**Objects and Classes**

**Създаване на Обект**

1. **При вече дефиниран клас:**

Student student = new Student();

student.FirstName = firstName;

student.LastName = lastName;

student.Age = age;

student.City = city;

**или**

Student student = new Student()

{

FirstName = firstName,

LastName = lastName,

Age = age,

City = city

};

Създаването на обект от вече дефиниран клас наричаме инстанциране (instantiation). Инстанция (instance) е фактическият обект, който се създава от класа по време на изпълнение на програмата.

**Build-in API Classes – Вградени/ системни класове, или както Кенов ги нарича „Държавни“**

Статични и нестатични класове – разликата между тях е, че от статичен клас не можеш да създадеш инстанция!!!!! Например не можем да кажем – DateTime date = new DateTime();

**DateTime Class:**

* Четене от конзолата:

DateTime date = DateTime.ParseExact(Console.ReadLine(),"d-M-yyyy", CultureInfo.InvariantCulture);

**Random Class: non-static**

Random randomNumber = new Random(); == var Random

Console.WriteLine(randomNumber.Next(2, 6)); - от 2 до 5 [2, 6) !!! 6 не се включва!

При по-старите версии на .Net (Framework) не е добра идея създаването на Random във for цикъл, тъй като ще се генерират едни и същи числа.

**BigInteger:**

Using System.Numerics – за много големи числа! Няма долна и горна граница и може да се стигне до грешка – **OutOfMemoryExeption!!! Не е много бърз!**

BigInteger someNumber = new BigInteger(934157136952);

**Дефиниране на класове**

public string Name { get; set; } - данни (Properties)

public int Age { get; set; }

public string Color { get; set; }

public string SayHello() - поведение

{

return "Mew";

}

* get - вземи Cat.Name
* set - записвам (задавам стойност в обекта)
* засега помним НА ИЗУСТ! Че е **public** и слагаме { get; set; }

**Конструктори**

public class Cat

{

public Cat(string name, string color)

{

// за създаването на нов обект задължително трябва да се зададат стойности по конкретните параметри. – в случая да се зададат име и цвят на коткатa!:

}

}

var newCat = new Cat("Pesho", "Black");

Когато не дефинираме нито един конструктор в даден клас, компилаторът ще създаде автоматично един без параметри, наречен “конструктор по подразбиране”.

**Правилна подредба в даден клас:**

1. Контруктори
2. Пропъртита (state състояние)
3. Методи (поведение)

Клас, който има само private конструктори, не може да бъде инстанциран. Такъв клас обикновено има само статични членове и се нарича utility клас.

**Хвърляне на грешка:**

public class Cat

{

public Cat(string name, string color)

{

if (name.Length < 2)

{

throw new Exception("Cat name cannot be less than 2 symbols.");

}

}

}

**this. -> сочи нещо от обекта!!!**

Въпреки това, понякога се налага при колизия на имената да бъде използвано полето, а не локалната променлива със същото име. В този случай, за да извлечем стойността на полето, използваме ключовата дума this. За целта достъпваме полето чрез оператора точка, приложен към this. По този начин еднозначно указваме, че искаме да използваме стойността на полето, не на локалната променлива със същото име.

**В C# не е позволено да се декларира метод, който притежава име, което съвпада с името на класа (следователно и с името на конструкторите). Ако въпреки всичко бъде деклариран метод с името на класа, това ще доведе до грешка при компилация.**

**Override-ване на метод**

public override string ToString()

{

return $"{this.Title} - {this.Content}:{this.Author}";

}

На един ред мога да създам нов обект(инстанция) и да го добавя в лист

listOfTeams.Add(new Team(currentTeamName, currentTeamCreator));

people.OrderBy(x => x.Age).ToList().ForEach(x => Console.WriteLine(x));

**щом няма void, трябва да се презапише!!!**

**Ctr + shift + A създаване на Class**