Programare orientată pe obiecte – Test de laborator
Seria 13 – 31 mai 2016

Enunț

La restaurantul X se testează o nouă aplicație software de gestiune a comenzilor. Aplicația are o interfață prietenoasă iar programarea sa este realizată folosind conceptele POO. Clasele care stau la baza creării aplicației software sunt:

- clasa **Produs**, având câmpurile **denProd** (șir de caractere de dimensiune variabilă, reprezentând denumirea produsului) și **pretProd** (float, reprezentând prețul unei porții din produs)
- clasa **Comanda** caracterizată de **Num** (întreg, reprezentând numărul comenzii), **Prod** (obiect Produs, reprezentând produsul comandat), **nrPortii** (întreg, reprezentând numărul de porții comandate din produsul respectiv) și **data** (cu 3 câmpuri întregi: zi, lună și an, reprezentând data comenzii)
- clasa **Ospatar**, având urmatoarele câmpuri: **Nume** (șir de maxim 30 de caractere, reprezentând numele ospătarului), **comenzi** (vector alocat dinamic de obiecte Comanda), **nrComenzi** (întreg, reprezentând numărul de comenzi preluate), **gen** (caracter, cu valorile M sau F) și **vârsta** (întreg).

O variantă demo este pusă la dispoziția clienților interesați, astfel că în interfața denumită sugestiv main(), sunt executate automat secvențele de mai jos. Implicit, data unei comenzi este data curenta. Numărul comenzii se generează automat, la crearea unei comenzi noi, prin incrementarea numărului comenzii anterioare.

```
Produs meniu[4] = { Produs(”frigarui”, 30), Produs(”cola”,7.5), Produs(”cafea”,5) };
Comanda c1(”frigarui”, 2., 31, 5, 2016), c2(”cola”,3), c3(”cafea”,1), c4 = c2, c5;
c3 = c3 + 4; // se comanda încă 4 cafele
c2++; // se mai comandă o cola
c1.del(); //se anulează comanda c1
cin >> c5; //se citește comanda c5
cout << c4 << endl << c5; //se afișează comanzile c4 și c5
Comanda *com1 = new Comanda[4], *com2 = new Comanda[4];
com1[0] = c1; com1[1] = c2; com1[2] = c3; com2[0] = c4; com2[1] = c5;
Ospatar o1(“Ionescu”,com1,3,'M',25), o2(“Popescu”,com2,2,'F',30);
cout << o1 << o2;
if (o2 > o1) cout << “Ospatarul ” << o2.nume() << “a servit mai multe comenzi decat ospatarul ”
<< o1.nume() << endl;
else if (o2 == o1) cout << “Numar egal de comenzi intre ospatarii ” << o2.nume() << “si ” <<
o1.nume() << endl;
else cout << “Ospatarul ” << o2.nume() << “a servit mai puține comenzi decat ospatarul ” <<
o1.nume() << endl;
```

Varianta demo conține doar mențiunile standard enunțate anterior, dar versiunea completă a aplicației software include și gestionarea comenzilor speciale de tip **ComSpec**. O comandă soecială este o comandă standard care e modificată pe gustul

clientului, astfel că ospătarul trebuie să adăuge cerința clientului **ObsC** (șir de caractere de dimensiune variabilă) și prețul suplimentar **pretSupl** (float), rezultat din cerința clientului, pentru fiecare porție comandată.

În plus, restaurantul X oferă si posibilitatea de a livra comenzile la domiciliu (atat pe cele standard, cat și pe cele speciale) și pentru aceasta, aplicația software trebuie să gestioneze comenzi online, de tip **ComOnline**. O comandă online trebuie să conțină adresa de livrare a fiecărei comenzi **adrLivr** (șir de caractere de dimensiune variabilă), precum și comisionul de livrare **comLivr** (întreg), care este un procent de 5% din valoarea comenzii, dar poate scădea până la 0, în funcție de valoarea comenzii.

Cerințe

A. Să se implementeze clasele necesare și orice alt element de program ce se impune pentru a executa correct funcția main() descrisă în variant de demo a sof
B. Să se evidențieze funcțiile complete ale softului de gestiune a comenzilor prin executarea următoarelor cerințe:

1. Afișarea tuturor comenzilor prelucrate în ziua de 31.05.2016.
2. Afișarea numărului de porții de “papanași” standard si “papanași cu nucă de cocos” comandate în luna mai 2016.
3. Afișarea procentului valorii totale cumulate a comenzilor online din valoarea totală cumulată a tuturor comenzilor înregistrate în luna mai 2016.
4. Afișarea ospătarului care a servit comenzi în valoare totală cumulată cea mai mare, față de toți ceilalți ospătari, de la începutul anului 2016 și până la data curentă.

Precizări

1. Timpul de lucru este de 90 de minute.
2. La sfârșitul timpului de lucru, studenții vor salva pe stick-ul de memorie al profesorului supraveghetor fișierul sursă cu extensia cpp. Acesta trebuie să conțină pe primul rând un comentariu cu numele și prenumele studentului, grupa și compilatorul folosit.
3. Sursa predată trebuie să compileze. Sursele care au erori de compilare nu vor fi luate în considerare. Înainte de predarea surselor, studenții vor pune în comentariu eventualele părți din program care au erori de compilare sau nu funcționează corespunzător.
4. Se acceptă și soluții parțiale, care nu respectă toate cerințele din enunț, dar sunt funcționale. Acestea vor fi depunctate corespunzător.
5. În implementarea programului se vor utiliza cât mai multe dintre noțiunile de programare orientată pe obiecte, care au fost studiate pe parcursul semestrului și care se potrivesc cerințelor din enunț.
6. Condițiile minimale de promovare a testului sunt ca programul să fie scris cu clase și să ruleze corect funcția main() în varianta demo, conform cerințelor prezentate în enunț.
7. Orice tentativă de fraudă se va pedepsi conform regulamentelor Universității.