pornire (gestiunea dependențelor). |

## 

## **2. DETALIEREA CLASELOR ȘI A PRINCIPIILOR POO**

### **2.1 Stratul Model (model package)**

Acest strat definește entitățile de business, aplicând **Încapsularea** și **Moștenirea**.

| **Clasa** | **Relație** | **Detalii Implementare** |
| --- | --- | --- |
| **User** | Baza | Clasa de bază. Conține atributele comune (id, name, email, password) și implementează Serializable. |
| **Client** | Moștenire | **Extinde User**. Adaugă atribute specifice (deliveryAddress, phoneNumber) și colecția List<Order> (Cerinta 3). |
| **Product** | Entitate | Atribute pentru stoc (stockQuantity), preț, nume, și ProductType (enum). |
| **Order** | Entitate | Conține Map<Product, Integer> (produse comandate și cantitatea - Cerinta 3), clientId, totalAmount și OrderStatus (enum). |
| **ProductType/OrderStatus** | Enum | Definirea constantelor pentru tipul de produs și starea comenzii. |

### **2.2 Stratul Persistenței (repository package)**

Acest strat gestionează stocarea datelor în fișiere text (Cerinta 1, 2, 4) și aplică **Polimorfismul** prin IRepository.

| **Clasa/Interfață** | **Principiu POO** | **Descriere** |
| --- | --- | --- |
| **IRepository<T, ID>** | **Polimorfism/Abstractizare** | Interfață generică ce definește contractul CRUD (save, findById, findAll, delete) și metodele de persistență (loadAllData, saveAllData). |
| **Repository (Client/Product/Order)\*** | Implementare | Clasă concretă care **implementează IRepository**. Utilizează **Map<ID, Entitate>** (Cerinta 3) pentru stocarea în memorie (Cerinta 4) și implementează logica I/O pentru citirea/scrierea în fișiere text. |
| **DataProcessingException** | Excepție Custom | Excepție netratată (RuntimeException) ridicată la erori de I/O sau de parsare a fișierelor (Cerinta 5). |

### **2.3 Stratul Logicii de Business (service package)**

Acest strat conține regulile și logica complexă a aplicației.

| **Clasa** | **Logica de Business / Rol** |
| --- | --- |
| **ServiceProduct** | Validarea prețului/stocului (Cerinta 5). Scăderea stocului (decreaseStock). Calculul valorii totale a stocului (Raport Cerinta 1). |
| **ServiceClient** | Validarea formatului email/parolă (Cerinta 5). Logica de Autentificare (authenticate) și de înregistrare. |
| **ServiceOrder** | **Logica Tranzacției:** Calculează totalul, validează stocul și coordonează modificarea stocului prin apelarea ServiceProduct (Colaborare Service-uri). |
| **InvalidDataException** | Excepție Custom |

## **3. IMPLEMENTAREA CERINȚELOR SPECIFICE**

### **3.1 Cerința 1 & 2: Persistența Datelor (Fișiere Text)**

Aplicația utilizează fișiere text separate (products.txt, clients.txt, orders.txt) în directorul data/.

* **Salvare/Restaurare (C.2):** Toate Repositories utilizează loadAllData() în constructor pentru restaurare și saveAllData() (apelată la shutdown) pentru salvarea datelor din memoria RAM în fișiere.
* **Rapoarte (C.1):** Metoda ServiceOrder.getUnitsSoldPerProduct() iterează prin comenzile din memorie (restaurate din fișier) pentru a genera statistica detaliată a unităților vândute pe produs, afișată în GUI.
* **Formatare:** S-a utilizat DateTimeFormatter.ISO\_LOCAL\_DATE\_TIME pentru a asigura serializarea și deserializarea sigură a obiectelor LocalDateTime (data comenzii).

### **3.2 Cerința 5: Gestiunea Excepțiilor și Validarea**

S-au definit două clase de excepții custom (necesare pentru C.5):

* **InvalidDataException:** Aruncată de Service-uri când datele introduse de utilizator (ex: în GUI) nu respectă regulile (ex: Pret <= 0, Parola < 6 caractere).
* **DataProcessingException:** Aruncată de Repositories când apar erori de I/O sau datele din fișier sunt corupte/incomplete.

### **3.3 Cerința 6 & 7: Interfața Grafică și Fluxul Logic**

Fluxul logic este gestionat de clasa Main și StoreGUI:

* **GUI (C.7):** Aplicația utilizează **Java Swing** și JTabbedPane pentru o organizare vizuală (Client Access, Products, Orders, Reports).
* **Pornire Securizată:** La pornire, aplicația forțează afișarea dialogului de Login/Register. Fereastra principală devine vizibilă doar după autentificarea cu succes (gestionată de initializeLoginFlow()).
* **Tranzacție (Place Order):** Logica de plasare a comenzii este complexă (adăugare în coș, validare stoc, apel ServiceOrder.placeOrder()) și este demonstrată interactiv prin GUI.