

## Lektion 7

Monday, November 17, 2025 8:52 AM

### Class Constructor

A constructor is a special method that is used to initialize objects. The advantage of a constructor, is that it is called when an object of a class is created. It can be used to set initial values for fields:

```

namespace ClassConstructor
{
    class Person
    {
        //Properties of the class
        public string Name { get; set; }
        public int Age { get; set; }

        //Constructor to initialize Name and Age
        public Person(string name, int age)
        {
            Name = name; //Set the Name property
            Age = age; //Set the Age property
        }

        //Method to display details about a person
        public void DisplayInfo()
        {
            Console.WriteLine($"Name: {Name}, Age: {Age}");
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            //Create an instance of the "Person"-class using the constructor
            Person person = new Person("Alice", 25);

            //Call the method to display
            person.DisplayInfo();
        }
    }
}

```

Resultat:

Microsoft Visual Studio Debug

Name: Alice, Age: 25

C:\Users\...\Documents\GitHub\G...

### Class Grader

```

internal class Program
{
    class Student
    {
        private int Marks;
        private int ObtainedMarks;

        //Constructor
        public Student(int marks, int obtainedMarks)
        {
            Marks = marks;
            ObtainedMarks = obtainedMarks;
        }

        //Method to calculate the %
        public double CalculatePercentage()
        {
            return (double) ObtainedMarks / Marks * 100;
        }

        //Method to generate message
    }
}

```

```

public string GetMessage()
{
    double percentage = CalculatePercentage();

    if (percentage >= 90)
    {
        return $"Exellent! {percentage:F2}%";
    }
    else if (percentage >= 75)
    {
        return $"Good job! {percentage:F2}%";
    }
    else if (percentage >= 50)
    {
        return $"You passed with! {percentage:F2}%";
    }
    else
    {
        return $"You scored! {percentage:F2}%. Better luck next time.";
    }
}

0 references
static void Main(string[] args)
{
    Console.WriteLine("Enter the totalt marks: ");
    int marks = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("Enter the marks obtained: ");
    int obtainedMarks = Convert.ToInt32(Console.ReadLine());

    Student student = new Student(marks, obtainedMarks);

    double percentage = student.CalculatePercentage();
    string message = student.GetMessage();

    Console.WriteLine($"Percentage: {percentage:F2}%");
    Console.WriteLine(message);
}
}

```

Resultat:

```

Enter the totalt marks: 6
Enter the marks obtained: 2
Percentage: 33,33% You scored! 33,33%. Better luck next time.

```

## Class Fields

```

0 references
internal class Program
{
    3 references
    class Product
    {
        //Private fields to store the product data
        private string name;
        private double price;
        private int stock;

        //Constructor , initializes a new instance of the product class with the specified name, price, stock
        public Product(string name, double price, int stock)
        {
            this.name = name;
            SetPrice(price);
            SetStock(stock);
        }

        //Method to set the name of the product
        public void SetName(string name)
        {
            this.name = !string.IsNullOrEmpty(name) ? name : "Invalid name";
        }

        //Method to get the name of the product
        public string GetName()
        {
            return name; //return the name of the product
        }

        //Method to set the price of the product
        public void SetPrice(double price)
        {
            //Assign a positive price or default 0 if the input is invalid
            this.price = price > 0 ? price : 0;
        }
    }
}

```

```

//Method to get the price of the product
0 references
public double GetPrice()
{
    return price; //return the price of the product
}

//Method to set quantity of the product in stock
2 references
public void SetStock(int stock)
{
    //Assign a non-negative value or default to 0 if input is invalid
    this.stock = stock >= 0 ? stock : 0;
}

//Method to get the quantity of the product
0 references
public int GetStock()
{
    return stock; // return the quantity of the stock
}

//Method to return details of the product
2 references
public string ReturnDetails()
{
    return $"Name: {name}\nPrice: {price}$\nStock: {stock}";
}

0 references
static void Main(string[] args)
{
    //Create an instance of the product
    Product product = new Product("Laptop", 999.99, 10);

    //Display product details
    Console.WriteLine(product.ReturnDetails());

    //Update product details
    product.SetName("Desktop");
    product.SetPrice(599.99);
    product.SetStock(20);
    Console.WriteLine(product.ReturnDetails());
}

```

Resultat:

```

Name: Laptop
Price: 999,99$
Stock: 10
Name: Desktop
Price: 599,99$
Stock: 20

```

## Class Properties

```

namespace ClassProperties
{
    //Define a class named Program that contains the main execution logic
    0 references
    internal class Program
    {
        //Define a nested class named Person to represent a person
        3 references
        class Person
        {
            //Private fields to store the name and age of a person
            private string name;
            private int age;

            //Constructor to initialize a new instance of the person class
            1 reference
            public Person(string name, int age)
            {
                //Assign the parameters to the private fields
                this.name = name;
                this.age = age;
            }

            //Method to set the name field with validations
            0 references
            public void SetName(string name)
            {
                //Assign the new name if it's not null or empty, otherwise set "Invalid name"
                this.name = !string.IsNullOrEmpty(name) ? name : "Invalid name";
            }

            //Method to get the current value of the name field
            0 references
            public string GetName()
            {

```

```

        return name;
    }

    //Method to set the age field with validations
    0 references
    public void SetAge(int age)
    {
        //Assign the new age if it's within a valid range 0-150, otherwise set to -1
        this.age = age >= 0 && age <= 150 ? age : -1;
    }

    //Method to get the current value of the age field
    0 references
    public int GetAge()
    {
        return age;
    }

    //Method to return a string containing the person's details
    1 reference
    public string ReturnDetails()
    {
        //Format and return the name and age as a string
        return $"Name: {name}\nAge: {age}";
    }

}

//Main method to entry point of the program
//above code does nothing until main method calls to it
0 references
static void Main(string[] args)
{
    //Create a new person object with the name "Bob" and age "55"
    Person person = new Person("Bob", 55);

    //Display the person details in the console
    Console.WriteLine(person.ReturnDetails());
}
}

```

Resultat:

```

Name: Bob
Age: 55
C:\Users\admin1\Documents\C#\ClassP2

```

### Simplified version

```

namespace ClassPropertiesP2
{
    0 references
    internal class Program
    {
        3 references
        class Person
        {
            private string name;
            private int age;

            1 reference
            public string Name
            {
                get => name;
                set => name = !string.IsNullOrEmpty(value) ? value : "Invalid name";
            }

            1 reference
            public int Age
            {
                get => age;
                set => age = value >= 0 && value <= 150 ? value : -1;
            }

            1 reference
            public Person(string name, int age)
            {
                Name = name;
                Age = age;
            }

            1 reference
            public string ReturnDetails()
            {
                return $"Name: {name}\nAge: {age}";
            }
        }
    }
}

```

```

        }
    }
}
0 references
static void Main(string[] args)
{
    Person person = new Person("Bob", 55);
    Console.WriteLine(person.ReturnDetails());
}
}
}

```

Alternativt

```

0 references
internal class Program
{
    3 references
    class Person
    {
        2 references
        public string Name { get; set; }
        2 references
        public int Age { get; set; }

        /*
        public string Name
        {
            get => name;
            set => name = !string.IsNullOrEmpty(value) ? value : "Invalid name";
        }

        public int Age
        {
            get => age;
            set => age = value >= 0 && value <= 150 ? value : -1;
        }
        */

        1 reference
        public Person(string name, int age)
        {
            Name = name;
            Age = age;
        }

        1 reference
        public string ReturnDetails()
        {
            return $"Name: {Name}\nAge: {Age}";
        }
    }
    0 references
    static void Main(string[] args)
    {
        Person person = new Person("Bob", 55);
        Console.WriteLine(person.ReturnDetails());
    }
}

```

Ger samma resultat som ovan; Name: Bob, Age: 55

### Class To String

```

0 references
internal class Program
{
    8 references
    class Person
    {
        private string name;
        private int age;

        2 references
        public Person (string name, int age)
        {
            this.name = name;
            this.age = age;
        }

        0 references
        public void SetName(string name)
        {
            this.name = !string.IsNullOrEmpty(name) ? name : "Invalid name";
        }
    }
}

```

```
0 references
public string GetName()
{
    return name;
}

0 references
public void SetAge(int age)
{
    this.age = age >= 0 && age <= 150 ? age : -1;
}

0 references
public int GetAge()
{
    return age;
}

0 references
public override string ToString()
{
    return $"Name: {name}\nAge: {age}";
}

1 reference
public override bool Equals(object obj)
{
    if(obj is Person)
    {
        Person person = obj as Person;
        return name.Equals(person.name) && age == person.age;
    }
    return false;
}

0 references
static void Main(string[] args)
{
    Person person = new Person("Bob", 34);
    Person test = new Person("Bob", 34);

    if (person.Equals(test))
    {
        Console.WriteLine("Same");
    }
    else
    {
        Console.WriteLine("Not same");
    }
}
```

Resultat:



