# Creating Your First Class and Objects

**Gill Cleeren**
CTO Xebia Microsoft Services Belgium

@gillcleeren

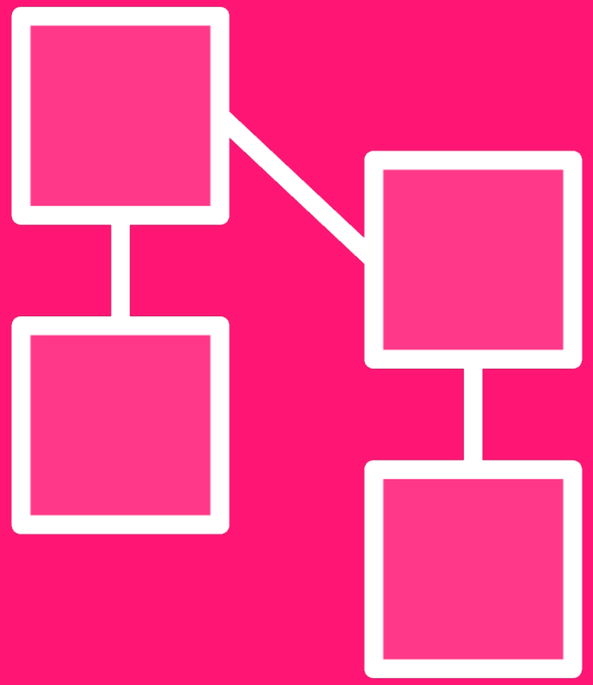
# Overview

Understanding classes

Creating the Employee class

Using objects

# Understanding Classes



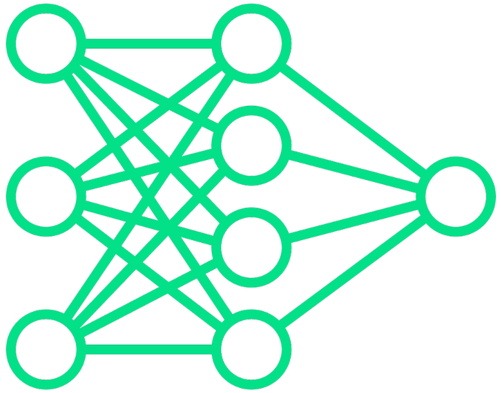
# With just variables, we only get so far.

If we want to represent a structure, we need a custom type.

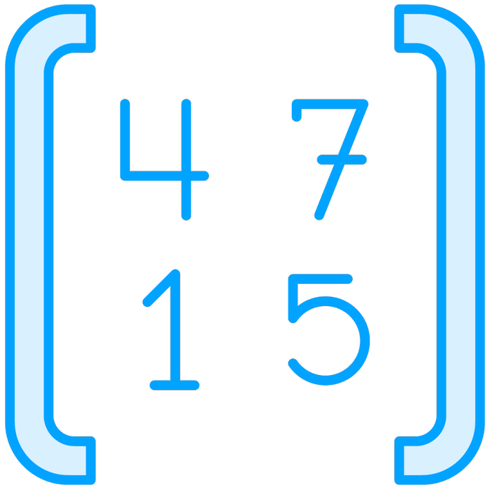
**Typical models**

- Employee
- Customer
- Message
- Transaction

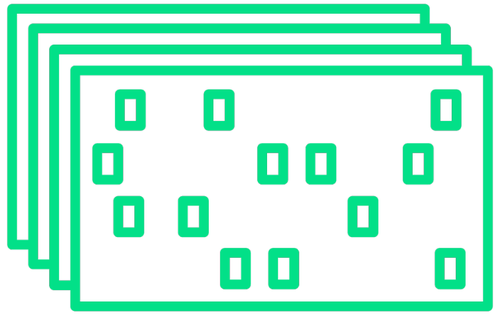
# Custom Types



Class
Most commonly used



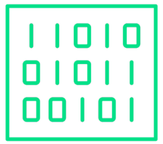

Struct



Record

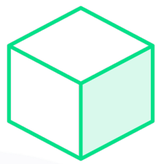
# Classes in C#

Blueprint of an object

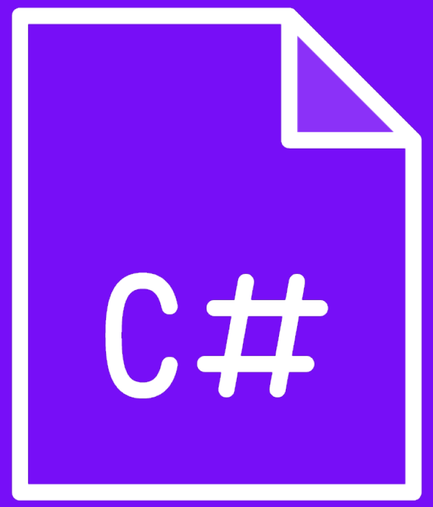


Defines data and functionality to work on its data

Created using class keyword



Foundation of OO (object-orientation)




# In C#, most code will live inside a class

Program.cs and Utilities class used up until now

Most code will live inside a class

# The Class Template

```
public class MyClass
{
    public int a;
    public string b;

    public void MyMethod()
    {
        Console.WriteLine("Hello world");
    }
}
```

# Contents of a Class

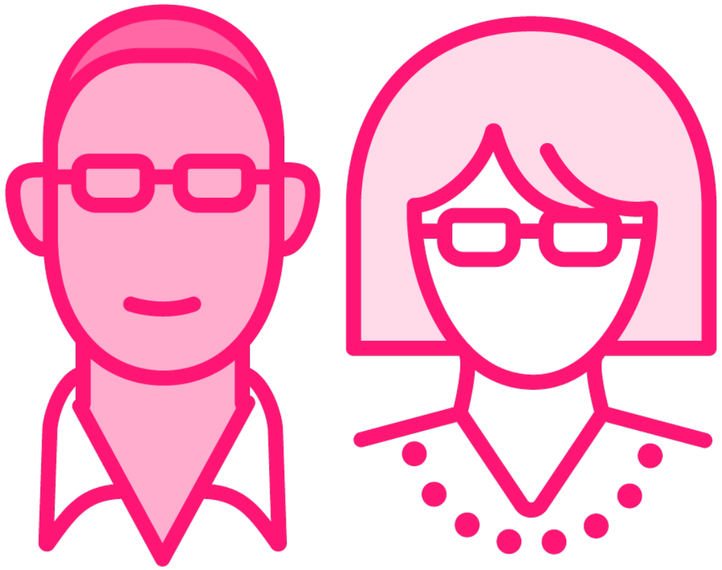
Fields

Methods

Properties

Events

# Creating the Employee Class

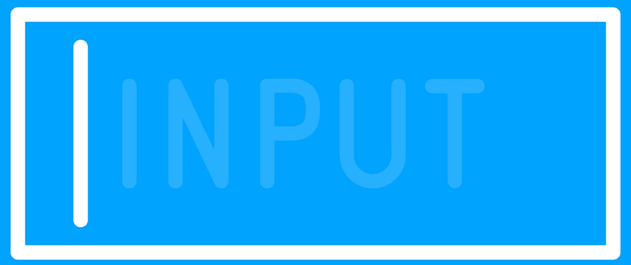


**Thinking of an Employee in real life**

- Identity: Name
- Attributes: Age, Wage
- Behaviors: Get paid, Perform work

```
public class Employee
{
    //class code will come here
}
```

## Creating the Employee Class



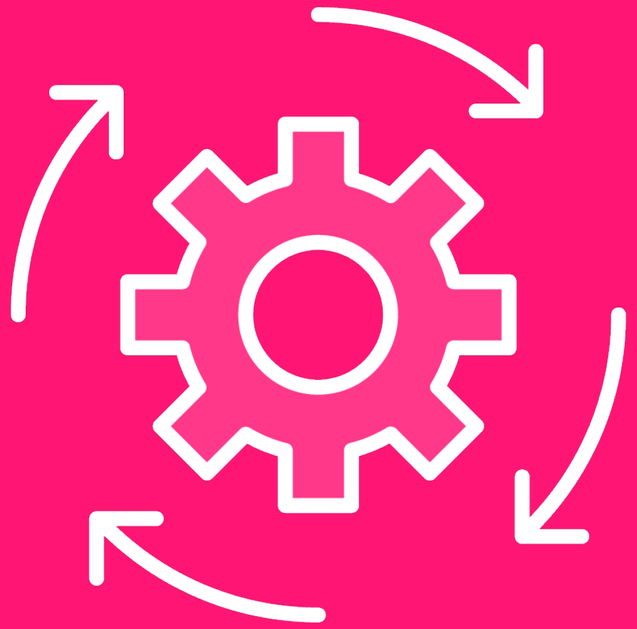

# Adding Fields

Class-level variables

Contain value

# Adding the Employee Fields

```
public class Employee
{
    public string firstName;
    public int age;
}
```



# Adding Methods

Perform actions

Often change the state

# Adding Methods

```
public class Employee
{
    public string firstName;
    public int age;

    public void PerformWork()
    {
        //method code goes here
    }
}
```

# Access Modifiers

public

private

protected

# Demo

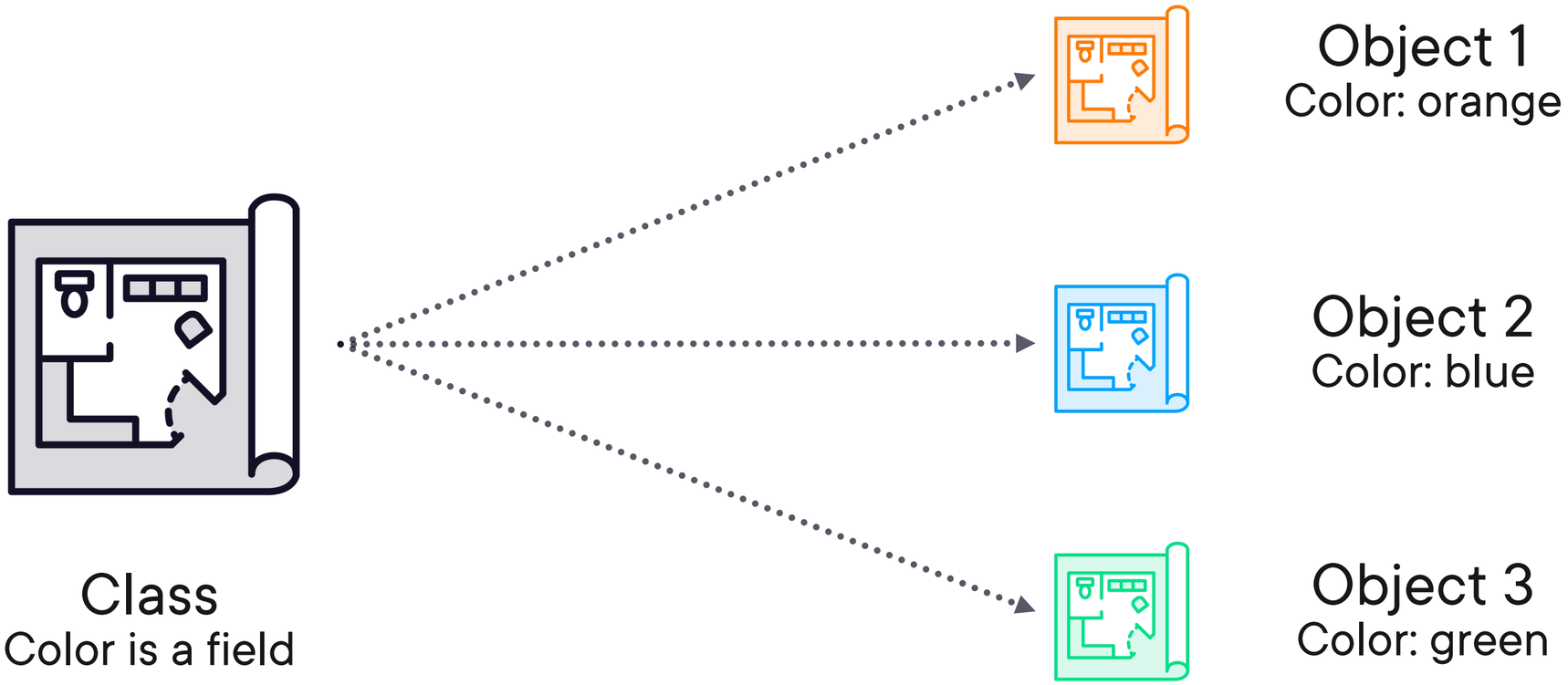
Creating the Employee class

Adding data using fields

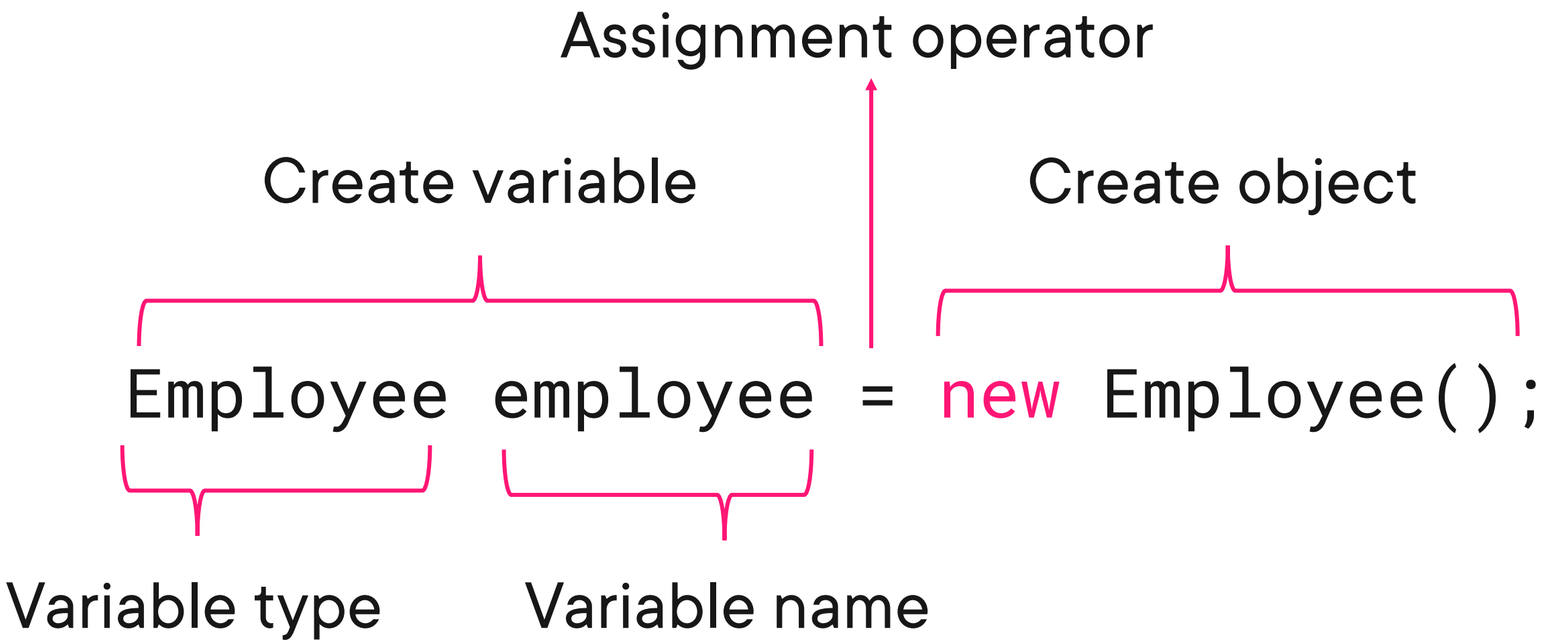
Adding methods

# Using Objects

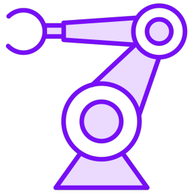
# Classes and Objects



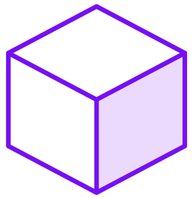
# Creating a New Object



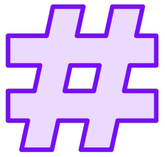
# Constructors



Called when instantiating an object happens



Default or custom



Used to set initial values

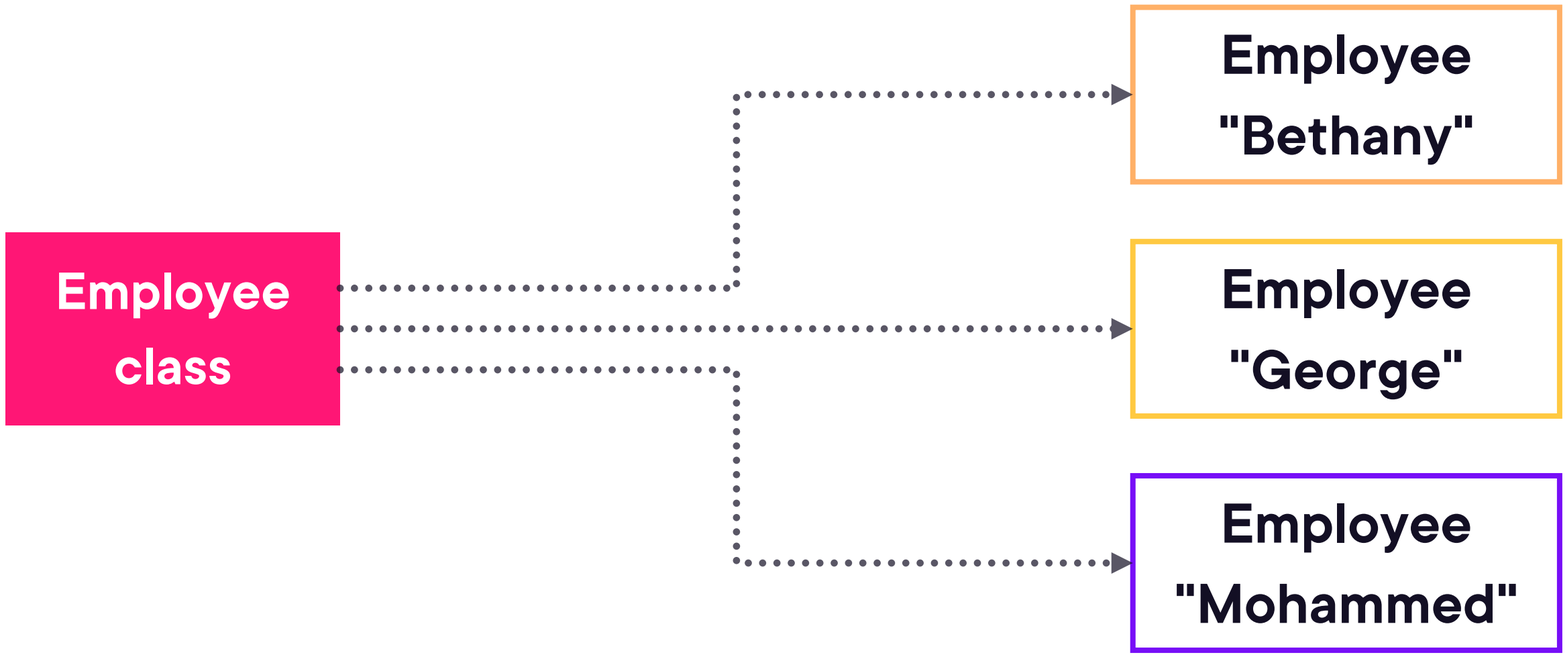
# Adding a Constructor with Parameters

```
public class Employee
{
    public string firstName;
    public int age;

    public Employee(string name, int ageValue)
    {
        firstName = name;
        age = ageValue;
    }
}
```


# Using the Constructor

# Creating Objects Using the Constructor



# The Default Constructor

```
public class Employee
{
    public Employee()
    { }
}
```

# Is there always a default constructor?

No! Only if we define no other constructors!

```
public class Employee(string name, int ageValue)
{
}
```

# Using Primary Constructors

## Introduced with C# 12

```
Employee employee = new Employee();
```

◀ **Instantiating the object**

```
employee.PerformWork();
```

◀ **Invoking a method**

```
employee.firstName = "Bethany";
```

◀ **Changing a field**

```
int wage = employee.ReceiveWage();
```

◀ **Returning a value from a method**


# A Shorthand to Create an Instance

# Demo

Adding a constructor

Creating an object

Using the dot operator

# Demo

## Working with several objects

# Summary

**Classes are the main building block in C#**

**Define fields and methods**

**Are the blueprint for creation of objects**

- Constructors

Up Next:

# Understanding value and reference types