Laborator01 - Operaţii cu mulţimi

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

namespace ConsoleApplication11
{
    class Multime
    {
        static int dimMax = 100;
        int[] v;
        int length; //numarul de elemente

        public Multime() // constructor implicit
        {
            v = new int[dimMax];
            length = 0;
        }

        public Multime(Multime M) //constructor de copiere
        {
            v = new int[dimMax];
            for (int i = 0; i < M.length; i++)
                v[i] = M.v[i];
            this.length = M.length;
        }

        public Multime(int[] v,int n) // constructor de initializare
        {
            this.v = new int[dimMax];
            int min = (v.Length < n) ? v.Length : n;  //min>dimMax????
            min = (min < dimMax) ? min : dimMax;
            for (int i = 0; i < min; i++)
                this.v[i] = v[i];
            length = min;
        }

        //Accesori
        public int Length
        {
            get { return length; }
            set { length = value; }
        }

        public static int DimMax
        {
            get { return dimMax;}
            set { dimMax = value;}
        }

        //Iterator
```

```
        //Multime M, M[i] -> v[i]
        public int this[int i]
        {
            get { return v[i];}
            set { v[i] = value;}
        }

        public bool Exista(int x)
        {
            for (int i = 0; i < length; i++)
                if (v[i] == x)
                    return true;
            return false;
        }

        public bool Full()
        {
            return length == dimMax;
        }

        public bool Empty()
        {
            return length == 0;
        }

        public void Add(int x)
        {
            if (!Exista(x))
            //{
            //  v[length] = x;
            //  length++;
            // }
                v[length++]=x;
            // v[0],...., v[length-1],x
        }

        public void Delete(int x)
        {
            for (int i = 0; i < length; i++)
                if (v[i] == x)
                {
                    v[i] = v[length - 1];
                    length--;
                }
                //1 2 3 4 5 6 7 8 9
                //1 2 9 4 5 6 7 8 9
        }

        public override string ToString()
        //{ 1, 2, 3, 4}
        {
            string s = "{";
            for (int i = 0; i < length; i++)
                s += v[i] + ", ";
            s += "\b\b }";
            return s;
        }
```

```
        //supraincarcarea operatorilor
        //C=A+B, reuniunea dintre A si B
        public static Multime operator+(Multime A, Multime B)
        {
            Multime C = new Multime(A); // apel constructor de copiere
            for (int i = 0; i < B.length; i++)
                if (!A.Exista(B[i]))   //iterator  B.v[i]=B[i]
                    C.Add(B[i]);
            return C;
        }
        // operator*   intersectia dintre A si B
        // operator-   A-B
    }
    class Program
    {
        static void Main(string[] args)
        {
            int[] a = { 10, 20, 30, 40, 50 };
            Multime A = new Multime(a,5);  // constructorul de initializare
            int[] b = { 100, 200, 303, 400 };
            Multime B = new Multime(b,7);
            Multime C = A + B;
            Console.WriteLine("{0} + {1} = {2}", A,B,C); // apel automat al
metodei ToString

            Console.WriteLine("{0}", C[2]);  //iterator C.v[2]
            Console.WriteLine("{0}", A.ToString());

            //de apelat toate metodele definite in clasa Multime
            Console.ReadKey();
        }
    }
}
```