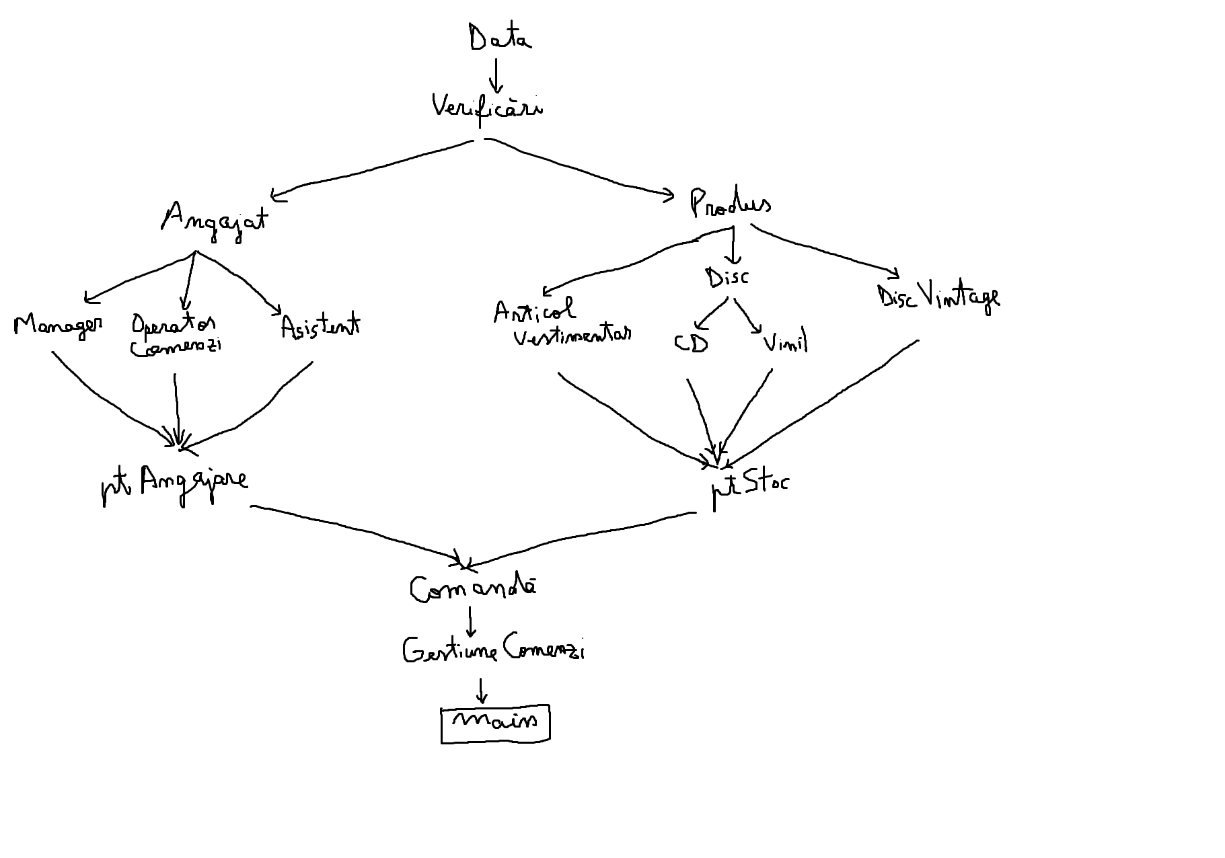
**DOCUMENTAȚIA TEHNICĂ PENTRU ‘ROCK THE SHOP’ - PROIECTUL LA PROGRAMARE ORIENTATĂ PE OBIECTE**

Duduc Dragoș, 324AA sgr.a, 31 dec 2024

**Structura fișierelor:**

**Data:**

* folosită pentru reținerea datei calendaristice în format zz/ll/aaaa.
* am inclus biblioteca <chrono> pentru a avea acces la data reală, extrasă din sistemul de calcul pe care rulează programul și am integrat acest aspect în constructorul fără parametri, adică o variabilă ‘Data d’ va fi inițializată automat cu data curentă, convenabil pentru alte clase și metode
* am realizat metode precum isDateValid(), care verifică dacă o dată este reală (luna are valori între 1 și 12, ziua are valori între 1 si 28/29/30/31 în funcție de lună) și că nu este în viitor, sau getTimeGap(), care returnează timpul dintre două date, folosită mai ales la aflarea vechimii unui angajat

**Verificări:**

* bibliotică cu diverse funcții de verificare pentru nume, CNP, raritatea discurilor vintage sau parola pentru accesul angajatului

**Structurile de angajați:**

* **Angajat** – clasa de bază (interfața), contine mare parte din atributele unui angajat și metode de schimbare a numelui, de verificare a datei de naștere a angajatului în vederea acordării bonusului de 100 RON, sau de demisie. Funcția de demisie ( resign() ) primește ca parametru vectorul în care sunt memorați angajații și șterge poziția pe care se află elementul pentru care este apelată funcția.
* **Manager, OperatorComenzi, Asistent** – clasele derivate din baza Angajat, drept atribute au coeficientul de salariu inițializat cu valoarea corespunzătoare, și ca metode au getSalariu(), moștenită din funcția virtuala din bază.
* **ptAngajare** – bibliotecă cu diverse funcții precum okAng(), care verifică dacă sunt angajați cel puțin un manager, trei operatori de comenzi și un asistent, sau angajare(), care citește datele de la tastatură și returnează un angajat de tipul specificat de utilizator. Biblioteca ptAngajare a fost create pentru funcția angajare() deoarece aveam nevoie de o bibliotecă care să include toate cele trei tipuri de derivate, pentru a le putea accesa constructorii cu parametri.

**Structurile de produse**

* **Produs** – clasa de bază (interfața), conține atribute commune tuturor produselor și metode virtuale pentru aflarea prețului de bază sau a prețului total (inclusiv taxele de ambalare și livrare).
* **ArticolVestimentar, CD, Vinil, DiscVintage** – clasele derivate din baza Produs (mențiune: pentru că au atribute identice, CD și Vinil sunt derivate din clasa intermediară Disc care conține toate atributele și metodele necesare)
* **ptStoc** – creată în aceeași idee ca ptAngajat, o bibliotecă care să includă toate derivatele și în care să pot crea o funcție care să returneze constructorii tuturor derivatelor, conține funcția de adaugareProdus(), modificareStoc() sau ștergereStoc().

**Comandă**

* conține atributele necesare unei comenzi (produsele comandate, sub forma unui vector de pair-uri <Produs, cantitate>, durata de completare, calculată cu formula (60 \* nrProduse) secunde, valoare, operatorul care o procesează); constructorul cu parametri verifică automat dacă se poate realiza (preț de bază > 100 RON, maxim 3 articole vestimentare și 5 disc-uri)

**GestiuneComenzi**

* conține trei vectori de comenzi (active, în așteptare și finalizate) și vectorul de angajați ai companiei
* metode precum numarComenziActive(), care primește ca parametri un operator de comenzi și returnează numărul de comenzi pe care operatorul respective le procesează, gaseșteOperator(), care returnează operatorul din companie cu cel mai mic număr de comenzi curente în procesare, actualizeazaStareComenzi, care verifică dacă a trecut durata de completare pentru fiecare comandă active, iar dacă a fost completată, o mută în vectorul de comenzi completate și marchează acest lucru printr-un mesaj, după care parcurge vectorul de comenzi în așteptare și verifică dacă vreun operator este liber ( cu găseșteOperator() ) pentru a primi o nouă comandă
* metodele de generare a raportărilor, care analizează vectorul de comenzi completate, sortează informația în felul specificat și generează fișierul csv cerut.

**MAIN**

În main am introdus datele pentru angajați și produse și am creat o variabilă de tip GestiuneComenzi, care va fi folosită în interfața de utilizator.

Pentru interfată, am realizat un switch cu 16 cazuri, unul pentru fiecare situație de care am avut nevoie. Din pagina de bază se pot realiza comenzi sau, dacă utilizatorul este angajat al firmei, se poate autentifica cu CNP-ul și parola contului său de angajat și poate accesa setările magazinului, adică poate să interacționeze cu structura de angajați a magazinului, cu structura de produse, poate monitoriza comenzile plasate până la momentul respectiv sau poate genera raportări. În ceea ce privește structurile de date folosite, angajații sistemului sunt memorați într-un vector de adrese de bază, iar produsele sunt memorate într-un arbore binar echilibrat (map) prin care cheia de acces este ID-ul produsului, iar obiectul reținut în map este un pair Produs\* + cantitate.

**TESTE**

* în setul 1 nu am introdus numărul corespunzător de manageri și se poate testa că un client nu va putea realiza comenzi
* în setul 2 nu am introdus numărul corespunzător de produse și, din nou, magazinul nu va putea functiona corespunzător
* în setul 3 magazinul functionează corespunzător, cu suficiente produse cât pentru a se testa realizarea de comenzi, se poate verifica că o comandă nu poate fi plasată dacă nu are preț de bază de cel puțin 100 RON sau dacă are mai mult de 3 articole vestimentare sau 5 discuri, se poate verifica gestiunea automată a comenzilor și generarea de rapoarte.