Lab 4 - Giurgiu Petru 1721B

Tema 1

Arhitectura

app1

CustomCompare.cs → este o clasa care implementeaza interfata IComparer prin care noi implementam sortarea elementelor

ProdusAbstractMgr.cs → clasa ProdusAbstractMgr este clasa care se ocupa de gestionarea claselor ProduseMgr si ServiciuMgr continand un ArrayList unde vor fi introduse produsele si serviciile si o metoda Write2Console unde se sorteaza elementele ArrayList-ului cu ajutorul clasei CustomCompare iar dupa se afiseaza in consola

ProduseMgr.cs → clasa ProduseMgr mosteneste clasa ProdusAbstractMgr, aceasta clasa se ocupa cu adaugarea de produse in ArrayList-ul elemente din clasa ProdusAbstractMgr avand campul nrProduse(reprezentand numarul de produse pe care utilizatorul vrea sa le adauge in ArrayList-ul elemente) care primeste o anumita valoare prin constructorul clasei atunci cand aceasta este instantiata, in aceasta clasa mai avem metoda AdaugareServicii care se ocupa strict cu adaugarea serviciiilor cerand utilizatorului datele necesare pentru instantierea unei clase de tipul Produs, verificand daca aceasta instanta mai exista in ArrayList-ul elemente si doar daca nu exista atunci o adauga

Program.cs → clasa Program este entry point-ul in aplicatie unde cerem utilizatorului numarul de produse si servicii pe care vrea sa le introduca, instantiand clasele ProuseMgr si SeviciuMgr cu acestea, dupa care apelam metodele AdaugareProduse si AdaugareServicii ale celor 2 clase instantiate, iar la finalul clasei gasim apelarea metodei Write2Console care ne afiseaza elementele adaugate

ServiciuMgr.cs → clasa ServiciuMgr mosteneste clasa ProdusAbstractMgr, aceasta clasa se ocupa cu adaugarea de servicii in ArrayList-ul elemente din clasa ProdusAbstractMgr avand campul nrServicii(reprezentand numarul de servicii pe care utilizatorul vrea sa le adauge in ArrayList-ul elemente) care primeste o anumita valoare prin constructorul clasei atunci cand aceasta este instantiata, in aceasta clasa mai avem metoda AdaugareServicii care se

ocupa strict cu adaugarea serviciiilor cerand utilizatorului datele necesare pentru instantierea unei clase de tipul Serviciu, verificand daca aceasta instanta mai exista in ArrayList-ul elemente si doar daca nu exista atunci o adauga

entitati

Produs.cs → clasa Produs mosteneste clasa ProdusAbstract avand in plus campul producator si metoda aferenta acestuia de accesare si setare si face override metodei Descriere afisand proprietatile produsului

ProdusAbstract.cs → clasa ProdusAbstract este clasa de baza de tip abstract cu campurile id, nume si codIntern, proprietatile aferente acestora care ne permit sa accesam si sa setam campurile clasei si o metoda Descriere

Serviciu.cs → clasa Serviciu mosteneste clasa ProdusAbstract, facand doar override metodei Descriere in care se afiseaza proprietatile serviciului

```
// entitati/ProdusAbstract.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace entitati
{
    public abstract class ProdusAbstract
    {
        long id;
        String nume;
        String codIntern;

        protected ProdusAbstract(long id, string nume, string codIntern)
        {
            this.Id = id;
            this.Nume = nume;
            this.CodIntern = codIntern;
        }

        public long Id { get => id; set => id = value; }
        public string Nume { get => nume; set => nume = value; }
        public string CodIntern { get => codIntern; set => codIntern = value; }

        public abstract String Descriere();
    }
}
```

```
// entitati/Produs.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace entitati
{
    public class Produs : ProdusAbstract // mostenim clasa ProdusAbstract
    {
        string producator;
        public Produs(long id, string nume, string codIntern, string producator) :
base( id, nume, codIntern)
            {
            this.Producator = producator;
            }

        public string Producator { get => producator; set => producator = value; }

        public override string Descriere()
        {
            return $@"Produs:
Nume: {Nume},
Id: {Id},
Cod Intern: {CodIntern}
Producator: {Producator}
";
        }
    }
}
```

```
// entitati/Serviciu.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace entitati
{
    public class Serviciu : ProdusAbstract // mostenim clasa ProdusAbstract

    {
        public Serviciu(long id, string nume, string codIntern) : base(id, nume, c
odIntern) {}

        public override string Descriere()
        {
            return $@"Serviciu:
Nume: {Nume},
Id: {Id},
```

```
Cod Intern: {CodIntern}
";
        }
    }
}
```

```
// app1/CustomComparer.cs
using entitati;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    class CustomComparer : IComparer // implementam interfata IComparer
    {
        Comparer _comparer = new Comparer(System.Globalization.CultureInfo.Current
Culture);

        public int Compare(object x, object y) // aici se primeste ca parametri el
ementele ArrayList-ului in cazul nostru, cate 2 la un moment dat, acestea fiind co
mparate dupa nume
        {
            return _comparer.Compare(((ProdusAbstract)(x)).Nume, ((ProdusAbstract)
(y)).Nume);
        }
    }
}
```

```
// app1/ProdusAbstractMgr.cs
using entitati;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public abstract class ProdusAbstractMgr
    {
        protected static ArrayList elemente = new ArrayList();
```

```
        public void Write2Console()
        {
            elemente.Sort(new CustomComparer()); // sortam elementele cu ajutorul
 clasei CustomComparer
            foreach(ProdusAbstract prod in elemente)
            {
                Console.WriteLine(prod.Descriere());
            }
        }
    }
}
```

```
// app1/ProduseMgr.cs
using entitati;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public class ProduseMgr : ProdusAbstractMgr // mostenim clasa ProdusAbstractMg
r
    {
        int nrProduse;
        public ProduseMgr(int nrProduse)
        {
            this.nrProduse = nrProduse;
        }

        public void AdaugareProduse ()
        {
            int cnt = 0; // cnt care ne ajuta sa vedem cate elemente am introdus i
n ArrayList
            while ( cnt < nrProduse)
            {
                Console.WriteLine("Id produs: ");
                long id = int.Parse(Console.ReadLine());
                Console.WriteLine("Nume produs: ");
                string nume = Console.ReadLine();
                Console.WriteLine("Cod intern producator: ");
                string codIntern = Console.ReadLine();
                Console.WriteLine("Producator: ");
                string producator = Console.ReadLine();
                Produs prod = new Produs(id, nume, codIntern, producator);

                bool exist = false;

                foreach (ProdusAbstract obj in elemente)
                {
                    if (obj.Id.Equals(prod.Id) && obj.Nume.Equals(prod.Nume) && ob
```

```
j.CodIntern.Equals(prod.CodIntern)) // verificam daca produsul instantiat exista d
eja in ArrayList-ul elemente
                    {
                        exist = true;
                    }
                }

                if (exist) // daca exista nu este adaugat in ArrayList
                {
                    Console.WriteLine("Produs exista deja!");
                    continue;
                }
                else
                {
                    elemente.Add(prod); // daca nu exista este adaugat la sfarsitu
l ArrayList-ului elemente
                    cnt++;
                    Console.WriteLine("Produs adaugat!");
                }
            }
        }
    }
}
```

```
// app1/ServiciuMgr.cs
using entitati;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public class ServiciuMgr : ProdusAbstractMgr // mostenim clasa ProdusAbstractM
gr
    {
        int nrServicii;
        public ServiciuMgr(int nrServicii)
        {
            this.nrServicii = nrServicii;
        }
        public void AdaugareServicii()
        {
            int cnt = 0;
            while (cnt < nrServicii)
            {
                Console.WriteLine("Id serviciu: ");
                long id = int.Parse(Console.ReadLine());
                Console.WriteLine("Nume serviciu: ");
                string nume = Console.ReadLine();
                Console.WriteLine("Cod intern serviciu: ");
                string codIntern = Console.ReadLine();
```

```
                Serviciu serv = new Serviciu(id, nume, codIntern);

                bool exist = false;

                foreach(ProdusAbstract obj in elemente)
                {
                    if(obj.Id.Equals(serv.Id) && obj.Nume.Equals(serv.Nume) && ob
j.CodIntern.Equals(serv.CodIntern)) // verificam daca serviciul instantiat exista
 deja in ArrayList-ul elemente
                    {
                        exist = true;
                    }
                }


                if (exist) // daca exista nu este adaugat in ArrayList
                {
                    Console.WriteLine("Serviciul exista deja!");
                    continue;
                }
                else
                { // daca nu exista este adaugat la sfarsitul ArrayList-ului eleme
nte
                    elemente.Add(serv);
                    cnt++;
                    Console.WriteLine("Serviciu adaugat!");
                }
            }
        }
    }
}
```

```
// app1/Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using entitati;

namespace app1
{
    class Program
    {
        static void Main(string[] args)

        {
            Console.WriteLine("Nr. produse: ");
            int nrProduse = int.Parse(Console.ReadLine());

            Console.WriteLine("Nr. servicii: ");
            int nrServicii = int.Parse(Console.ReadLine());
```

```
            ProduseMgr produseManager = new ProduseMgr(nrProduse);
            ServiciuMgr seviciiManager = new ServiciuMgr(nrServicii);

            produseManager.AdaugareProduse();
            seviciiManager.AdaugareServicii();

            produseManager.Write2Console();
        }
    }
}
```

Tema 2

Avem aceeasi arhitectura ca la tema 1 doar cu mici modificari:

app1

ColectieTipizata.cs → in clasa ColectieTipizata mostenim clasa System.Collections.CollectionBase, unde avem mai multe metode, **Adauga**(care adauga elemente de tipul ProdusAbstract la sfarsitul List-ei mostenite din clasa System.Collections.CollectionBase), metoda **Extrage**(care sterge elemente din lista mostenita din clasa System.Collections.CollectionBase, pe baza id-ului pe care il dam noi ca parametru acestei metode), metoda **Contine**(care returneaza valoarea true in cazul in care in lista mostenita exista un element cu codulIntern la fel ca cel pe care il dam ca parametru acestei metode, in cazul in care nu exista, ne returneaza valoarea false daca nu exista un astfel de element), metoda **Sortare**(care sorteaza InnerList-ul pe baza clasei CustomComparer care implementeaza interfata IComparer)

CustomCompare.cs

ProdusAbstractMgr.cs → sortarea elementelor se face prin metoda Sortare a clasei ColetieTipizata

ProduseMgr.cs → vom adauga elemente in ColectieTipizata prin metoda Adaugare, nu prin Add cum faceam la ArrayList

Program.cs

ServiciuMgr.cs → vom adauga elemente in ColectieTipizata prin metoda Adaugare, nu prin Add cum faceam la ArrayList

entitati

Produs.cs

ProdusAbstract.cs

Serviciu.cs

Celelate clase raman ca si la Tema 1

```
// app1/ColectieTipizata.cs
using entitati;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public class ColectieTipizata : System.Collections.CollectionBase // mostenim clas
a System.Collections.CollectionBase
    {

        public void Adauga(ProdusAbstract produs)
        {

            this.List.Add(produs); // adaugam produsul de tipul ProdusAbstract care il
dam ca si parametru List-ei, care o mostenim din clasa System.Collections.CollectionBa
se
        }
        public void Extrage(long id)
        {
            this.List.Remove(id); // sterge elemente din lista mostenita din clasa Sys
tem.Collections.CollectionBase, pe baza id-ului pe care il dam noi ca parametru aceste
i metode
        }
        public bool Contine(string codIntern)
        {
            String tit = (String)codIntern;
            return this.List.Contains(tit); //  returneaza valoarea true in cazul in c
are in lista mostenita exista un element cu codulIntern la fel ca cel pe care il dam c
a parametru acestei metode, in cazul in care nu exista, ne returneaza valoarea false d
aca nu exista un astfel de element
        }
```

```
        public ProdusAbstract this[string codIntern]
        {
            get
            {
                if
                (this.List.Contains(codIntern))
                    return
                    (ProdusAbstract)(this.List[int.Parse(codIntern)]);
                else return null;
            }
        }

        public void Sortare()
        {
            this.InnerList.Sort(new CustomComparer()); // sorteaza InnerList-ul pe baz
a clasei CustomComparer care implementeaza interfata IComparer
        }
    }

}
```

```
// app1/ProdusAbstractMgr
using entitati;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public abstract class ProdusAbstractMgr
    {
        protected static ColectieTipizata elemente = new ColectieTipizata();

        public void Write2Console()
        {
            elemente.Sortare(); // sortarea se face pe baza metodei Sortare din cadrul
clasei ColectieTipizata
            foreach(ProdusAbstract prod in elemente)
            {
                Console.WriteLine(prod.Descriere());
            }
        }
    }
}
```

```
// app1/ProduseMgr.cs

using entitati;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public class ProduseMgr : ProdusAbstractMgr
    {
        int nrProduse;
        public ProduseMgr(int nrProduse)
        {
            this.nrProduse = nrProduse;
        }

        public void AdaugareProduse ()
        {
            int cnt = 0;
            while ( cnt < nrProduse)
            {
                Console.WriteLine("Id produs: ");
                long id = int.Parse(Console.ReadLine());
                Console.WriteLine("Nume produs: ");
                string nume = Console.ReadLine();
                Console.WriteLine("Cod intern producator: ");
                string codIntern = Console.ReadLine();
                Console.WriteLine("Producator: ");
                string producator = Console.ReadLine();
                Produs prod = new Produs(id, nume, codIntern, producator);

                bool exist = false;

                foreach (ProdusAbstract obj in elemente)
                {
                    if (obj.Id.Equals(prod.Id) && obj.Nume.Equals(prod.Nume) && obj.Co
dIntern.Equals(prod.CodIntern))
                    {
                        exist = true;
                    }
                }

                if (exist)
                {
                    Console.WriteLine("Produs exista deja!");
                    continue;
                }
                else
                {
                    elemente.Adauga(prod); // adaugam prod de tipul Produs ColectieTip
izate elemente prin metoda customizata Adauga
                    cnt++;
                    Console.WriteLine("Produs adaugat!");
```

```
                }
            }
        }

        public void AfisareProduse()
        {
            foreach (ProdusAbstract produs in elemente)
            {
                Console.WriteLine(produs.Descriere());
            }
        }
    }
}
```

```
// app1/ServiciuMgr.cs
using entitati;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app1
{
    public class ServiciuMgr : ProdusAbstractMgr
    {
        int nrServicii;
        public ServiciuMgr(int nrServicii)
        {
            this.nrServicii = nrServicii;
        }
        public void AdaugareServicii()
        {
            int cnt = 0;
            while (cnt < nrServicii)
            {
                Console.WriteLine("Id serviciu: ");
                long id = int.Parse(Console.ReadLine());
                Console.WriteLine("Nume serviciu: ");
                string nume = Console.ReadLine();
                Console.WriteLine("Cod intern serviciu: ");
                string codIntern = Console.ReadLine();
                Serviciu serv = new Serviciu(id, nume, codIntern);

                bool exist = false;

                foreach(ProdusAbstract obj in elemente)
                {
                    if(obj.Id.Equals(serv.Id) && obj.Nume.Equals(serv.Nume) && obj.Cod
Intern.Equals(serv.CodIntern))
                    {
                        exist = true;
                    }
```

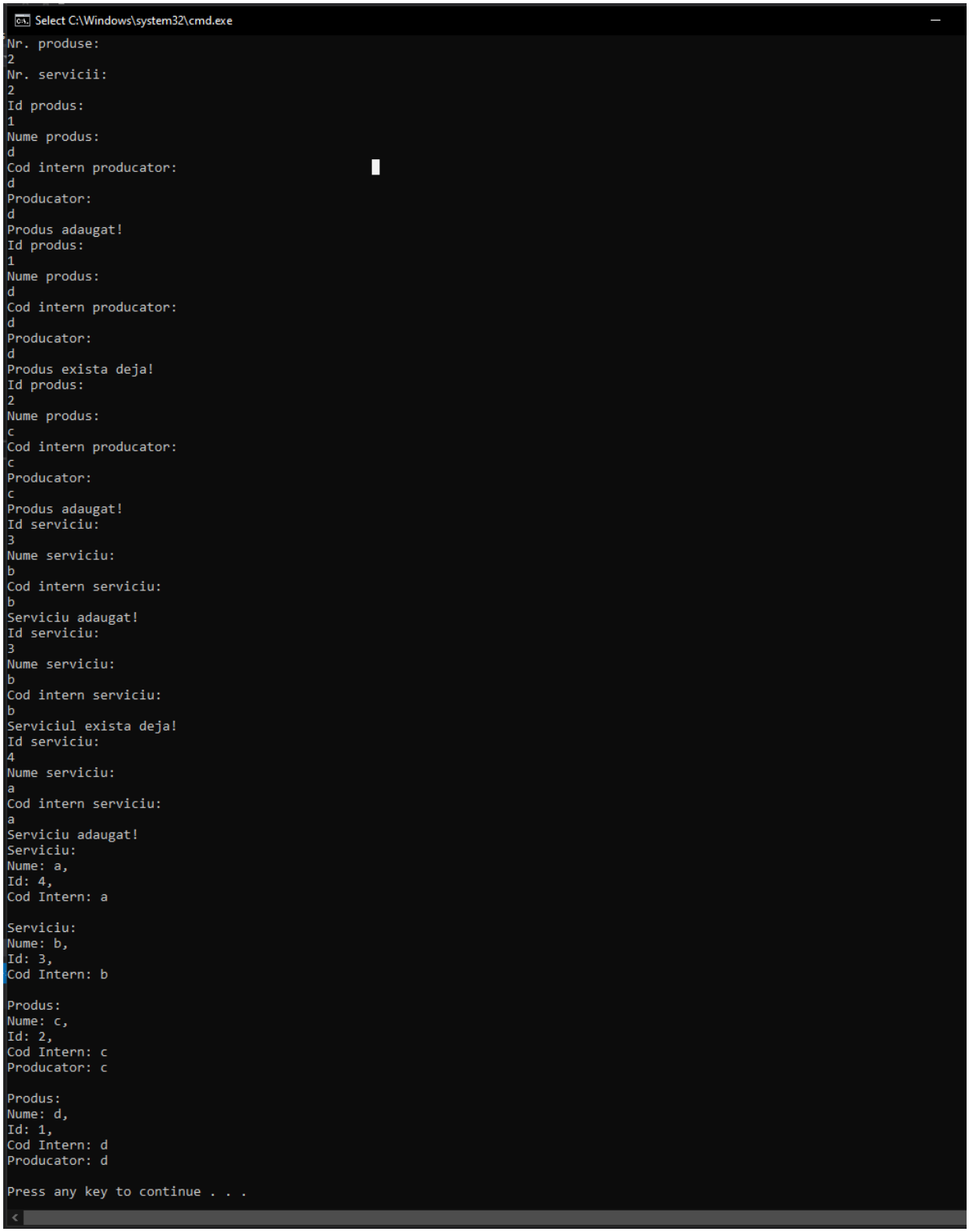
```
                }


                if (exist)
                {
                    Console.WriteLine("Serviciul exista deja!");
                    continue;
                }
                else
                {
                    elemente.Adauga(serv); // adaugam serv de tipul Serviciu ColectieT
ipizate elemente prin metoda customizata Adauga
                    cnt++;
                    Console.WriteLine("Serviciu adaugat!");
                }
            }
        }

        public void AfisareServicii()
        {
            foreach (ProdusAbstract serviciu in elemente)
            {
                Console.WriteLine(serviciu.Descriere());
            }
        }
    }
}
```

Diferenta dintre implementarea cu ArrayList si cea cu ColectieTipizata este ca adaugarea elementelor in ArrayList se face prin metoda implicita a acestuia Add, iar la ColectiaTipizata a trebuit sa implementam noi o metoda de Adaugare customizata. ArrayList-ul perimite adaugarea unui obiect de orice tip, iar ColectiaTipizat doar obiect de tip ProdusAbstract. Se poate considera un dezavantaj pentru arrayList. performata mai scazuta datorita conversiilor de tip, pot aparea erori.

Executia programului:


Select C:\Windows\system32\cmd.exe
Nr. produse:
2
Nr. servicii:
2
Id produs:
1
Nume produs:
d
Cod intern producator:
d
Producator:
d
Produs adaugat!
Id produs:
1
Nume produs:
d
Cod intern producator:
d
Producator:
d
Produs exista deja!
Id produs:
2
Nume produs:
c
Cod intern producator:
c
Producator:
c
Produs adaugat!
Id serviciu:
3
Nume serviciu:
b
Cod intern serviciu:
b
Serviciu adaugat!
Id serviciu:
3
Nume serviciu:
b
Cod intern serviciu:
b
Serviciul exista deja!
Id serviciu:
4
Nume serviciu:
a
Cod intern serviciu:
a
Serviciu adaugat!
Serviciu:
Nume: a,
Id: 4,
Cod Intern: a
Serviciu:
Nume: b,
Id: 3,
Cod Intern: b
Produs:
Nume: c,
Id: 2,
Cod Intern: c
Producator: c
Produs:
Nume: d,
Id: 1,
Cod Intern: d
Producator: d
Press any key to continue . . .