LABORATOR 3

2.

import java.io.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

class Produs {

private String denumire;

private double pret;

private int cantitate;

private LocalDate dataExpirare;

public Produs(String denumire, double pret, int cantitate, LocalDate dataExpirare) {

this.denumire = denumire;

this.pret = pret;

this.cantitate = cantitate;

this.dataExpirare = dataExpirare;

}

public String getDenumire() {

return denumire;

}

public void setDenumire(String denumire) {

this.denumire = denumire;

}

public double getPret() {

return pret;

}

public void setPret(double pret) {

this.pret = pret;

}

public int getCantitate() {

return cantitate;

}

public void setCantitate(int cantitate) {

this.cantitate = cantitate;

}

public LocalDate getDataExpirare() {

return dataExpirare;

}

public void setDataExpirare(LocalDate dataExpirare) {

this.dataExpirare = dataExpirare;

}

@Override

public String toString() {

return String.format("%s, %.2f, %d, %s", denumire, pret, cantitate, dataExpirare);

}

}

public class Main {

private static List<Produs> listaProduse = new ArrayList<>();

private static double incasariTotale = 0;

public static void main(String[] args) {

incarcaProduse("produse.csv");

while (true) {

afiseazaMeniu();

Scanner scanner = new Scanner(System.in);

int optiune = scanner.nextInt();

scanner.nextLine();

switch (optiune) {

case 1:

afiseazaToateProdusele();

break;

case 2:

afiseazaProduseleExpirate();

break;

case 3:

vindeProdus();

break;

case 4:

afiseazaProduseCuPretMinim();

break;

case 5:

salveazaProduseSubCantitate();

break;

case 6:

System.out.println("La revedere!");

System.exit(0);

default:

System.out.println("Opțiune invalidă. Încercați din nou.");

}

}

}

private static void afiseazaMeniu() {

System.out.println("\n-- Meniu --");

System.out.println("1. Afișează toate produsele");

System.out.println("2. Afișează produsele expirate");

System.out.println("3. Vinde produs");

System.out.println("4. Afișează produsele cu prețul minim");

System.out.println("5. Salvează produsele cu cantitate sub o valoare");

System.out.println("6. Ieșire");

System.out.print("Selectați o opțiune: ");

}

private static void incarcaProduse(String numeFisier) {

try (BufferedReader br = new BufferedReader(new FileReader(numeFisier))) {

String linie;

while ((linie = br.readLine()) != null) {

String[] elemente = linie.split(",");

String denumire = elemente[0].trim();

double pret = Double.parseDouble(elemente[1].trim());

int cantitate = Integer.parseInt(elemente[2].trim());

LocalDate dataExpirare = LocalDate.parse(elemente[3].trim());

Produs produs = new Produs(denumire, pret, cantitate, dataExpirare);

listaProduse.add(produs);

}

} catch (IOException e) {

System.err.println("Eroare la citirea din fisier: " + e.getMessage());

}

}

private static void afiseazaToateProdusele() {

System.out.println("\n-- Toate produsele --");

for (Produs produs : listaProduse) {

System.out.println(produs);

}

}

private static void afiseazaProduseleExpirate() {

LocalDate dataCurenta = LocalDate.now();

System.out.println("\n-- Produse expirate --");

for (Produs produs : listaProduse) {

if (produs.getDataExpirare().isBefore(dataCurenta)) {

System.out.println(produs);

}

}

}

private static void vindeProdus() {

Scanner scanner = new Scanner(System.in);

System.out.print("Introduceți denumirea produsului pentru vânzare: ");

String denumireProdus = scanner.nextLine();

for (Iterator<Produs> iterator = listaProduse.iterator(); iterator.hasNext(); ) {

Produs produs = iterator.next();

if (produs.getDenumire().equalsIgnoreCase(denumireProdus)) {

System.out.print("Introduceți cantitatea dorită: ");

int cantitateVanzare = scanner.nextInt();

if (cantitateVanzare > produs.getCantitate()) {

System.out.println("Cantitate insuficientă pe stoc.");

} else {

incasariTotale += cantitateVanzare \* produs.getPret();

produs.setCantitate(produs.getCantitate() - cantitateVanzare);

if (produs.getCantitate() == 0) {

iterator.remove();

}

System.out.println("Vânzare realizată cu succes!");

}

return; // Ieșim din metoda vindeProdus după vânzare

}

}

System.out.println("Produsul nu a fost găsit.");

}

private static void afiseazaProduseCuPretMinim() {

if (listaProduse.isEmpty()) {

System.out.println("Nu există produse în magazin.");

return;

}

double pretMinim = listaProduse.get(0).getPret();

for (Produs produs : listaProduse) {

if (produs.getPret() < pretMinim) {

pretMinim = produs.getPret();

}

}

System.out.println("\n-- Produse cu preț minim --");

for (Produs produs : listaProduse) {

if (produs.getPret() == pretMinim) {

System.out.println(produs);

}

}

}

private static void salveazaProduseSubCantitate() {

Scanner scanner = new Scanner(System.in);

System.out.print("Introduceți cantitatea minimă dorită: ");

int cantitateMinima = scanner.nextInt();

try (PrintWriter writer = new PrintWriter(new FileWriter("produse\_sub\_cantitate.csv"))) {

for (Produs produs : listaProduse) {

if (produs.getCantitate() < cantitateMinima) {

writer.println(produs);

}

}

System.out.println("Produsele cu cantitate sub " + cantitateMinima + " au fost salvate în 'produse\_sub\_cantitate.csv'");

} catch (IOException e) {

System.err.println("Eroare la scrierea în fișier: " + e.getMessage());

}

}

}