Curs04

(1) Tipuri generice (parametrizate) in C#
(2) Implementarea interfetelor

(1) Un tip generic este o clasa/interfata care permite parametrizarea anumitor date astfel incat acestea, la momentul executiei pot fi substituite cu mai multe tipuri specifice.

Sunt utilizate de obicei in containere (colectii de obiecte).

De exemplu: List (using System.Collections.Generic) este o colectie de obiecte de tip "lista" care are un parametru generic "T". Acesta poate fi inlocuit cu orice de tip.

List: o colectie de numere intregi
List o colectie de siruri de caractere
List o colectie de produse
...

Se pot implementa tipuri generice custom, definite de programator.

```
public class MyContainer<T> //container de obiecte de tip T
{
    List<T> _objects;

    //ctor
    public MyContainer()
    {
        _objects = new List<T>();
    }
}
```

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

namespace GenericApp
{
    //stiva generica de obiecte (poate stoca obiecte de "orice tip")
    public class MyGenericStack<T>
    {
        private T[] _elements; //suportul pentru elementele stivei de tip generic
T

        public int Size { get; private set; }

        public MyGenericStack()
        {
            _elements = new T[0]; //initializarea unui "array" de tip T cu 0
componente
        }
```

```
        public MyGenericStack<T> Add(T newElem)
        {
            T[] _temp = new T[Size + 1]; //aloc un nou array in care "sa incapa"
si newElem
            for(int i = 0; i < _elements.Length; i++)
            {
                _temp[i] = _elements[i];
            }

            _temp[Size] = newElem; //inseram newElem pe ultima pozitie
            _elements = _temp; //schimbam referinta catre noul "array" temp
            Size++;

            return this; //this este referinta catre obiectul curent
        }

        //scoatem elementul din varful stivei
        public T Remove()
        {
            if(Size == 0)
            {
                //varianta 1: generam o exceptie!
                throw new Exception("Stack is empty!"); //se genereaza o exceptie
daca se utilizeaza metoda Remove
                //pe o stiva goala

                //varianta 2
                //return default(T); operatorul "default" se utilieaza pentru un
tip generic T si returneaza valoare
                //default a tipului respectiv; de exemplu, 0 pentru int, null
pentru "Product" si orice tip referinta, false pentru bool
            }

            T[] _temp = new T[Size - 1]; //aloc un nou array in care "sa incapa"
si newElem

            for (int i = 0; i < _elements.Length-1; i++)
            {
                _temp[i] = _elements[i];
            }

            T result = _elements[Size - 1];
            _elements = _temp; //schimbam referinta catre noul "array" temp
            Size--;

            return result;
        }

        //afisare stiva la consola
        public void DisplayInfo()
        {
            if(Size == 0)
            {
                Console.WriteLine("[]");
                return;
            }

            //Console.WriteLine("[");
```

```
            //for(int i = 0; i < Size - 1; i++)
            //{
            //    Console.Write(_elements[i] + ", ");
            //}

            //Console.WriteLine(_elements[Size - 1] + "]");


            Console.WriteLine($"[{string.Join(",", _elements)}]");
        }
    }
}
```

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

namespace GenericApp
{
    class Program
    {
        static void Main(string[] args)
        {
            //declararea si definirea obiectelor de tipuri generice se face
substituind tipul generic cu
            //un tip concret
            MyGenericStack<Book> books = new MyGenericStack<Book>(); //acum,
compilatorul C# peste tot pe unde vede
            //T va inlocui cu Book

            books
                .Add(new Book()
            {
                Code = "1212",
                Title = "Cei trei muschetari",
                Author = "Alexandre Dumas",
                Year = 1844
            })
                .Add(new Book()
            {
                Code = "1250",
                Title = "Dupa douazeci de ani",
                Author = "Alexandre Dumas",
                Year = 1864
            });

            books.DisplayInfo();

            var b = books.Remove(); //elimin ultima carte

            books.DisplayInfo();

            Console.WriteLine(b);
            books.Remove();
            books.DisplayInfo(); //stiva goala
```

```
            //Scoaterea dintr-o stiva goala (tratarea exceptiilor in C#)

            try
            {
                books.Remove();
            }
            catch(Exception ex)
            {
                Console.WriteLine($"You cannot remove elements from empty stack!
error: {ex.Message}");
            }

            Console.ReadKey();
        }
    }
}
```

Constrangeri asupra tipurilor generice

Daca dorim sa ordonam obiectele din stiva???
Fiind de tip generic T, nu stim ce inseamna compararea intre doua obiecte de tip T
atunci impunem conditia tipurile T sa implementeze interfata IComparable

cum se procedeaza: where T:IComparable

```
 public class Book : IComparable
    {
        public string Code { get; set; }
        public string Title { get; set; }
        public int Year { get; set; }
        public string Author { get; set; }

        public int CompareTo(object obj)
        {
            var o = obj as Book;

            return Code.CompareTo(o.Code);

        }

        public override string ToString()
        {
            return $"[{Code}; {Title}; {Year}; {Author}]";
        }
    }
```