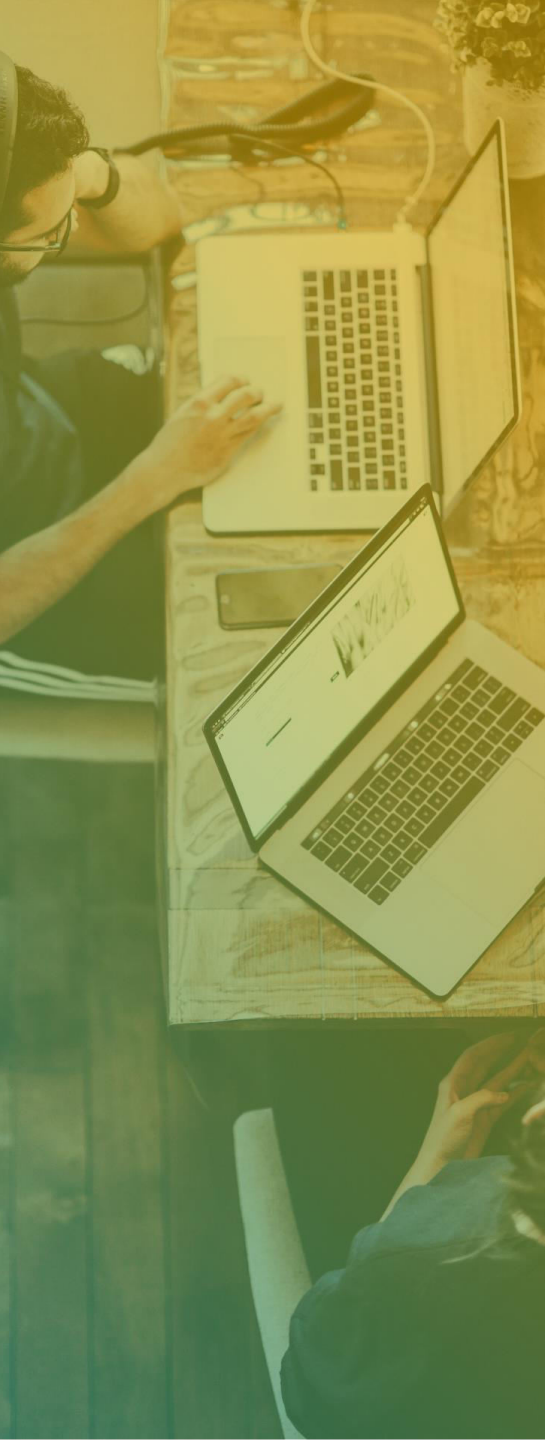


C# Object Oriented Programming

Classes and Objects

C# & .NET Development Fundamentals

Illés Horpácsi

CUBIX
INSTITUTE OF TECHNOLOGY

Class, type, object, instance

- `PrimeCounter pc = new(2, 10);`
  - PrimeCounter: Type / Class
  - pc: Instance / Object
- `public class PrimeCounter { … }`
  - Declaration of the Type / Class
- …
  - Field
  - Property
  - Method
  - Constructor
  - etc.

CUBIX
INSTITUTE OF TECHNOLOGY

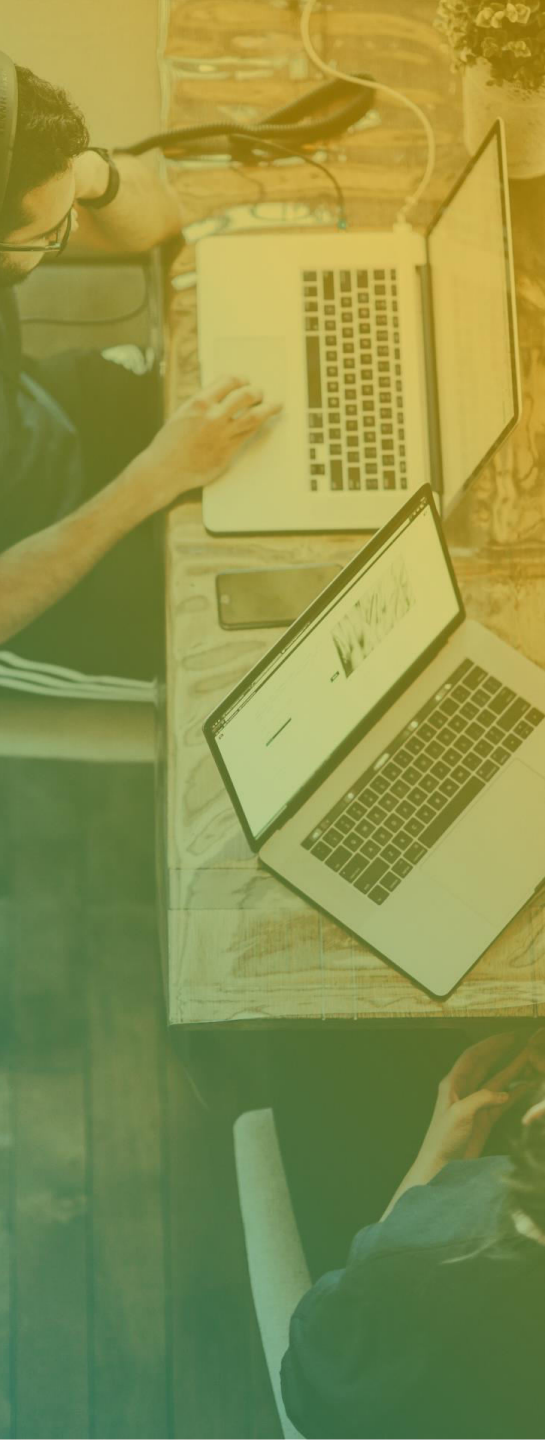
Examples

```
class Man
{
    public void SayHello() { }
}

class Student : Man
{
    public void Learn() { }
}

class Teacher : Man
{
    public void Teach() { }
}
```

```
class Examples
{
    public void TryIOut()
    {
        Man m = new();
        m.SayHello();

        Student s1 = new();
        s1.Learn();
        s1.SayHello();

        Student s2 = new();
        Student s3 = new Student();

        Teacher t = new();

        Random r = new();
        int i = r.Next();

    }
}
```

OOP Paradigm / Principles

- Encapsulation
  - Keep together
- Inheritance
  - Borrow knowledge from parent(s)
- Polymorphism
  - Same method, own action
- Abstraction
  - Hide complex inner logic



C# Object Oriented Programming

Fields, Properties and Methods

C# & .NET Development Fundamentals

Illés Horpácsi

CUBIX
INSTITUTE OF TECHNOLOGY

Class, type, object, instance

- Field
  - currentSpeed
- Property
  - LicensePlate
- Method
  - Accelerate
  - BrakingDistance

- Usage:
  - Car c = new();
  - Console.WriteLine(c.LicensePlate);
  - C.Accelerate(100);
  - Console.WriteLine(c.BrakingDistance(true));

- currentSpeed may be used only within
  - Do NOT create public field!
- c.LicensePlate has no parameters-> ()

```
class Car
{
    private int currentSpeed;

    public string LicensePlate { get; set; }

    public void Accelerate(int speed)
    {
        currentSpeed += speed;
    }

    public int BrakingDistance(bool wet)
    {
        if (wet)
        {
            return currentSpeed * 2;
        }
        return currentSpeed;
    }

}
```

Property

- Auto property
  - Snippet: prop

```
public int MyProp { get; set; }
```

```
public int MyProp { get; }
```

```
public int MyProp { get;  private set; }
```

- Full property
  - Snippet: propfull

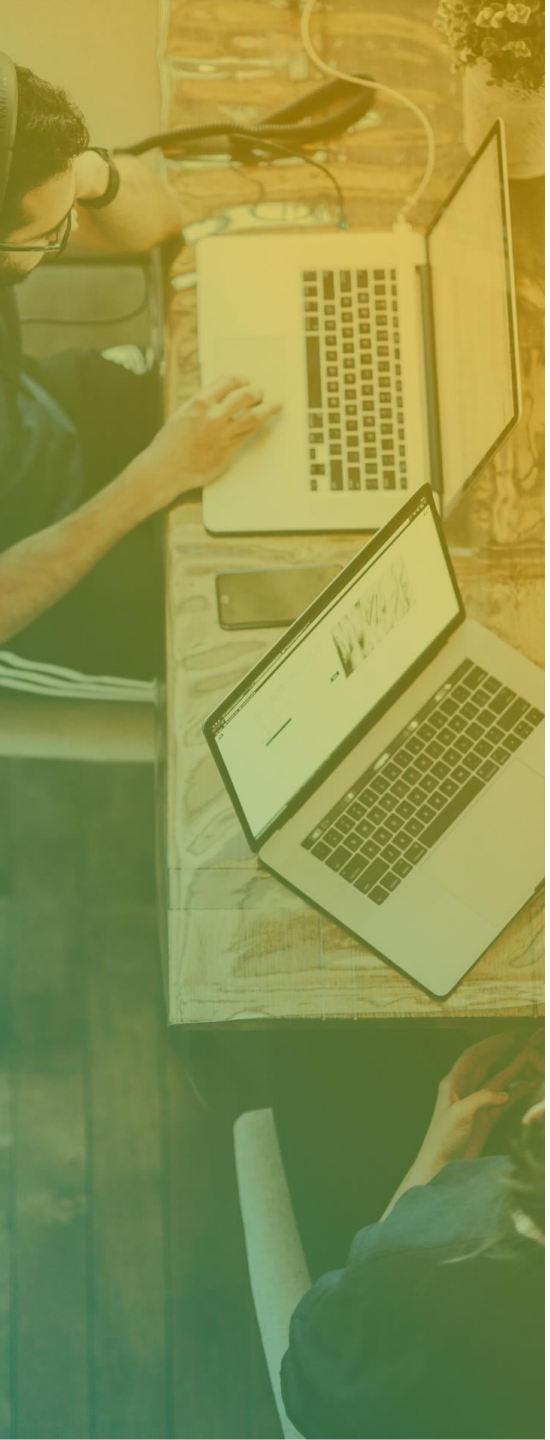
```
private int myProp;

public int MyProp
{
    get { return myProp; }
    set { myProp = value; }
}
```

```
private double price;

public double Price
{
    get { return price * 1.27; }
    set { price = value; }
}
```

CUBIX
INSTITUTE OF TECHNOLOGY

Method

- Complex logic
- Has some paramaters (maybe 0, but has place for it!)
- May have no return value (void)
  - If not void, every branch must have a return value
- Return value's type may differ from the type(s) of its parameter(s)
- public, private, internal, ...



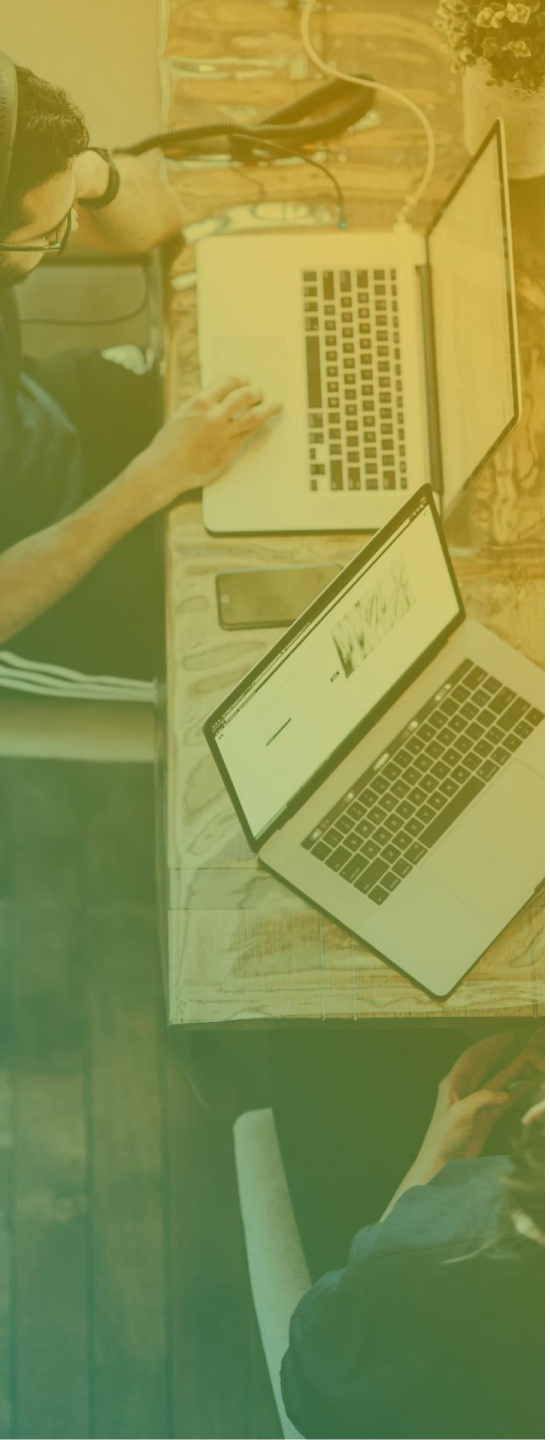
C# Object Oriented Programming

Constructor, Visibility

C# & .NET Development Fundamentals

Illés Horpácsi

EUBIX
INSTITUTE OF TECHNOLOGY



C# Object Oriented Programming

Inheritance, Interface, Abstract class

C# & .NET Development Fundamentals

Illés Horpácsi

CUBIX
INSTITUTE OF TECHNOLOGY

Inheritance

- Example: UniStudent : Student
- Only one parent class allowed
- Any base class has a parent: object
  - Eg. ToString() is inherited from there
- If a base class „knows” something, it will be inherited
  - Except private methods, fields
- Method may work differently
  - virtual / override
- Can be marked as „sealed”

Abstract class

- Example: abstract class Man
- Can't create an instance: Man m = new(); will not work
- May have methods, that can be used in child classes
  - It has a method body (logic)
- May have abstract methods
  - just declares a method that must be implemented in child/children
  - It has NO body (logic)

Interface

- Example: interface IMove
- Name used to start with capital I
- Not inherited from, but implemented
- A class can implement an unlimited number of them
- C# already includes many of them
  - IComperable, IComparer, IEnumerator, IEnumerable…