# Curs03

Introducere in C#

-clase
-metode
-proprietati
-compararea obiectelor
-reprezentare ca string

Limbaj pur orientat pe obiecte.

```
public class Product
  {
      private int _id;
      private string _name;
      private double _price;


      //getters & setters
      //public string GetName()
      //{
      //    return _name;
      //}

      //public void SetName(string name)
      //{
      //    _name = name;
      //}

      //proprietate de tip read/write
      public string Name
      {
          //getter
          get
          {
              return _name;
          }
          //setter
          private set
          {
              _name = value;
          }
      }

      //poate fi read-only: lipseste sectiunea set!!


      //constructori
      public Product()
      {
          _id = 1;
          Name = "default";
          _price = 100;
```

```
        }

        public Product(int id, string name, double price)
        {
            _id = id;
            _name = name;
            _price = price;
        }
    }
```

Initializarea obiectelor in C# se poate face:

(1) prin intermediul constructorilor

(2) prin intermediul proprietatilor publice (get;set)

**Observatie importanta!**

Toate obiectele claselor in C# sunt create dinamic, pe HEAP.
(adica intotdeauna instantierea unei clase se face cu "new")

Product p; //aici nu se instantiaza vreun obiect (spre deosebire de C++)
//doar este declarata o referinta catre clasa Product!

pentru instantiere: p = new Product(1, "tv", 2000);

Toate obiectele C# sunt manevrate prin referinte! ("p" este o referinta catre un obiect)

Exista totusi varianta de a crea obiecte de tip "valoare" (ca in c++), pe "stack"c daca se folosesc structuri ("struct" in loc de "class")

Ierarhie de clase comuna!

Toate clase au o superclasa "Object".

De obicei, este util sa suprascriem (in clasa noastra custom) 3 metode virtuale declarate in clasa Object

(1) - reprezentare ca string a obiectului curent

```
public override string ToString()
        {
            return $"{_id};{_name};{_price}";
        }
```

(2) - compararea obiectelor (prin Equals) - se compara valorile campurilor

-observatie: este utila mai ales in situatia utilizarii obiectelor in diverse containere (colectii) oferite de platforma .NET pentru verificarea existentei, eliminare! De exemplu: List

```
    public override bool Equals(object obj)
        {
            if (obj == null)
            {
                return false;
            }

            if (!(obj is Product))
```

```
            {
                return false;
            }

            Product product = obj as Product; //(conversie cu operatorul as)

            return _id == product._id && _name == product._name;
        }


 static void Main(string[] args)
        {
            Product p = new Product(1, "car", 3000);
            Console.WriteLine(p);


            List<Product> list = new List<Product>()
            {
                new Product(1, "car", 3000),
                new Product(2, "tv", 2000)
            };

            if (list.Contains(p))
            {
                Console.WriteLine("{0} exista in lista!", p);
            }
            else
            {
                Console.WriteLine("{0} nu exista in lista!", p);
            }

            list.Remove(p); //si aici, atentie, se utilizeaza "Equals" !!!!

            Console.ReadKey();

            //p.Name = "Ionescu";
        }
```

Alta varianta de comparare, daca se doreste o relatie de ordine
intre obiecte!!

Implementarea interfetei IComparable

```
public class Product : IComparable
{

    //returneaza 0 daca obiectele sunt egale ca valoare!!!!
    //<0 (-1) daca obiectul curent precede obj
    //>0 (1) daca obj predece obiectul curent
      public int CompareTo(object obj)
        {
            if(this.Equals(obj))
            {
                return 0;
            }

            if (_id < (obj as Product)._id)
```

```
            {
                return -1;
            }

            return 1;
        }
}
```

Cod unic pentru obiecte: suprascriere GetHashCode din Oject

```
public override int GetHashCode()
        {
            return 3 * _id;
        }
```

Implementarea metodei este utila cand se doreste utilizarea obiectului
ca si cheie unica intr-un dictionar

Exemplu:

```
Dictionary<Product, string> d = new Dictionary<Product, string>();
Product p = new Product(1, "car", 3000);
d.Add(p, p.Name);
```

d[p] va fi "car"