Lucrare de laborator nr. 1
Recapitulare diagrame UML. Principiile SOLID

1.1 Scopul lucrării

În cadrul acestei lucrări se urmărește recapitularea diagramelor UML (diagrama cazurilor de utilizare și diagrama de clase) și consolidarea principiilor SOLID.

1.2 Exerciții rezolvate

Exercițiu 1.1. Se doreşte dezvoltarea unui software interactiv destinat studiului geometriei triunghiului prin atingerea următoarelor scopuri:

- desenarea interactivă a triunghiurilor prin înlocuirea creionului şi a riglei cu *mouse*-ul;
- calcularea și afișarea unor proprietăți: lungimile laturilor, măsurile unghiurilor, perimetrul, aria, raza cercului înscris, raza cercului circumscris, etc.
- determinarea și vizualizarea unor elemente specifice unui triunghi:
  - puncte importante într-un triunghi: centrul de greutate, ortocentrul, punctul lui Lemoine, etc.
  - linii importante într-un triunghi: mediane, mediatoare, înălțimi, bisectoare, simediane, dreapta lui Euler, linii mijlocii, etc.
  - ceruri speciale: cercul circumscris triunghiului, cercul înscris în triunghi, cercul celor 9 puncte, etc.
  - triunghiuri speciale: triunghiul ortic, triunghiul median, etc.
- salvarea / încărcarea unui triunghi.

Realizați diagrama cazurilor de utilizare și diagrama de clase care să cuprindă tipurile de date necesare implementării cerințelor specifice cazurilor de utilizare.

Soluție:

O variantă pentru diagrama cazurilor de utilizare este prezentată în figura 1.1, iar pentru diagrama de clase în figura 1.2.

Diagrama cazurilor de utilizare cuprinde:

- un actor - utilizatorul aplicației software;
- 11 cazuri de utilizare;
- relații:
  - 8 relații de asociere între actor și cazurile de utilizare;
  - 3 relații de dependență și 6 relații de extindere între cazurile de utilizare.

Datorită celor 8 relații de asociere utilizatorul îi sunt permise următoarele 8 opțiuni:

- Desenarea unui triunghi;
- Salvarea unui triunghi într-un fișier;
- Încărcarea (citirea) unui triunghi dintr-un fișier;
- Vizualizarea unor proprietăți specifice triunghiului;
- Vizualizarea punctelor importante specifice triunghiului;
- Vizualizarea liniilor importante specifice triunghiului;
- Vizualizarea cercurilor speciale asociate triunghiului;
- Vizualizarea triunghiurilor speciale asociate triunghiului.

Figura 1.1. Diagrama cazurilor de utilizare corespunzătoare exercițiului 1.1

Diagrama de clase cuprinde 9 clase, dintre care 2 clase sunt abstracte. Din clasa abstractă ***ElementGeometric*** sunt derivate clasele: ***Punct***, ***Dreapta***, ***Segment*** și ***FiguraGeometrica***, iar din clasa ***Dreapta*** este derivată clasa ***Semidreapta***. Din clasa abstractă ***FiguraGeometrica*** sunt derivate clasele: ***Triunghi*** și ***Cerc***.

Între clasele ***Punct*** și ***Dreapta*** există relație de compunere deoarece un obiect de tip ***Dreapta*** este identificat de 2 obiecte de tip ***Punct***. Între clasele ***Punct*** și ***Segment*** există relație de compunere deoarece un obiect de tip ***Segment*** este identificat de 2 obiecte de tip ***Punct***. Între clasele ***Punct*** și ***Triunghi*** există relație de compunere deoarece un obiect de tip ***Triunghi*** este identificat de 3 obiecte de tip ***Punct***. Între clasele ***Punct*** și ***Cerc*** există relație de compunere deoarece un obiect de tip ***Cerc*** este identificat de un obiect de tip ***Punct***.

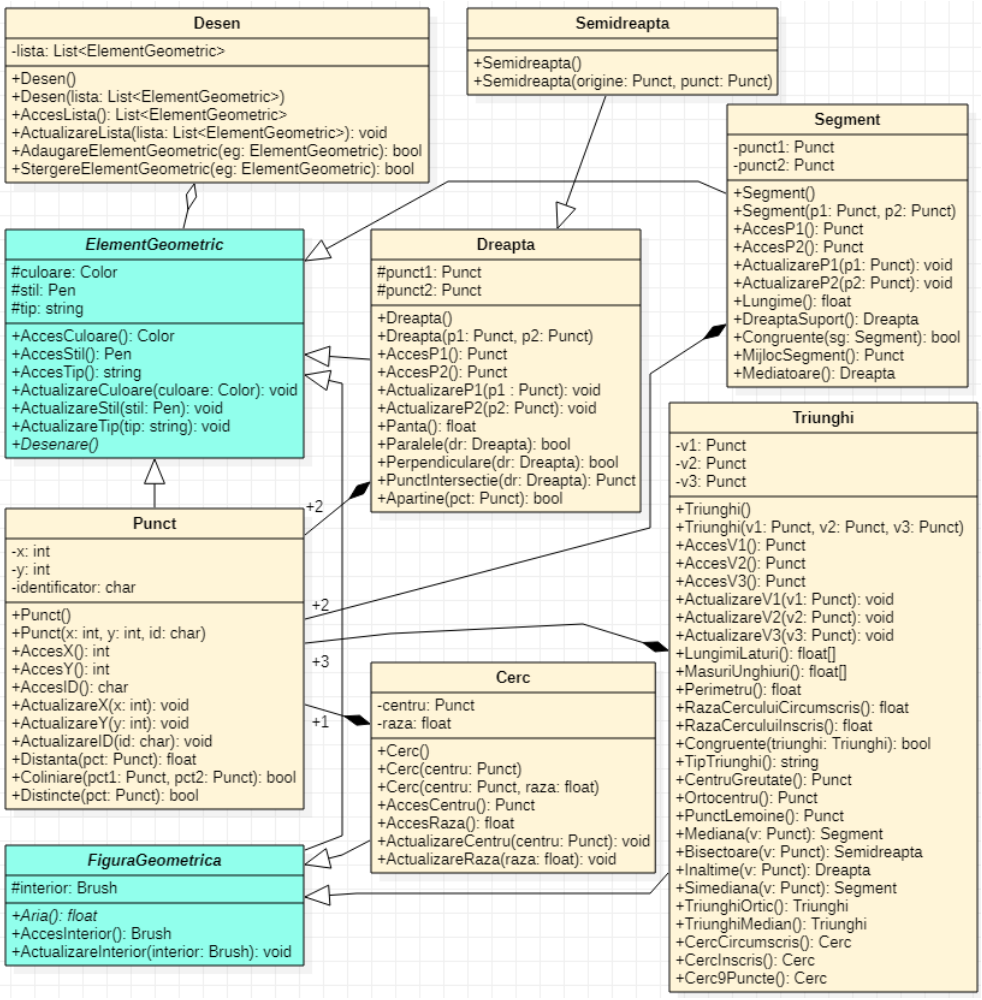

Figura 1.2. Diagrama de clase corespunzătoare exercițiului 1.1

Exercițiu 1.2. Să se dezvolte o aplicație informatică care va putea fi utilizată la un concurs de patinaj artistic. Aplicația va avea 4 tipuri de utilizatori: spectator, organizator concurs, arbitru și patinator.

Utilizatorii de tip spectator pot efectua următoarele operații fără autentificare:

- Vizualizarea listei tuturor patinatorilor înscriși la concurs;
- Filtrarea listei patinatorilor după anumite criterii;
- Căutarea unui patinator;
- Vizualizarea clasamentului final și a unor statistici legate de patinatori (numărul de patinatori care a obținut nota maximă pentru fiecare probă, etc.).

Utilizatorii de tip organizatori ai concursului pot efectua:

- Toate operațiile permise utilizatorilor de tip spectator;
- Funcțiile CRUD în ceea ce privește persistența patinatorilor, mai puțin gestionarea notelor;
- Generare raport cu clasamentul final pentru fiecare probă în mai multe formate: csv, txt, xml.

Utilizatorii de tip arbitru pot efectua:

- Toate operațiile permise utilizatorilor de tip spectator;
- Gestionarea notelor concurenților.

Utilizatorii de tip patinator pot efectua următoarele operații:

- Toate operațiile permise utilizatorilor de tip spectator;
- Crearea unui cont și înscrierea la concurs.

Realizați diagrama cazurilor de utilizare și diagrama de clase care să cuprindă tipurile de date necesare implementării cerințelor specifice cazurilor de utilizare.

Soluție:

O variantă pentru diagrama cazurilor de utilizare este prezentată în figura 1.3. iar pentru diagrama de clase în figura 1.4.

Diagrama cazurilor de utilizare cuprinde:

- 4 actori - spectator, organizator concurs, arbitru și patinator;
- 12 cazuri de utilizare;
- relații:
  - 13 relații de asociere între actor și cazurile de utilizare;
  - 8 relații de dependență și o relație de extindere între cazurile de utilizare.

Figura 1.3. Diagrama cazurilor de utilizare corespunzătoare exercițiului 1.2

Diagrama de clase cuprinde 9 clase. Din clasa ***Persoana*** este derivată clasa ***Utilizator***, iar din clasa ***Utilizator*** sunt derivate clasele ***Patinator*** și ***Arbitru***. Clasa ***Proba*** este compusă din mai multe obiecte de tip ***Patinator*** și ***Arbitru***.

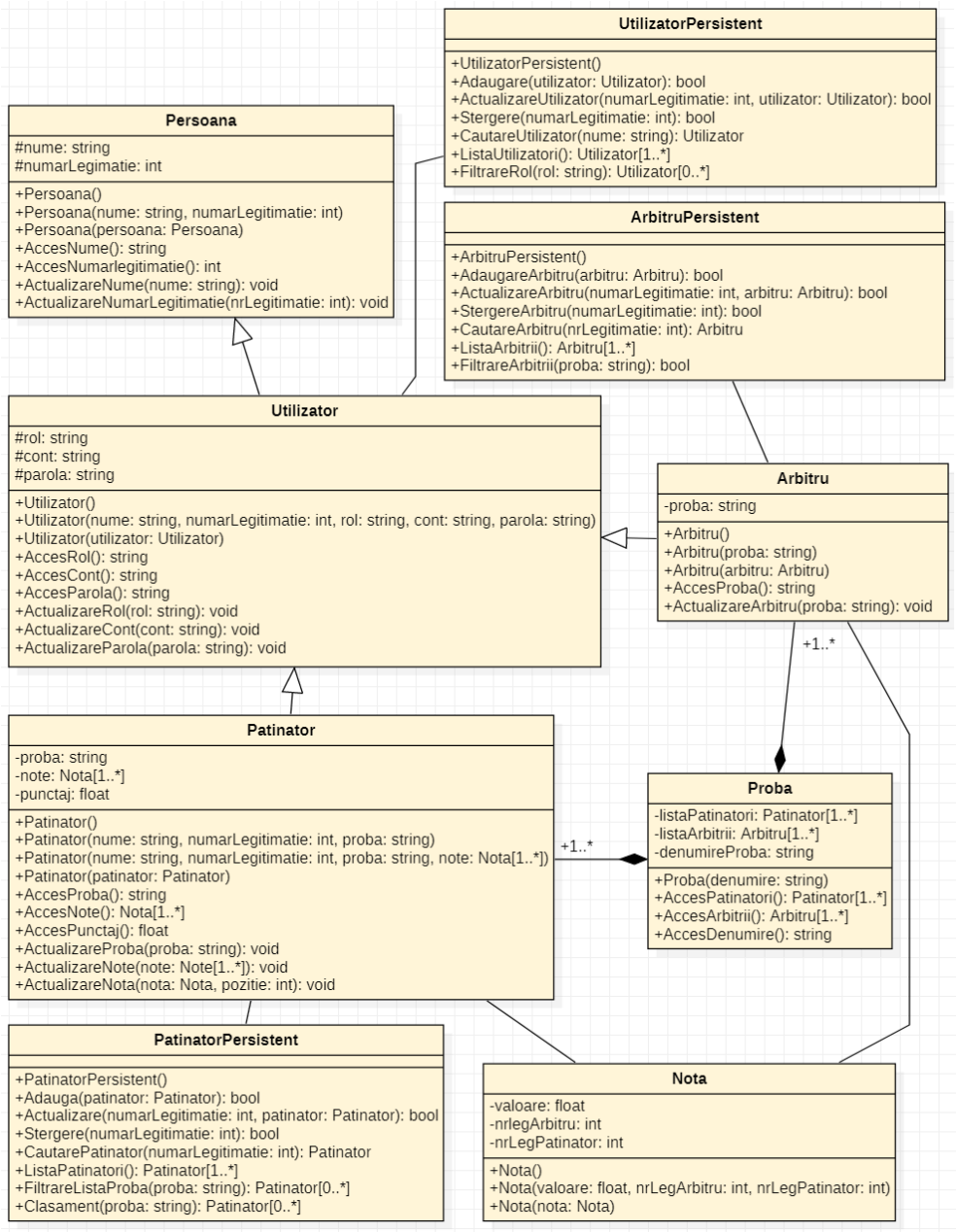

Figura 1.4. Diagrama de clase corespunzătoare exercițiului 1.2

Exercițiu 1.3. Modificați diagrama de clase prezentată în figura 1.5 astfel încât dacă se dorește adăugarea unei noi distanțe (de exemplu distanța Cebâșev) să fie îndeplinit principiul deschis-închis.

Soluție:

O variantă de rezolvare este prezentată în figura 1.6.

În cazul clasei ***Distanta2D***, pentru a îndeplini principiul deschis-închis, ar fi util proiectarea unei clase abstracte ce va cuprinde metoda abstractă ***Distanta()*** ce va fi implementată în clasele derivate: ***DEuclidiana2D***, ***DManhattan2D*** și ***DCebasev2D***. Clasa ***Distanta3D***, pentru a îndeplini principiul deschis-închis, este transformată într-o clasă abstracte ce va cuprinde metoda abstractă ***Distanta()*** ce va fi implementată în clasele derivate: ***DEuclidiana3D***, ***DManhattan3D*** și ***DCebasev3D***.

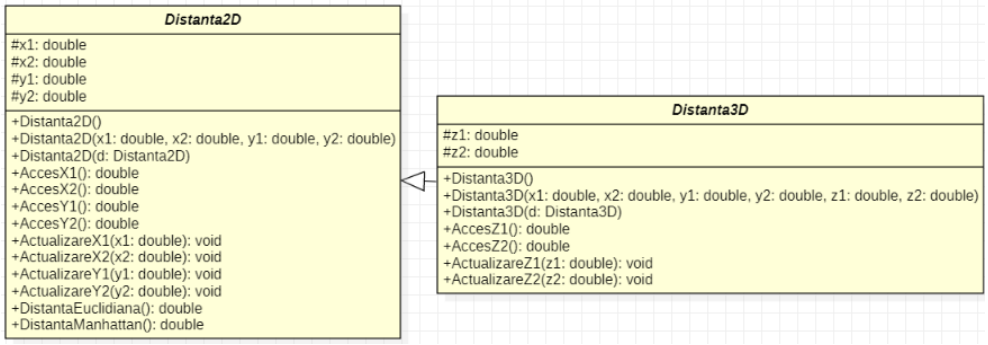

Figura 1.5. Diagramă de clase

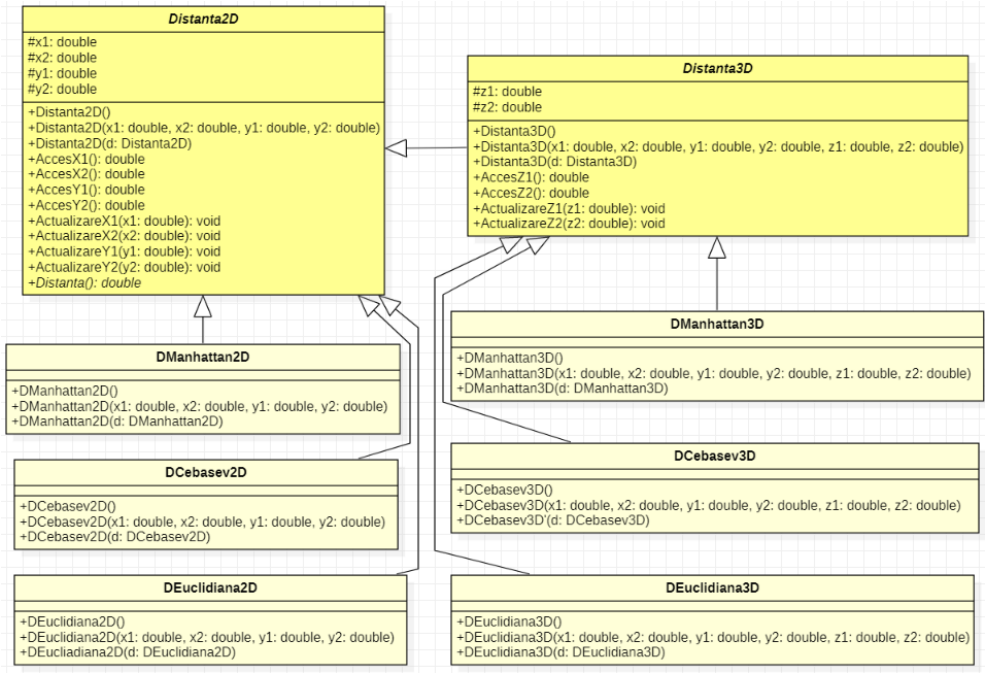

Figura 1.6. Diagramă de clase

Exercițiu 1.4. Modificați diagrama de clase prezentată în figura 1.7 astfel încât să fie îndeplinite principiile SOLID.

Soluție:

O variantă de rezolvare este prezentată în figura 1.8.

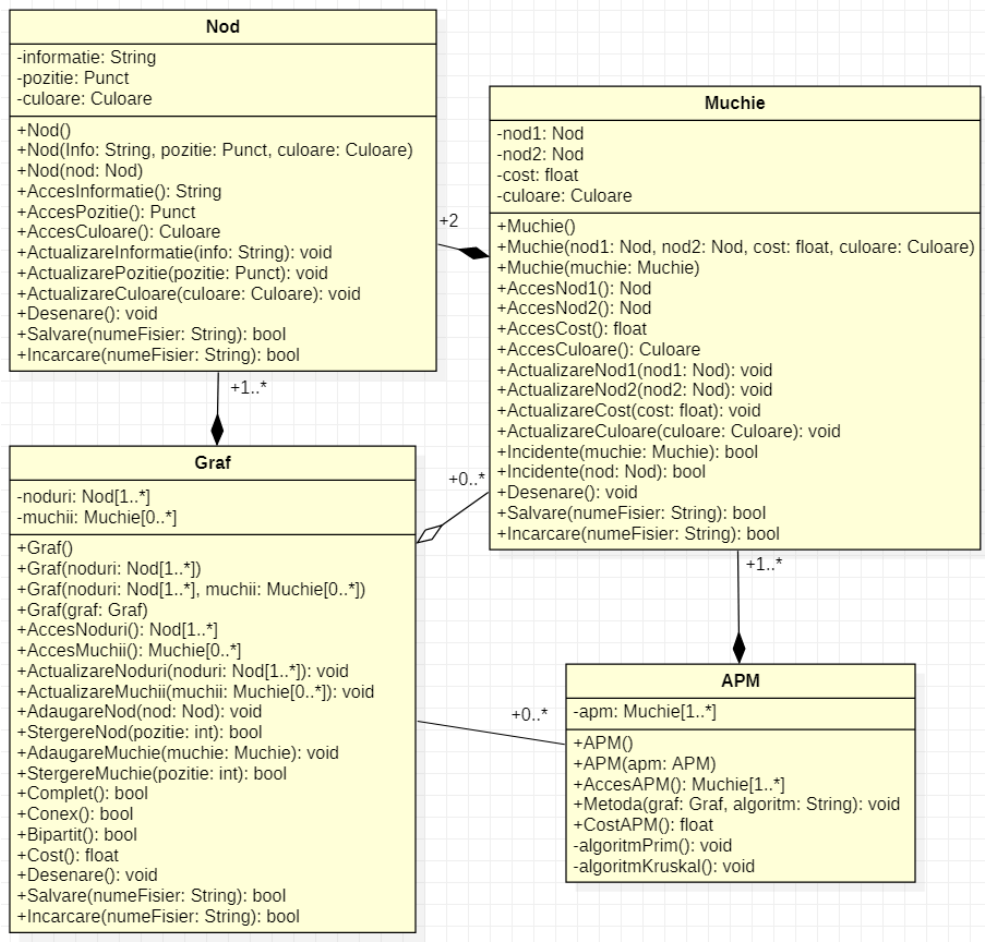

Figura 1.7. Diagramă de clase

Pentru a fi îndeplinit principiul responsabilității unice, clasa ***Graf*** ar trebui să fie caracterizată doar de metode specifice conceptului de graf. Metode ce se referă la persistența obiectelor de tip graf (salvare, încărcare) are trebui să facă parte din altă clasă. În acest scop este proiectată clasa ***PersistentaGraf***. Metoda ce se referă la reprezentarea geometrică a unui graf (desenare) ar trebui să facă parte din altă clasă, astfel fiind proiectată clasa ***GrafDesenat***. Similar, ar trebui create clase pentru conceptele de nod și muchie.

Deoarece clasele ***PersistentaGraf***, ***PersistentaNod*** și ***PersistentaMuchie*** vor fi caracterizate de cele 2 metode: salvare și incarcare, dar și atributul ce identifică fișierul utilizat la realizarea persistenței, se recomandă utilizarea unei clase abstracte ***Persistenta***.

Deoarece clasele ***GrafDesenat***, ***NodDesenat*** și ***MuchieDesenat*** vor fi caracterizate de metoda de desenare, se recomandă utilizarea unei interfeței ***IDesenare***.

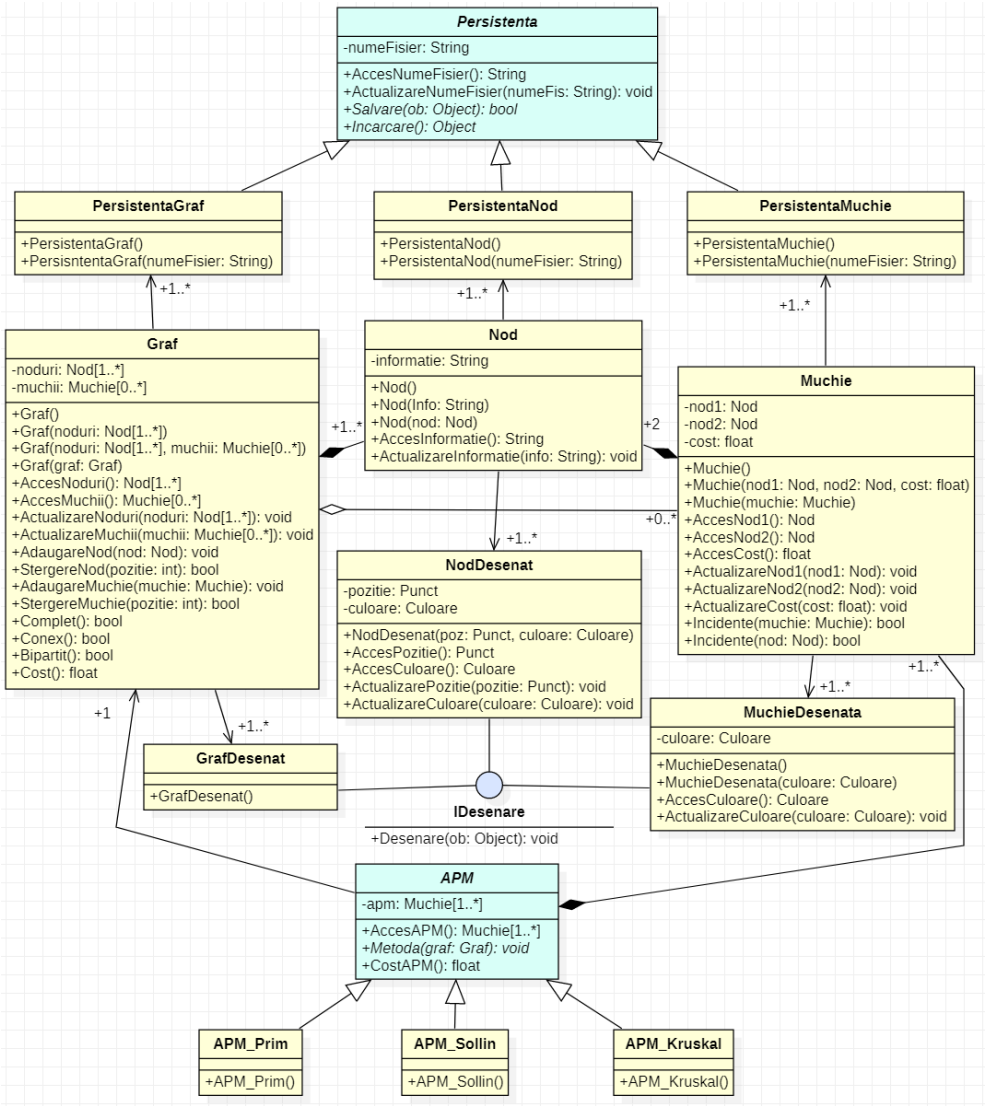

Figura 1.8. Diagramă de clase - SOLID

În cazul clasei APM utilizată la determinarea arborelui partial de cost minim, pentru a îndeplini principiul deschis-închis, ar fi util proiectarea unei clase abstracte ***APM*** ce va cuprinde metoda abstractă ***Metoda*** ce va fi implementată în clasele derivate. Astfel se vor putea utiliza alți algoritmi de determinare a arborelui parțial de cost minim (de exemplu, algoritmul lui Sollin).

1.3 Exerciții propuse spre rezolvare

Exercițiu 1.5. Modificați clasa prezentată în figura 1.9 astfel încât să fie îndeplinite principiile SOLID.

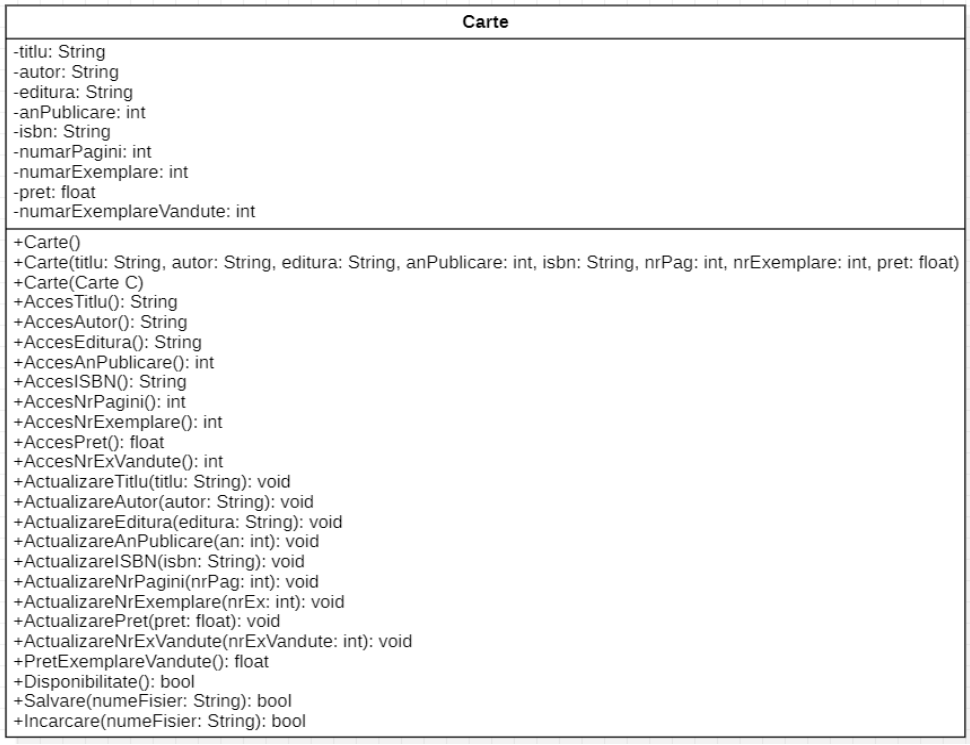

Figura 1.9. Clasa „Carte”

Exercițiu 1.6. Se doreşte dezvoltarea unui sistem software pentru managementul operațiilor dintr-o tipografie. Tipografia furnizează servicii de comercializare a hârtiei: tipărire, tăiere, legare şi lipire. Aceste servicii pot fi combinate rezultând servicii mai complexe. Fiecare serviciu are un anumit preţ. La acest preţ se adaugă adaosul comercial al tipografiei, dar se poate aplica şi un discount pentru comenzi mari. Sistemul soft trebuie să permită:

- Preluarea informațiilor despre client;
- Administrarea serviciilor simple (specificare denumire, preț, adaos comercial, discount aplicat);
- Administrarea serviciilor compuse (servicii componente, discount aplicat);
- Generare rapoarte:
  - Grafic comenzi livrate/anulate;
  - Grafic servicii simple/servicii compuse.

Realizați diagrama cazurilor de utilizare și diagrama de clase care să cuprindă tipurile de date necesare implementării cerințelor specifice cazurilor de utilizare.