Exercițiul 8

Clasa `Client` reprezintă o persoană caracterizată prin nume, un cod de identificare şi un nivel de discount.

```
class Client {
  private:
    int id;
    string nume;
    int discount;
  public:
    Client(int cid = 1, string cnume = "", int cdiscount = 5);
    Client(const Client&);

    Client& operator= (const Client&);

    int getID();
    string getNume();

    int getDiscount();
    void setDiscount(int);

    void print(ostream&);
};
```

Clasa `Client` va conține următoarele elemente:

- trei atribute private: `nume` - numele persoanei, `id` – cod de identificare şi `discount` – procentul de discount pe care i-l oferă un comerciant.
- constructor cu parametri, constructor de copiere, metode *getter* şi *setter* pentru procentul de discount şi metode *getter* pentru atributele `nume` şi `id`.
- operatorul de atrbuire supraîncărcat.
- o metodă `print()` ce afişează mesajul

"`Client[#id: AAA, nume: BBB, discount: CCC]`", unde `AAA` reprezintă codul de identificare, `BBB` numele iar `CCC` procentul de discount.

- După modelul de la clasa `Circle`, se doreşte să se specifice declarația clasei în fişierul `Client.h` iar definiția metodelor clasei să fie realizată în fişierul `Client.cpp`.

Referințe

Clasa `Author`, clasa `Carte`.

Clasa `Factura` reprezintă o factură identificată prin serie, client şi valoare totală.

```
class Factura {
  private:
    int serie;
    Client client;
    double valoare;
  public:
    Factura(int fid, Client& fc, double fv = 100.0);
    Factura(const Factura&);

    Factura& operator= (const Factura&);

    int getSerie();
    string getNumeClient();

    Client getClient();
    void setClient(const Client&);
    double getValoare();
    void setValoare(double);

    double getValoareCuDiscount();
    void print(ostream&);
};
```

Clasa `Factura` va avea următoarea structură:

- trei atribute private: `serie` – seria facturii, `client` – date de identificare ale clientului, `valoare` – valoarea totală a facturii.
- constructor cu parametri, constructor de copiere.
- operatorul de atrbuire supraîncărcat.
- metode *getter* şi *setter* pentru datele membre `client` şi `valoare` (`getClient()`, `setClient(Client&)`, `getValoare()`, `setValoare(double)`).
- o metodă *getter* pentru atributul `serie`.
- o metodă `getValoareCuDiscount()` ce returnează valoarea facturii cu reducerea acordata clientului.
- o metodă `getNumeClient()` ce returnează numele clientului.
- o metodă `print()` ce afişează mesajul

"`Factura[#serie: AAA, Client[#id: BBB, nume: CCC, discount: DDD], valoare: EEE (FFF)]`" unde `AAA` reprezintă seria facturii, `BBB` codul de identificare al clientului, `CCC` numele clientului, `DDD` procentul de reducere aplicat, `EEE` valoarea totală a facturii fără reducere iar `FFF` valoarea facturii după ce a fost aplicat procentul de reducere.

- După modelul de la clasa `Circle`, se doreşte să se specifice declaraţia clasei în fişierul `Factura.h` iar definiţia metodelor clasei să fie realizată în fişierul `Factura.cpp`.

Proiectul va trebui să conțină atât fișierele corespunzătoare clasei `Client` (`Client.h` și `Client.cpp`) cât și fișierele corespunzătoare clasei `Factura` (`Factura.h` și `Factura.cpp`).

Codul de testare al clasei `Factura` este următorul (`test-factura.cpp`):

```
int main() {
  Client a(101, "Cristina");
  Client b(104, "Ioan", 10);

  Factura f1(1011, a);

  cout << f1.getSerie() << " : " << f1.getNumeClient()
       << " : " << f1.getValoare() << endl;

  f1.setClient(b);
  f1.print(cout);
  cout << endl;

  Factura f2 = f1;
  Client c1(108, "Andrei", 7);

  f2.print(cout);
  cout << endl;

  f2.setClient(c1);
  f2.setValoare(1000);

  f2.print(cout);
  cout << endl;

  return 0;
}
```

To do:

1. Pentru *versiunea a doua a aplicației*, să se modifice clasa `Client` astfel încât numele unui client să fie păstrat într-un șir de caractere alocat dinamic:

```
class Client {
  private:
    int id;
    char* nume;
    int discount;
  public:
    Client(int cid = 1, const char* cnume = "", int cdiscount = 5);
    Client(const Client&);

    Client& operator= (const Client&);

    int getID();
    char* getNume();

    int getDiscount();
    void setDiscount(int);

    void print(ostream&);
};
```

Se va avea în vedere efectuarea unor eventuale modificări în clasa `Factura`, necesare pentru buna funcționare a codului existent.