Magazin electronic

Să se realizeze o aplicație de tip consolă în C++ care să utilizeze principiile POO pentru realizarea funcționalităților unui magazin electronic.

Cerințe:

- introducerea a două tipuri de utilizatori: magazin și client
- cerințe specifice magazinului:
  - adăugarea de produse
  - editarea de produse
  - ștergerea de produse
  - prelucrarea comenzilor
  - realizarea unui raport (într-un fișier text) cu privire la comenzile primite
- cerințe specifice clientului:
  - vizualizarea produselor din magazin
  - selectarea produselor dorite și introducerea acestora într-un coș electronic
  - scoaterea de produse din coșul electronic
  - completarea datelor personale ale clientului
  - trimiterea comenzii cu produsele din coșul electronic. Se vor furniza și datele clientului (de ex. nume, adresa, etc.)
  - realizarea unui raport (într-un fișier text) cu privire la comanda trimisă
- toate funcționalitățile vor fi puse la dispoziție prin intermediul unei interfețe de tip meniu realizat în consolă
- la deschiderea aplicației se va interoga tipul de utilizator cu care se intră și în funcție de utilizatorul ales (magazin sau client) se va pune la dispoziție meniul specific

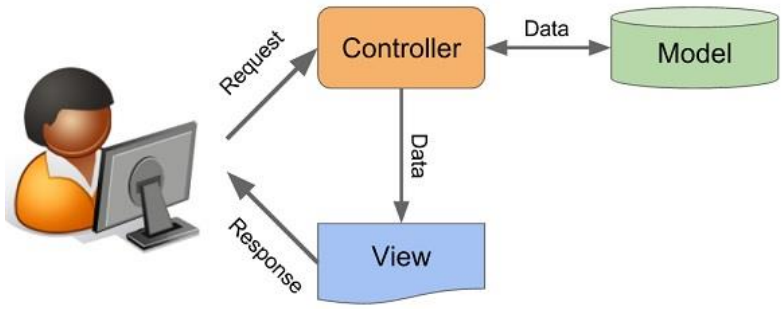
Cerințe tehnice:

- să se realizeze cel puțin 5 clase
- să se realizeze cel puțin o ierarhie de clase care să implementeze principiile moștenirii (cel puțin 3 clase utilizate în această ierarhie pentru a se evidenția beneficiile moștenirii)
- să se realizeze cel puțin o clasă de tip interfață, iar referința la această interfață să fie folosită în program când vor fi referite obiectele copil. Să se folosească polimorfismul.
- atributele să fie protejate prin mecanismul de încapsulare, iar accesul la acestea să se facă prin getteri și setteri (metode accesor)
- să se realizeze constructori default și cu parametri pentru clasele care conțin entități. Să se facă validarea parametrilor. Să se realizeze destructorul, constructorul de copiere și supraîncărcarea operatorului egal pentru aceste clase.
- să se facă în proiect cel puțin 3 supraîncărcări de operatori (diferiți de operatorul =, << și >>)
- să se stocheze persistent în fișiere binare produsele existente în magazin și comenzile trimise de clienți

- rapoartele menționate în cerințele de funcționalitate ale aplicației vor fi salvate în fișiere text
- să se folosească cel puțin o clasă din STL pentru stocarea unei structuri de date (ex. listă, coadă sau stivă etc.)

Sugestie implementare MVC (Model-View-Controller)

Model-View-Controller este un șablon (design pattern) utilizat în dezvoltare software pentru realizarea de interfețe.



(sursa: https://helloacm.com/model-view-controller-explained-in-c/)

Presupune organizarea codului sursă în trei tipuri de clase:

- Model - clase care descriu entitățile cu utilizatorul lucrează
- Controller - clase ce definesc comportamente și funcționalități ale programului
- View - clase ce definesc modul de afișare a datelor

Toate implementările din cerințele de mai sus vor fi testate în main.

Nu se acceptă soluții care au erori de compilare și se vor puncta cu nota 1/10. Cerințele care conțin erori la runtime (erori de HEAP) vor fi punctate pe jumătate. Netrimiterea proiectului la timp sau trimiterea unui fișier care nu are legătură cu proiectul se va nota cu 0. **Aveți grijă ce încărcați, să nu încărcați altceva! Dacă aveți probleme și nu puteți încărca la timp anunțați-mă înainte să expire termenul și vedem ce putem face, dar să nu anunțați după!** Cerințele care sunt implementate în clasă, dar care nu sunt testate în main vor fi punctate pe jumătate.

Se va trimite ori un fișier cpp cu rezolvarea proiectului ori se va trimite o arhivă cu proiectul vostru care va conține fișiere .h și .cpp, denumirea fișierului cpp/arhivei cu proiectul având următorul format: 10XX_EchipaX_Proiect.cpp/10XX_EchipaX_Proiect.rar, unde 10XX este numărul grupei din care faceți parte sau la care recuperați, dacă aveți restanță, și EchipaX este numărul echipei din care faceți parte. Proiectul va fi realizat de echipe de câte 4 – 5 studenți.