Țuluca Maria Laura – Grupa 322AA

Proiect POO – administrarea unui magazin ROCK the SHOP

În această aplicație din limbajul C++, m-am ocupat de implementarea si gestionarea unui magazin **Rock the Shop**, care comercializează produse din industria muzicală rock. Scopul este să automatizeze procesul de administrare a magazinului, oferind un meniu intuitiv pentru navigarea între acțiunile de gestionare a angajaților, stocurilor, comenzilor și rapoartelor. În implementare, am considerat că toate comenzile sunt procesate și înregistrate în luna curentă, iar salariile și rapoartele sunt calculate și create aferent.

Clasa Angajat.h și implementarea metodelor în Angajat.cpp: clasa are două atribute statice, counter(pentru numărul total de angajați) si id_counter(pentru id-uri unice); fiecare angajat are un ID unic(int) asignat la „angajare”, are un tip_angajat(string) care determină faptul că este Manager, Operator sau Asistent, un nume, prenume și CNP(toate de tip string), o dată de angajare(struct tm), un salariu(double) calculat prin metodă proprie. Ulterior implementării și testelor am mai adăugat atributele nr_comenzi(int) și val_comenzi(double), dar și metodele aferente pentru calcularea datelor necesare raporturilor din clasa Magazin. *Metodele* implementate conțin constructorul fără parametrii(inițializează un angajat cu valori implicite), constructorul cu parametrii(inițializează angajați cu valori furnizate), o metodă de validare a CNP-ului(verifică lungimea, corectitudinea cifrei de control), o metodă care calculează salariul(luând în calcul ziua de naștere, tipul de angajat, comenzile procesate), o metodă de resetare a counter-ului static în cazul unei demisii, o metodă de modificare a numelui, mai multe get-uri pentru returnarea datelor și operatorii << și >> pentru aceasta clasă.

Clasa de bază Produs.h și metodele implementate în Produs.cpp: atribut static cod_counter pentru asignarea unică de cod, atribute pentru cod(int), denumire(string), stoc(int) și preț de bază(double). Metodele implementate conțin constructor cu și fără parametrii, destructor virtual, metode virtuale calcul preț final(cu costuri de livrare), de afișare a datelor și de returnare a tipului special, metode statice de modificare stoc și ștergere produs după cod, metode de modificare stoc și preț bază, dar și multiple get-uri pentru aflare date.

Clasele derivate din clasa Produs.h – 1. ArticolVestimentar.h: atribute specifice de culoare și marcă, ambele string-uri, plus atributele moștenite de la clasa de bază. Metodele implementate conțin constructor cu parametrii și metodele supraîncărcate din Produs.h, unde tip special se referă la culoare.
2. Disc.h și DiscVintage.h(care este derivată din Disc.h): ambele au atributele moștenite de la clasa Produs, atributele specifice de tip string: casa de discuri, data de lansare, trupa, album și tipul discului, dar și metodele virtuale supraîncărcate din clasa Produs; pe lângă acestea, clasa DiscVintage mai are și atributele mint(bool) și coeficientRaritate(int), dar are în minus get-ul pentru tipul special(adică tip disc).

Clasa Comanda.h și metodele implementate în Comanda.cpp: fiecare comandă are asignat un ID determinat la înregistrare(tot printr-un id_counter static), o dată de înregistrare comandă(struct tm), o durată de soluționare(int), o valoare brută și valoare totală(double)

calculată cu costuri de livrare și împachetare și un vector de tuples de produse. Vectorul de tuples conține triplete de string, string, int cu informații despre denumire, cantitate și tip special de produs. Metodele implementate conțin constructorul cu parametrii, get-uri speciale pentru date, o metodă de calcul a datei de finalizare a unei comenzi și operatorul << pentru afișarea datelor comenzii.

Clasa Operator.h: am implementat clasa aceasta pentru gestionarea mai ușoară a procesării de comenzi; clasa are atributele id și un pointer la vector de comenzi (ca să-l aloc dinamic atunci când îl folosesc în set-ul din procesarea comenzilor), pe care le primește în timpul procesării comenzilor. Metodele implementate direct în Operator.h sunt considerate inline și conțin constructorul cu parametrii, get-uri pentru date, operatorul < necesar la ordonarea în set după id-ul operatorului, metoda de adăugare comandă în vector, operatorul == și operatorul !=.

Clasa Magazin.h și metodele implementate în Magazin.cpp: fiecare magazin are ca atribute un vector de adrese de angajati, un vector de adrese de produse și o coadă de comenzi, toate datele fiind citite din fișiere cu metode din clasa Magazin. Metodele implementate conțin constructorul cu și fără parametrii, constructorul de copiere și operatorul =, constructorii de copiere si operatorul = de mutare care nu se mai generau automat și destructorul. Mai mult, am implementat și metode pentru citire și creare a atributelor specifice magazinului, în care se verifică automat dacă datele corespund cu ce îmi trebuie mie(CNP-uri valide, angajați majori, comenzi valide, produse de tipuri valabile).

Metoda privată de *validare comandă* verifică numărul de produse dintr-o comandă, stocurile disponibile în raport cu cantitățile din comenzi și valoarea brută a comenzii. Pentru calcularea *valorii brute și totale* ale comenzilor am implementat metode private în clasa Magazin, ce compară produsele din comandă cu produsele din vectorul de produse. Tot ca metode private am implementat și calcularea *datei de finalizare* a unei comenzi în funcție de durată(pentru a putea crea o comandă la citirea din fișier), metoda de *transformare a unui string în tm*(folosit la crearea unui angajat), metodele de *modificari_stocuri* și *valori_raport* ce creează modificările necesare asupra angajaților și produselor în timpul procesării de comenzi.

Metoda *procesarea_comenzilor* gestionează simularea procesării comenzilor dintr-un magazin. Aceasta utilizează o coadă (*queue*) pentru comenzile neprocesate și un *set* pentru operatorii disponibili, organizați după ID-uri. Fiecărui operator îi este atribuit un vector de comenzi. Metoda parcurge comenzile și, în funcție de disponibilitatea operatorilor, fie le atribuie operatorului cu cele mai puține comenzi (limitând la un maxim de 3 comenzi), fie le amână pentru procesare ulterioară. Amânarea implică actualizarea datei comenzii la ziua următoare(nemodificând durata de procesare) și restructurarea cozii de comenzi cu ajutorul unei cozi temporare pentru a avea comanda amânată prioritară. Operatorii care nu mai au comenzi active pot fi reutilizați pentru comenzi ulterioare, iar procesarea continuă până când nu mai există comenzi de procesat sau operatori activi. Rezultatele procesării sunt scrise într-un fișier de ieșire *outDateProcesate.out*, inclusiv detalii despre comenzile procesate, comenzile amânate și momentele de timp.

Tot în clasa Magazin am implementat și metodele pentru *crearea rapoartelor,* ce

conțin date despre angajatul cu cele mai multe comenzi, angajații cu cele mai valoroase comenzi și angajații cu cele mai mari salarii. Pentru aceste metode m-am folosit de atributele implementate special în clasa Angajat și de funcții lambda pentru diferite tipuri de sortări.

În funcția main(), am implementat funcționalitățile aplicației, evidențiind modul în care componentele magazinului interacționează. Am creat un obiect de tip MagazinRock, am preluat datele despre produse, angajați și comenzi din fișiere, iar pentru transparență, am utilizat funcțiile de afișare pentru a organiza informațiile în fișierul de ieșire *outdate.out*.

Interfața aplicației constă într-un *meniu interactiv*, care permite utilizatorului să selecteze și să execute diverse operațiuni pentru gestionarea magazinului. Toate operațiunile din meniu sunt implementate prin *apeluri către funcții simple*, bine definite în clasa MagazinRock, fiecare având un nume sugestiv care reflectă scopul său.

Funcția ***menu()*** a clasei poate funcționa doar dacă sunt îndeplinite două condiții: utilizatorul alege să continue cu meniul și magazinul există. Existența magazinului este verificată la începutul buclei din menu(), prin metoda ***exists(),*** care se asigură că acesta are cel puțin două produse din fiecare categorie, un Manager, trei Operatori și un Asistent. La fiecare iterație, se afișează opțiunile generale: gestiunea angajaților, a produselor, procesarea comenzilor și generarea de rapoarte. Utilizatorul alege o operațiune introducând un număr (validat prin *citire_opt* pentru a preveni opțiuni greșite):

- **Gestionarea angajaților**: dacă utilizatorul alege opțiunea 1 din meniul principal, se afișează un submeniu cu operațiuni specifice pentru angajați: adăugarea (*adaugare _angajat*), modificarea unui angajat, schimbând numele său (*modificare _angajat*), ștergerea unui angajat din listă (*stergere_angajat*), afișarea datelor unui anumit angajat (*afisare_angajat*) sau a tuturor angajaților (*afisare_angajati*).
- **Gestionarea produselor**: opțiunea 2 din meniul principal, pentru care se afișează un submeniu pentru produse: adăugarea unui produs nou, fie articol vestimentar, disc sau disc vintage, fiecare cu detalii specifice (*adaugare_produs_vest, adaugare_produs_disc, adaugare_produs_discVintage*), modificarea stocului unui produs identificat prin cod (*modificare_produs*), ștergerea unui produs din listă (*stergere_produs*), afișarea unui produs după cod (*afisare_produs*) sau a tuturor (*afisare_produse*).
- **Procesarea comenzilor**: opțiunea 3 din meniul principal, pentru care se preia data primei comenzi din lista comenzilor și se procesează toate comenzile existente, actualizând stocurile și atribuind comenzi angajaților (*procesarea_comenzilor*).
- **Generarea rapoartelor**: opțiunea 4 din meniul principal, pentru care se afișează un submeniu pentru rapoarte: raportul cu angajatul care a procesat cele mai multe comenzi (*RAPORT_cel_mai_mare_nr_comenzi*), raportul cu topul angajaților care au procesat cele mai valoroase comenzi (*RAPORT_cele_mai_valoroase_comenzi*), raportul cu topul angajaților cu cele mai mari salarii în luna curentă (*RAPORT_cele_mai_mari_salarii*).
- **Continuarea sau ieșirea din meniu**: după fiecare operațiune, utilizatorul este întrebat dacă dorește să continue, iar dacă răspunde „da”, aplicația revine la meniul principal, altfel se oprește cu un mesaj de încheiere.

Teste efectuate:

- Pentru **magazin**: am testat cazurile în care nu am destui angajați sau destule produse; am verificat afișarea corectă a informațiilor despre un angajat specific și lista tuturor angajaților, dar și despre un produs specific sau toate produsele.
- Pentru **angajați** am încercat: adăugarea unui angajat cu CNP invalid sau care nu era major la data angajării; eliminarea, afișarea și modificarea unui angajat inexistent.
- Pentru **produse** am încercat: adăugarea și ștergerea de produse din categorii inexistente; modificarea și afișarea după cod-uri invalide a produselor; scăderea stocului cu un număr prea mare;
- Pentru **comenzi**: am testat comenzi cu total prea mic sau cu prea multe articole vestimentare și cd-uri; am testat cazurile în care toți operatorii erau ocupați;
- Pentru **rapoarte**: am testat generarea și conținutul fiecărui tip de raport.

Probleme și dificultăți:

- alegerea folosirii structurii **tm** m-a făcut să am dificultăți în formatarea corectă a datelor și gestionarea conversiilor între tipurile de date, mai ales când a fost necesar să lucrez cu date în format **string;**
- citirea din fișiere a fost o altă provocare, mai ales când a fost vorba de comenzi, unde manipulam date complexe, adică un vector de produse;
- având o multidudine de clase, implementarea relațiilor între acestea mi-a dat puțin de cap, adică apelarea funcțiilor corecte la momentul potrivit și gestionarea legăturilor între obiecte.
- o altă provocare a fost implementarea validărilor pentru a asigura corectitudinea datelor citite sau introduse de utilizator; întrucât validările au fost esențiale în toate etapele aplicației, a trebuit să ofer mare atenție implementării lor.
- am utilizat comanda *g++ -std=c++17 main.cpp Angajat.cpp Produs.cpp MagazinRock.cpp Comanda.cpp -o main* pentru a compila aplicația; a fost important să specific standardul C++17 pentru a avea toate funcționalitățile necesare disponibile.