Lucrare de laborator nr. 3
GRASP. Șablonul arhitectural MVP

3.1 Scopul lucrării

În cadrul acestei lucrări se urmărește consolidarea principiilor de proiectare a claselor GRASP (General Responsibility Assignment Principles), precum utilizarea eficientă a șablonului arhitectural MVP (Model-View-Presenter).

3.2 Exerciții rezolvate

Exercițiu 3.1. Se doreşte dezvoltarea unui sistem software pentru managementul operațiilor dintr-o tipografie. Tipografia furnizează următoarele servicii de comercializare a hârtiei: tipărire, tăiere, legare şi lipire. Fiecare serviciu este caracterizat de un anumit preţ, dar se poate aplica şi un discount pentru comenzi mari. Sistemul soft trebuie să permită administrarea unei comenzi și generarea facturii pentru o comandă finalizată.

Proiectați o diagramă de clase care să îndeplinească principiile GRASP.

Soluție:

În soluția prezentată, în prima etapă, s-au identificat 4 concepte necesare implementării: client, serviciu, serviciu comandat și comandă. Cele 4 clase corespunzătoare acestor concepte sunt prezentate în figura 3.1.

Clasa ***Client*** este caracterizată de atributele: nume, CNP, număr de telefon și adresa de email a clientului. Clasa ***Serviciu*** este caracterizată de atributele: denumire și prețul unitar al serviciului furnizat de tipografie. Clasa ***ServiciuComandat*** este caracterizată de atributele: serviciu solicitat de client, cantitatea solicitată și discountul aplicat. Clasa ***Comanda*** este caracterizată de atributele:

- Numărul de ordine al comenzii;
- Data solicitării comenzii;
- Clientul care a solicitat comanda;
- Lista serviciilor comandate;
- Discountul aplicat (dacă e cazul) pentru întreaga comandă.

O primă decizie trebuie luată legat de obiectele care ar putea calcula valoarea unei comenzi. Conform principiului ***Information Expert*** clasa ***ServiciuComandat*** trebuie să conțină o metodă care să calculeze prețul serviciului comandat (în baza informațiilor deținute: preț unitar al serviciului solicitat, cantitate solicitată și discount aplicat). Implentarea acestei metode este următoarea:

```
public float PretServiciuComandat()
{
    float pret = this.cantitate * this.serviciu.PretUnitar;
    pret -= pret * this.discount/100;
    return pret;
}
```

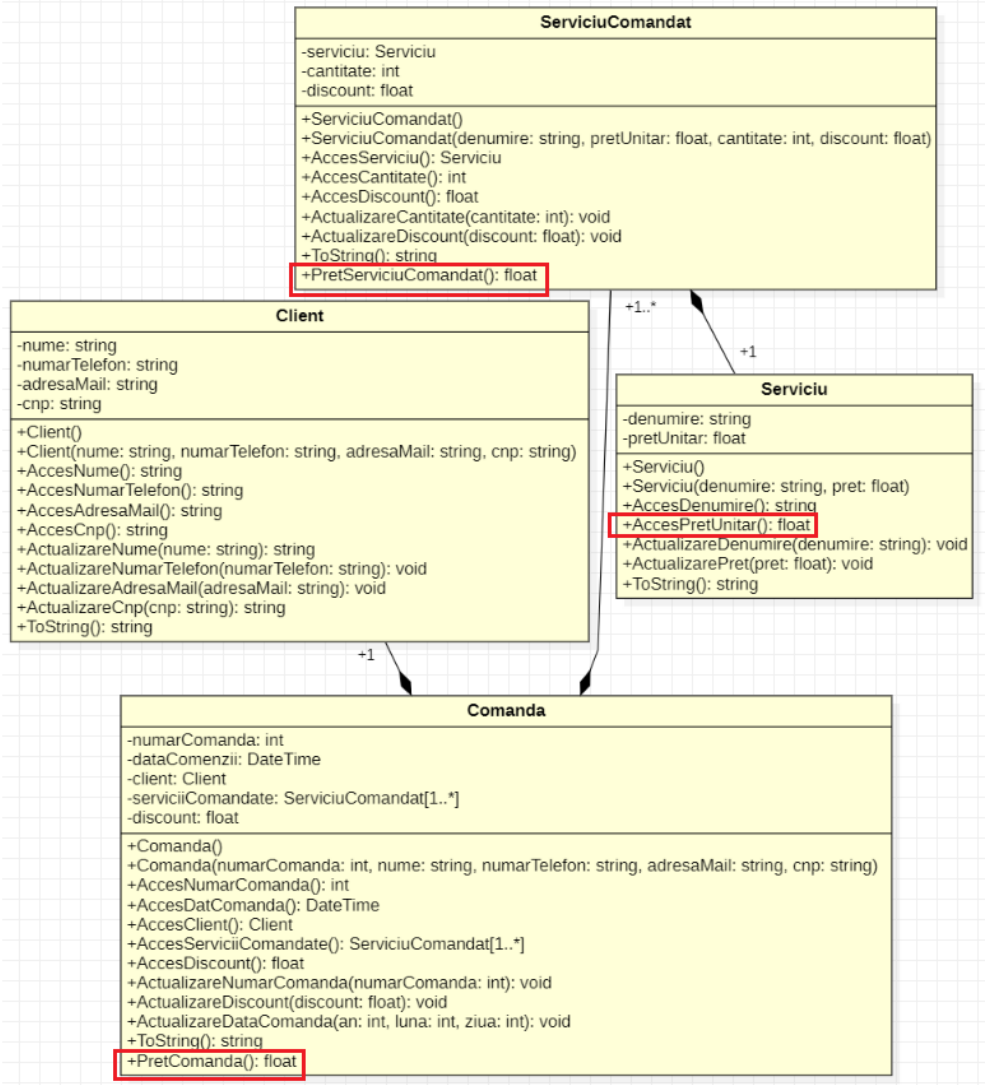

Figura 3.1. Diagrama de clase – INFORMATION EXPERT

Apoi clasa ***Comanda*** trebuie să conțină o metodă pentru calculul valorii comenzii, metodă ce va apela metoda ***PretServiciuComandat()*** pentru fiecare serviciu atașat comenzii. Implementarea acestei metode este următoarea:

```
public float PretComanda()
{
    float pret = 0;
    foreach (ServiciuComandat sc in this.serviciiComandate)
        pret += sc.PretServiciuComandat();
    pret -= pret * discount/100;
    return pret;
}
```

O altă decizie se referă la a stabili cine să fie responsabil de instanțierea obiectelor de tip ***ServiciuComandat***. Conform principiului ***Creator*** aceste tip de obiecte ar trebui să fie instanțiate de către clasa ***Comanda***, astfel fiind suportată o cuplare redusă între clase (figura 3.2).

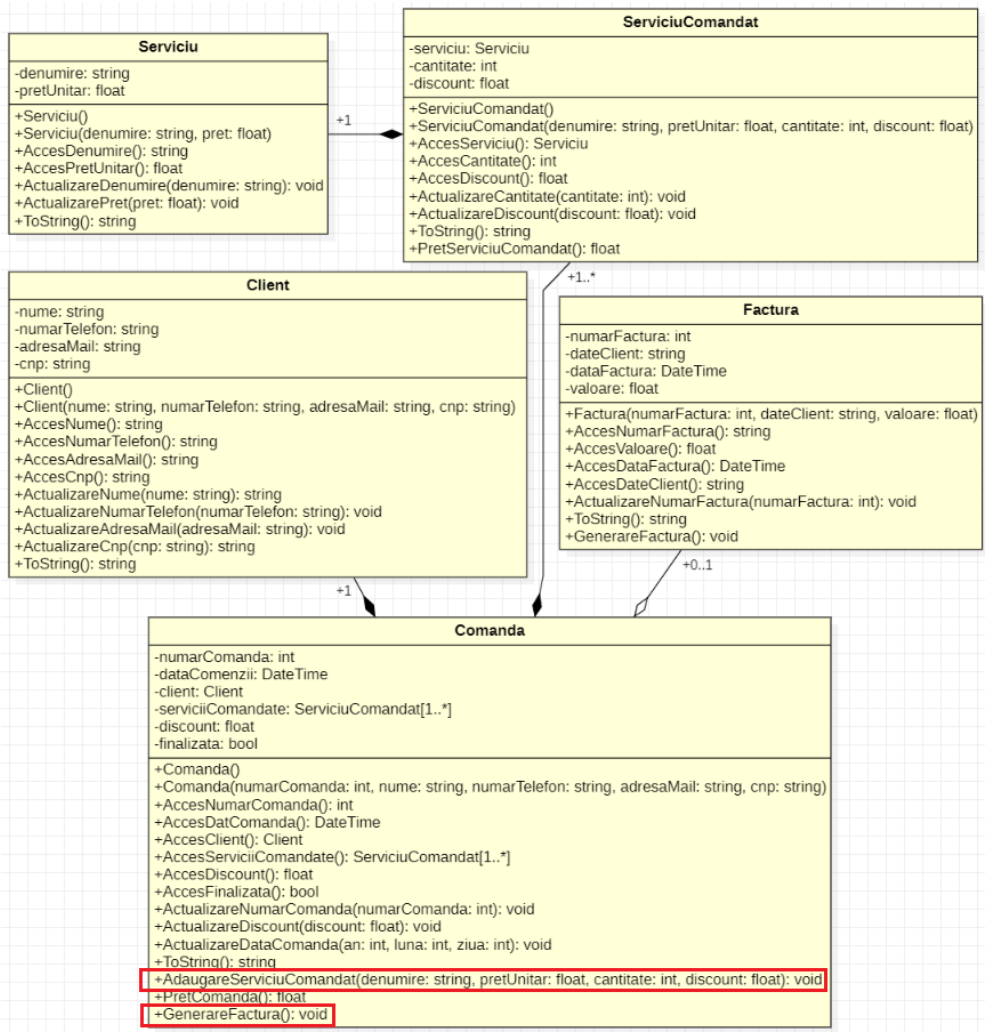

Figura 3.2. Diagrama de clase – CREATOR

În acest sens, clasei ***Comanda*** i se adaugă metoda ***AdaugareServiciuComandat()*** care primește ca parametrii informațiile despre noul serviciu, iar în corpul său instanțiază un nou obiect de tip ***ServiciuComandat*** și îl adaugă la lista serviciilor comenzii curente. Implementarea metodei este următoarea:

```
public void AdaugareServiciuComandat(string denumire, float
pretUnitar, int cantitate, float discount)
{
    ServiciuComandat sc = new ServiciuComandat(denumire, pretUnitar,
cantitate, discount);
    this.serviciiComandate.Add(sc);
}
```

În enunțul problemei, se specifică ca la momentul în care un angajat al tipografiei precizează că comanda solicitată este finalizată, să poată fi generată factura asociată comenzii (apoi factura poate fi fie tipărită, fie trimisă clientul pe adresa de email). În acest scop se proiectează o nouă clasă ***Factura***, iar clasei ***Comanda*** i se adaugă un atribut (finalizata) care să specifice starea comenzii (este sau nu finalizată).

Conform principiului ***Creator***, obiectele de tip ***Factura*** ar trebui să fie instanțiate de către clasa ***Comanda***. În acest scop se adaugă clasei ***Comanda*** metoda ***GenerareFactura()***:

```
public void GenerareFactura()
{
    this.finalizata = true;
    Factura factura = new Factura(this.numarComanda,
this.client.ToString(), this.PretComanda());
    factura.GenerareFactura();
        }
```

Opțiunile de crearea a unei comenzi, adăugare a unui serviciu comandat sau generarea unei facturi vor fi rezultatul apariției unor evenimente. Se pune problema de a stabili ce clasă ar trebui să aibă responsabilitatea gestinării acestor evenimente.

În acest sens, este utilă introducerea unei clase cu rol de control în clasa asociată interfeței utilizator și clasa ce corespunde modelului (în cazul exemplului ales – clasa ***Comanda***). Astfel se obține o nouă diagramă de clase prezentată în figura 3.3. Clasa ***TipografieControl*** va coordona toate evenimentele delegând altor obiecte (de tip ***Comanda***) toate activitițile ce trebuie să fie executate pentru fiecare eveniment.

Clasa ***TipografieUI*** are ca scop interacțiunea cu utilizatorii aplicației (angajații tipografiei): preluarea informaților introduse și afișarea rezultatelor (mesajelor).

Utilizând principiile GRASP se încearcă asigurarea unei cuplări slabe între clasele proiectate. În figura 3.4 este prezentată măsurarea cuplării între clasele proiectului.

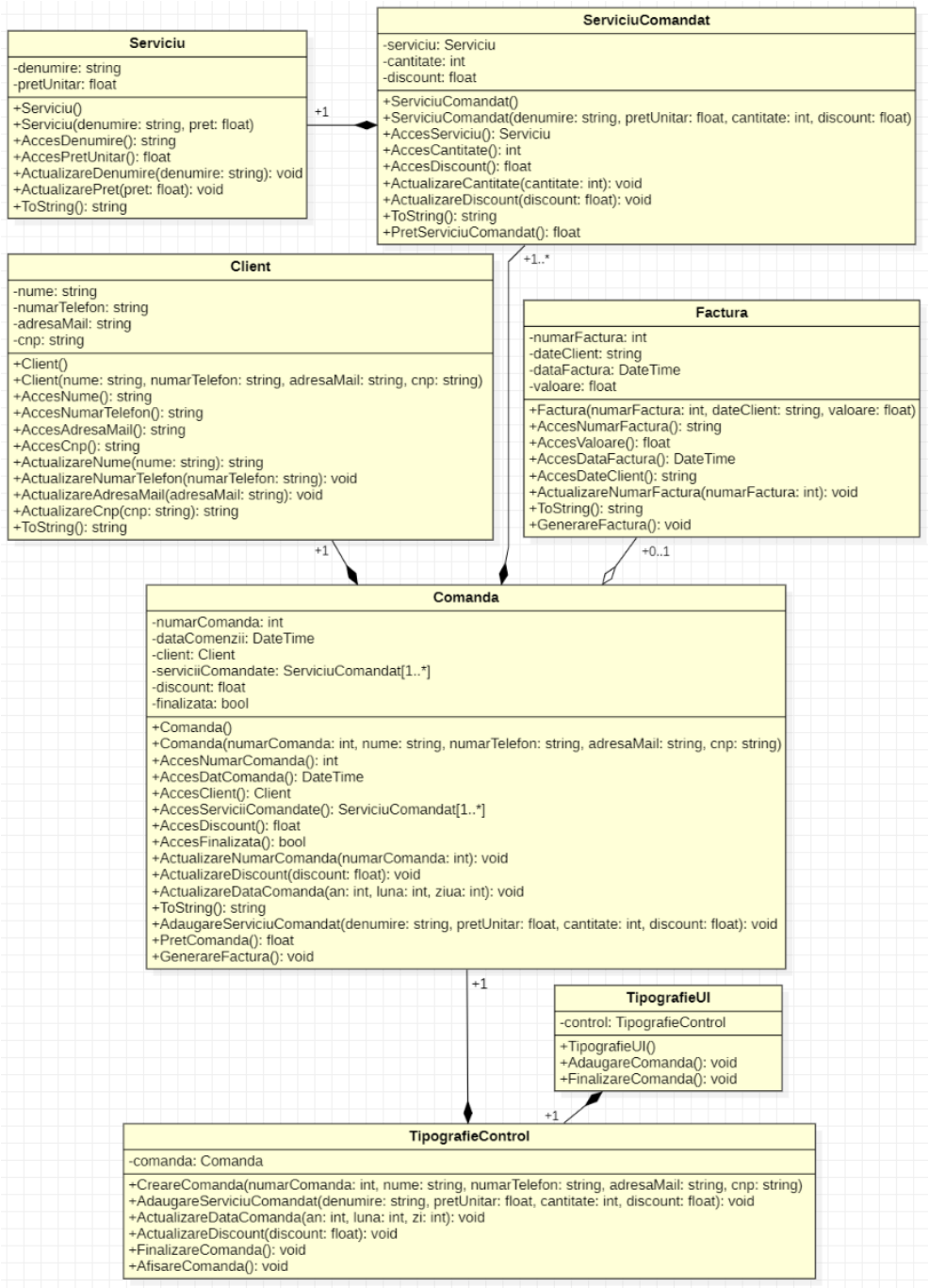

Figura 3.3. Diagrama de clase

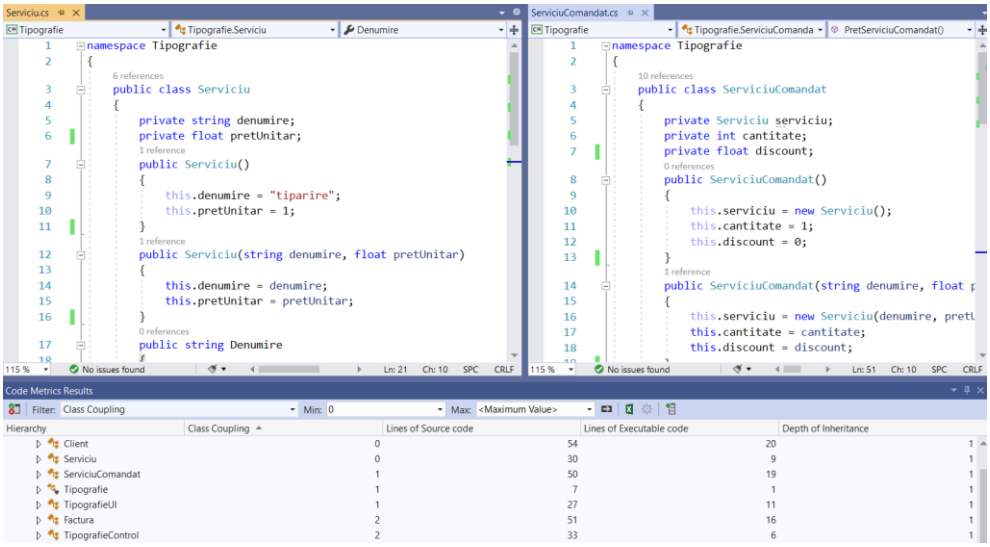

Figura 3.4. Măsurarea cuplării

Exercițiu 3.2. Dezvoltați, utilizând șablonul arhitectural MVP, o aplicație soft care să permită determinarea mediei aritmetice, mediei geometrice și mediei armonice a două numere reale.

Soluție:

Soluția prezentată utilizează varianta șablonului arhitectural MVP în care pachetele ***View*** și ***Model*** sunt decuplate (figura 3.5).

Pachetul ***Model*** cuprinde 4 clase: clasa abstractă ***Media*** și clasele: ***MediaAritmetica***, ***MediaGeometrica*** și ***MediaArmonica*** (figura 3.6). Pachetul ***View*** conține o clasă (***MediiGUI***) și o interfață (***IMedii***). ***Clasa MediiGUI,*** ce realizează interfața ***IMedii,*** are ca responsabilități:

- Gestionarea interacțiunilor cu utilizatorul;
- Gestionarea evenimentelor.

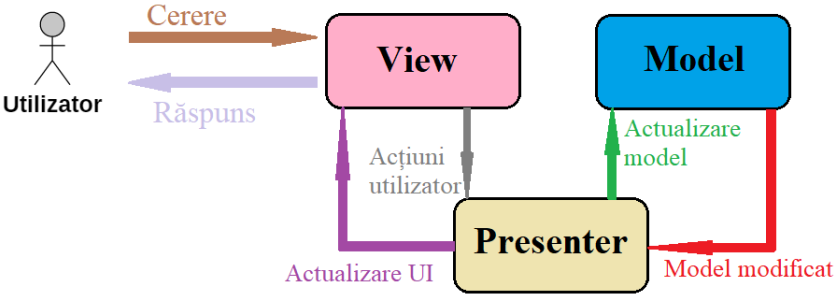

Figura 3.5. Arhitectura MVP

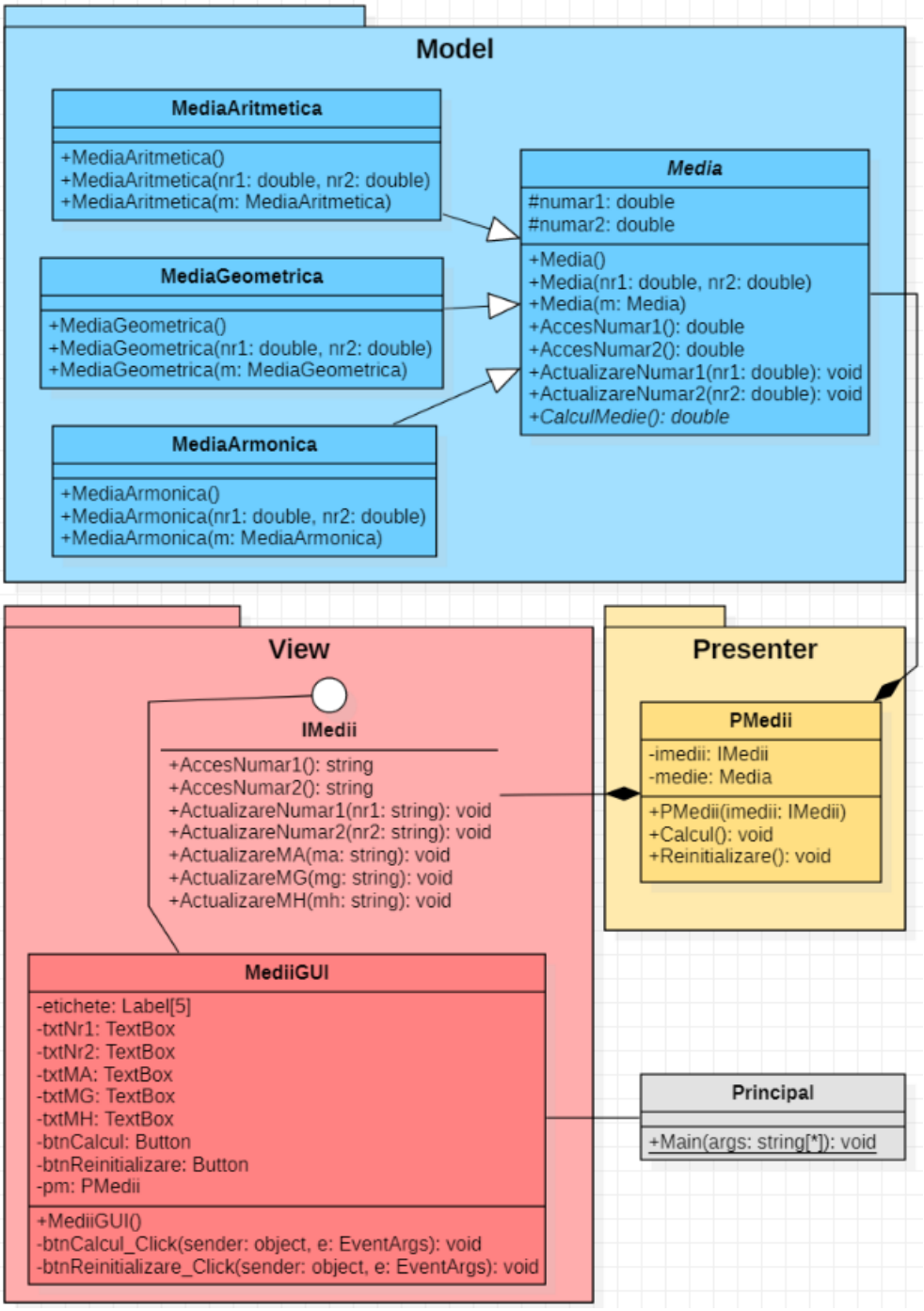

Figura 3.6. Diagrama de clase

Pachetul ***Presenter*** conține o clasă ***PMedii*** care are 2 atribute:

- ❖ Un obiect ce reprezintă modelul utilizat în rezolvarea problemei;
- ❖ Un obiect care realizează interfața ***IMedii*** pentru a comunica cu interfața grafică a aplicației.

Conform arhitecturii MVP crearea obiectelor este realizată în ordinea prezentată în diagrama de secvență din figura 3.7. Diagrama cuprinde și ordinea de apelare a metodelor pentru calculul și afișarea celor 3 medii.

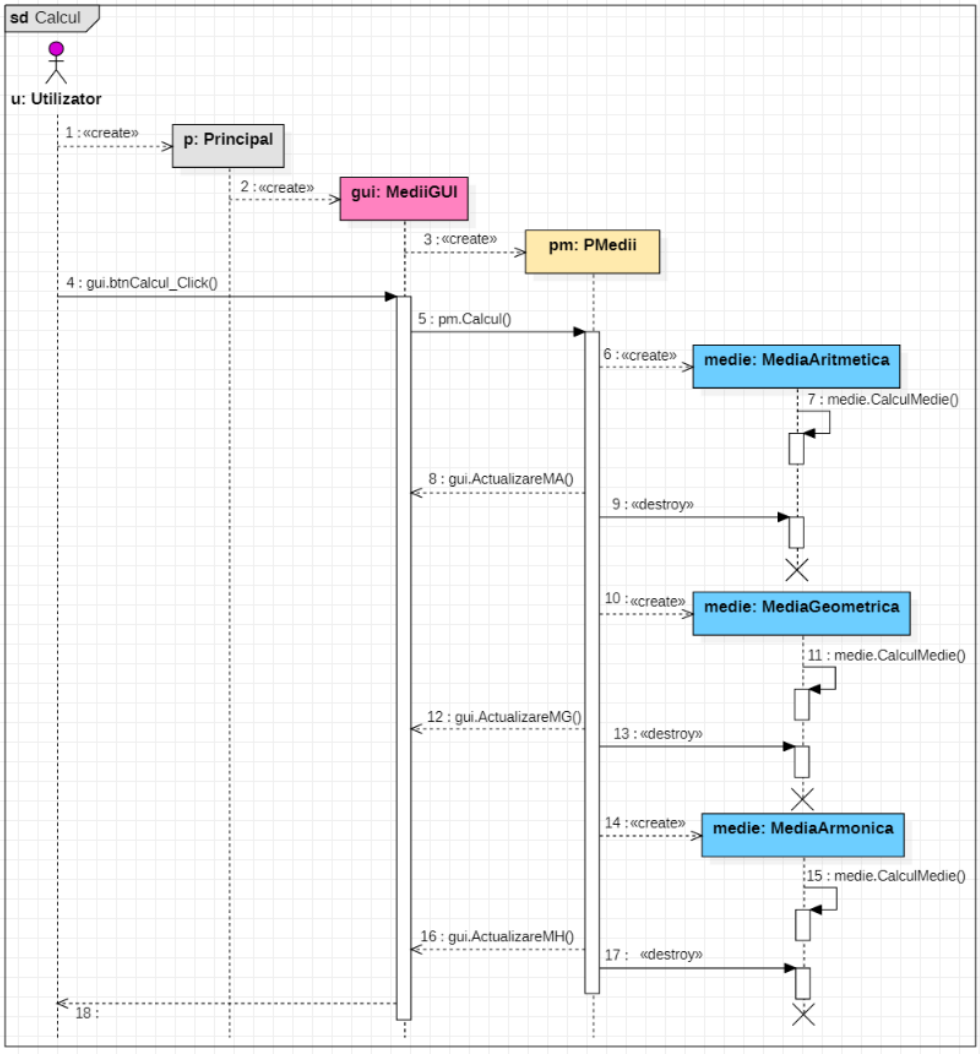

Figura 3.7. Diagrama de secvență

Clasa abstractă ***Media*** din pachetul ***Medii.Model*** are următoarea implementare:

```
namespace Medii.Model
{
    public abstract class Media
    {
        protected double numar1;
        protected double numar2;
        public Media()
        {
            this.numar1 = 1;
            this.numar2 = 1;
        }
        public Media(double nr1, double nr2)
        {
            this.numar1 = nr1;
            this.numar2 = nr2;
        }
        public Media(Media m)
        {
            this.numar1 = m.numar1;
            this.numar2 = m.numar2;
        }
        public double Numar1
        {
            get { return this.numar1; }
            set { this.numar1 = value; }
        }
        public double Numar2
        {
            get { return this.numar2; }
            set { this.numar2 = value; }
        }
        public abstract double CalculMedie();
    } }
```

Clasa ***MediaAritmetica*** are următoarea implementare:

```
namespace Medii.Model
{
    public class MediaAritmetica : Media
    {
        public MediaAritmetica() : base() { }
        public MediaAritmetica(double nr1, double nr2)
           : base(nr1, nr2) { }
        public MediaAritmetica(MediaAritmetica ma)
        {
            this.numar1 = ma.numar1;
            this.numar2 = ma.numar2;
        }
        public override double CalculMedie()
        {
            return (this.numar1 + this.numar2) / 2;
        }
    } }
```

Clasa ***MediaGeometrica*** din pachetul ***Medii.Model*** are următoarea implementare:

```
using System;
namespace Medii.Model
{
    public class MediaGeometrica : Media
    {
        public MediaGeometrica() : base() { }
        public MediaGeometrica(double nr1, double nr2)
          : base(nr1, nr2) { }
        public MediaGeometrica(MediaGeometrica ma)
        {
            this.numar1 = ma.numar1;
            this.numar2 = ma.numar2;
        }
        public override double CalculMedie()
        {
            return Math.Sqrt(this.numar1 * this.numar2);
        }
    }
}
```

Clasa ***MediaArmonica*** din pachetul ***Medii.Model*** are următoarea implementare:

```
namespace Medii.Model
{
    public class MediaArmonica : Media
    {
        public MediaArmonica() : base() { }
        public MediaArmonica(double nr1, double nr2)
          : base(nr1, nr2) { }
        public MediaArmonica(MediaArmonica ma)
        {
            this.numar1 = ma.numar1;
            this.numar2 = ma.numar2;
        }
        public override double CalculMedie()
        {
            return 2/(1/this.numar1 + 1/this.numar2);
        }
    }
}
```

Interfața ***IMedii*** din pachetul ***Medii.View*** are următoarea implementare:

```
namespace Medii.View
{
    public interface IMedii
    {
        string AccesNumar1();
        void ActualizareNumar1(string nr1);
        string AccesNumar2();
        void ActualizareNumar2(string nr2);
        void ActualizareMA(string ma);
        void ActualizareMG(string mg);
        void ActualizareMH(string mh);
    }
}
```

Clasa ***MediiGUI*** din pachetul ***Medii.View,*** ce reprezintă interfața grafică a aplicației (figura 3.8), are următoarea implementare:

```
using System;
using System.Windows.Forms;
using Medii.Presenter;
namespace Medii.View
{
    public partial class MediiGUI : Form, IMedii
    {
        PMedii pm =null;
        public MediiGUI()
        {
            InitializeComponent();
            pm = new PMedii(this);
        }
        public string AccesNumar1()
        {
            return this.txtNr1.Text;
        }
        public string AccesNumar2()
        {
            return this.txtNr2.Text;
        }
        public void ActualizareMA(string ma)
        {
            this.txtMA.Text = ma;
        }
        public void ActualizareMG(string mg)
        {
            this.txtMG.Text = mg;
        }
        public void ActualizareMH(string mh)
        {
            this.txtMH.Text = mh;
        }
        public void ActualizareNumar1(string nr1)
        {
            this.txtNr1.Text = nr1;
```

```
        }
        public void ActualizareNumar2(string nr2)
        {
            this.txtNr2.Text = nr2;
        }
        private void btnCalcul_Click(object sender, EventArgs e)
        {
            this.pm.Calcul();
        }
        private void btnReinitializare_Click(object sender,
EventArgs e)
        {
            this.pm.Reinitializare();
        }

    }
}
```

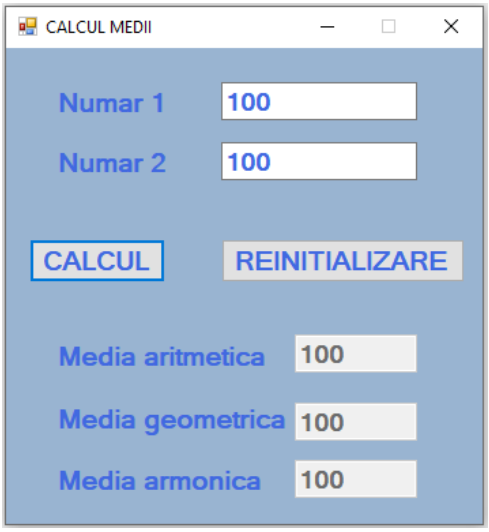

Figura 3.8. Interfața grafică utilizator

Clasa ***PMedii*** din pachetul ***Medii.Presenter*** are următoarea implementare:

```
using Medii.Model;
using Medii.View;
using System;
namespace Medii.Presenter
{
    public class PMedii
    {
        private IMedii imedii;
        private Media medie = null;
        public PMedii(IMedii imedii)
        {
```

```
            this.imedii = imedii;
        }
        public void Calcul()
        {
            try
            {
                double nr1, nr2, ma, mg, mh;
                nr1 = double.Parse(this.imedii.AccesNumar1());
                nr2 = double.Parse(this.imedii.AccesNumar2());
                this.medie = new MediaAritmetica(nr1, nr2);
                ma = this.medie.CalculMedie();
                this.imedii.ActualizareMA(ma.ToString());
                if (nr1 * nr2 >= 0)
                {
                    this.medie = new MediaGeometrica(nr1, nr2);
                    mg = this.medie.CalculMedie();
                    this.imedii.ActualizareMG(mg.ToString());
                }
                else
                    this.imedii.ActualizareMG("Nu se poate
calcula.");
                if (nr1 != 0 && nr2 != 0)
                {
                    this.medie = new MediaArmonica(nr1, nr2);
                    mh = this.medie.CalculMedie();
                    this.imedii.ActualizareMH(mh.ToString());
                }
                else
                    this.imedii.ActualizareMH("Nu se poate
calcula.");
            }
            catch (Exception)
            {
                this.Reinitializare();
            }
        }
        public void Reinitializare()
        {
            this.imedii.ActualizareNumar1("");
            this.imedii.ActualizareNumar2("");
            this.imedii.ActualizareMA("");
            this.imedii.ActualizareMG("");
            this.imedii.ActualizareMH("");
        }
    }
}
```

3.3 Exerciții propuse spre rezolvare

Exercițiu 3.3. Să se modifice diagrama de clase din figura 3.3 astfel încât să respecte șablonul arhitectural MVP.

Exercițiu 3.4. Să se modifice diagrama de clase din figura 1.2 (lucrare de laborator 1) astfel încât să respecte principiile GRASP și șablonul arhitectural MVP.