Creating Your First Classes and Objects

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren | www.xpirit.com/gill

Agenda

- Understanding classes
- Creating the Employee class
- Using the class
- Adding properties

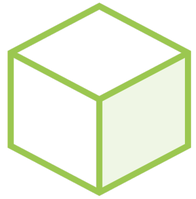
Understanding Classes

Classes in C#

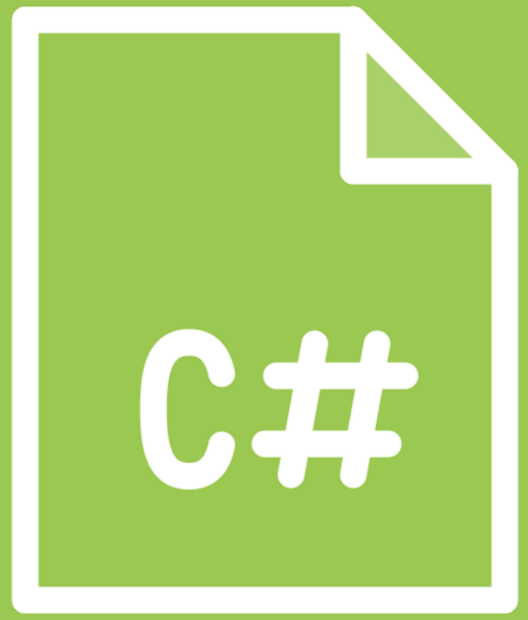
Blueprint of an object

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 class used up until now

All code will live inside a class

Classes are reference types

The Class Template

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

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

Contents of a Class

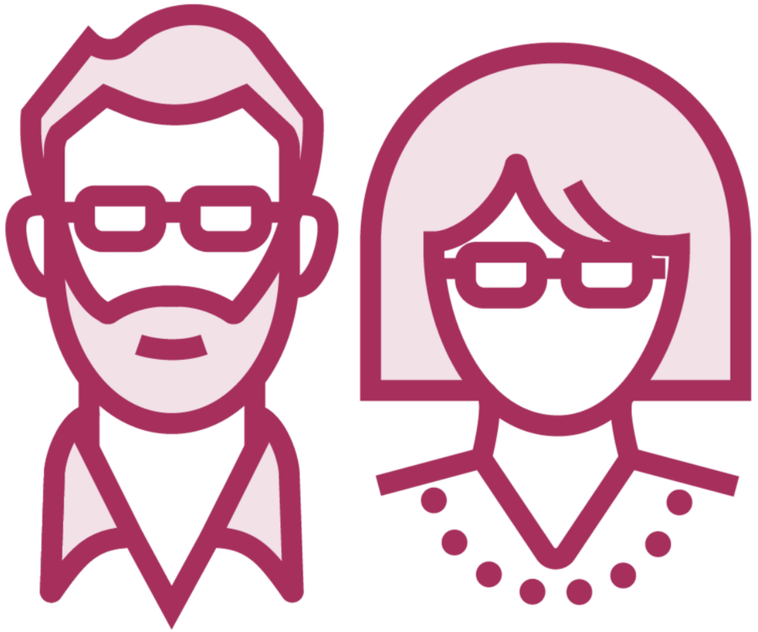
Fields

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

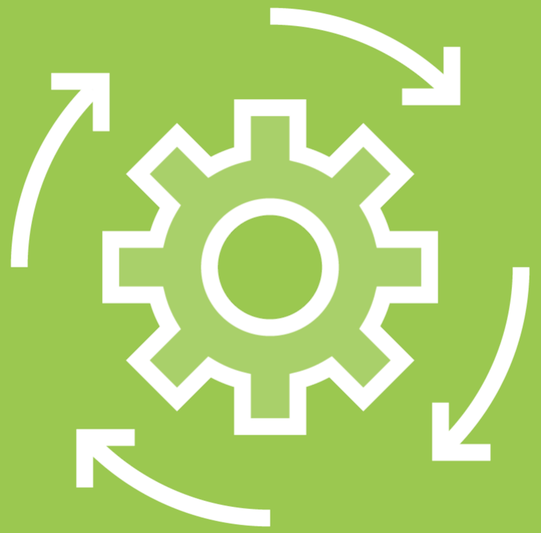
Class-level variables

Contain value

Often used in combination with properties and thus private

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

    }

}
```

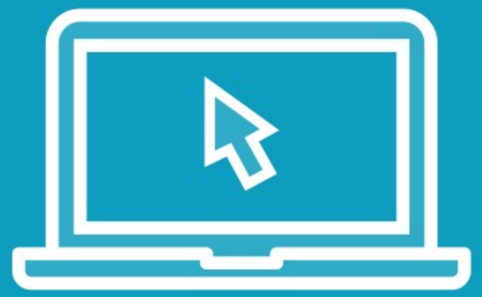
Access Modifiers

public

private

protected

Demo



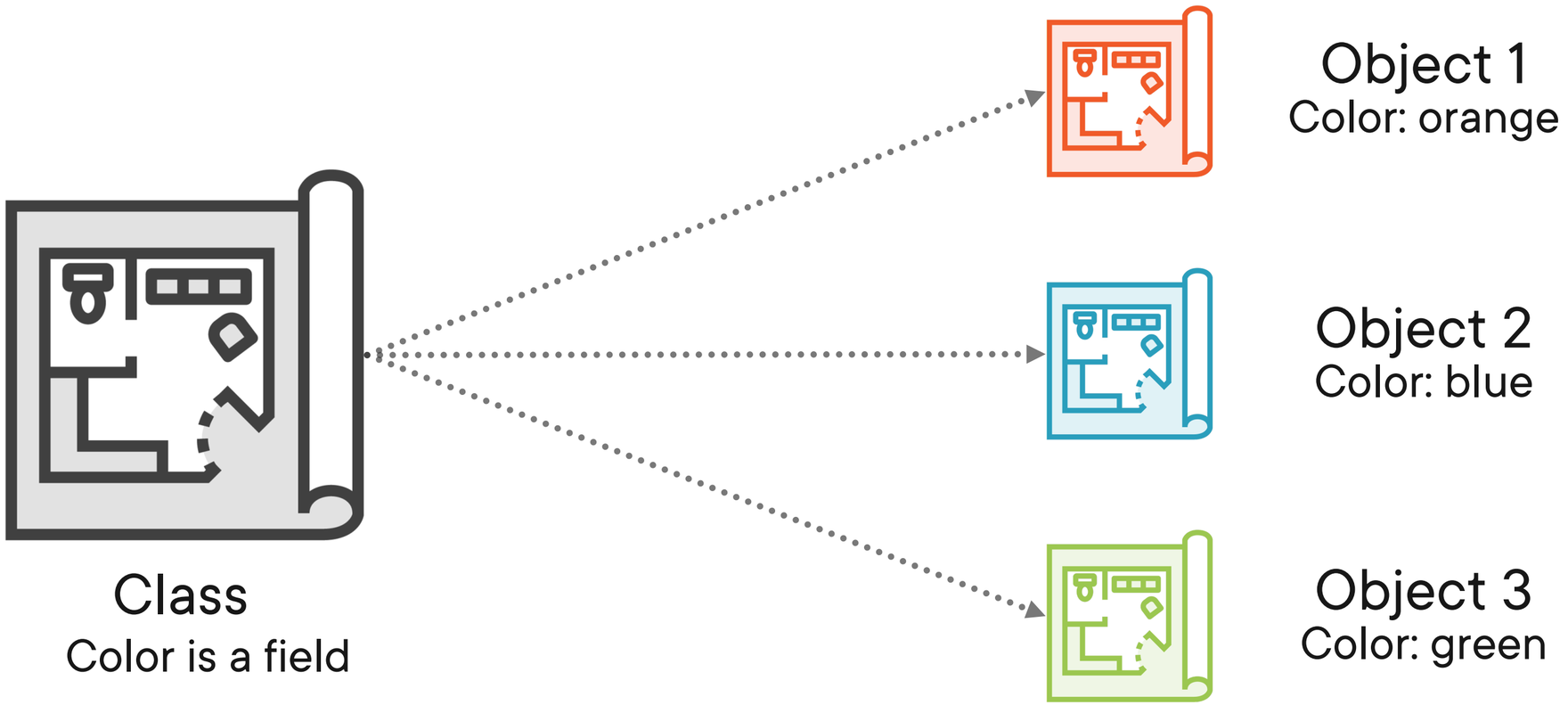
Creating the Employee class

Adding data using fields

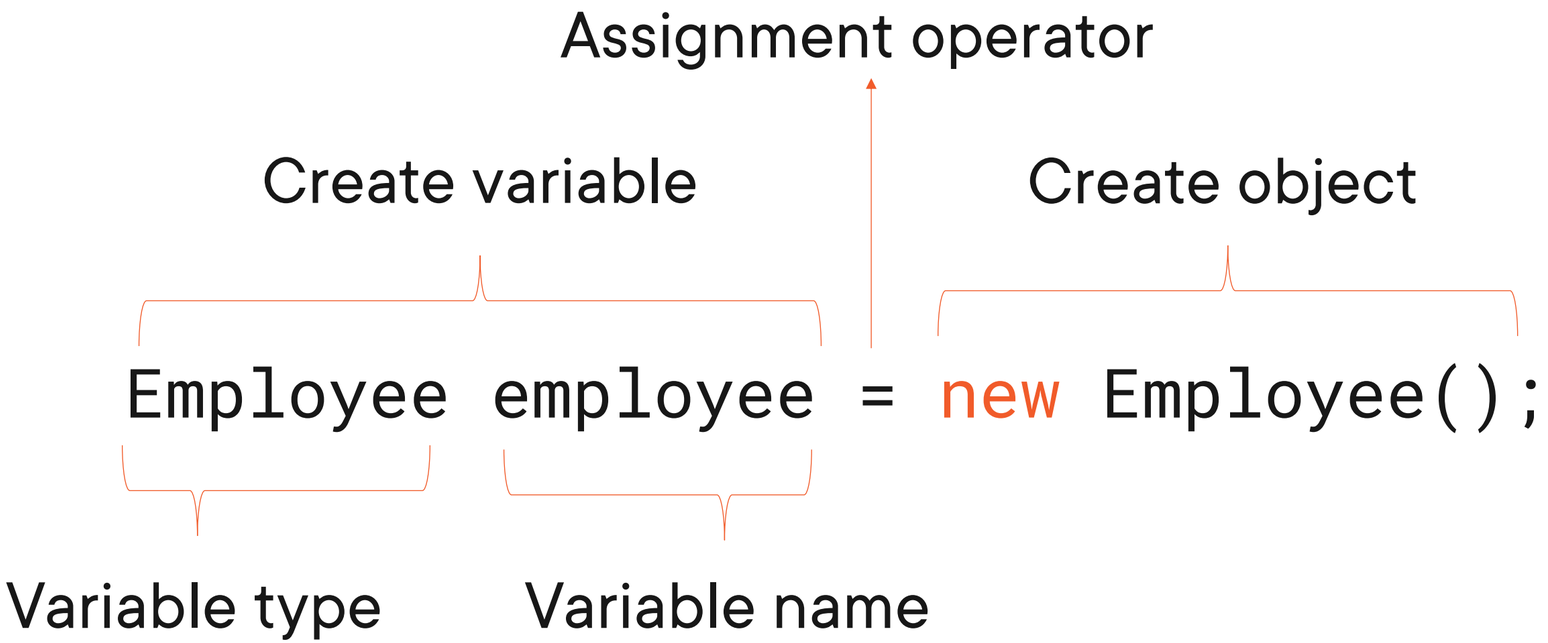
Adding methods

Using the Class

Classes and Objects



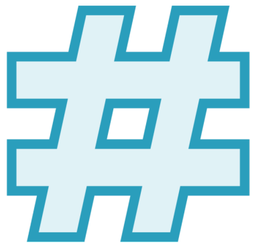
Creating a New Object



Constructors

Called when instantiating an object happens

Default or custom



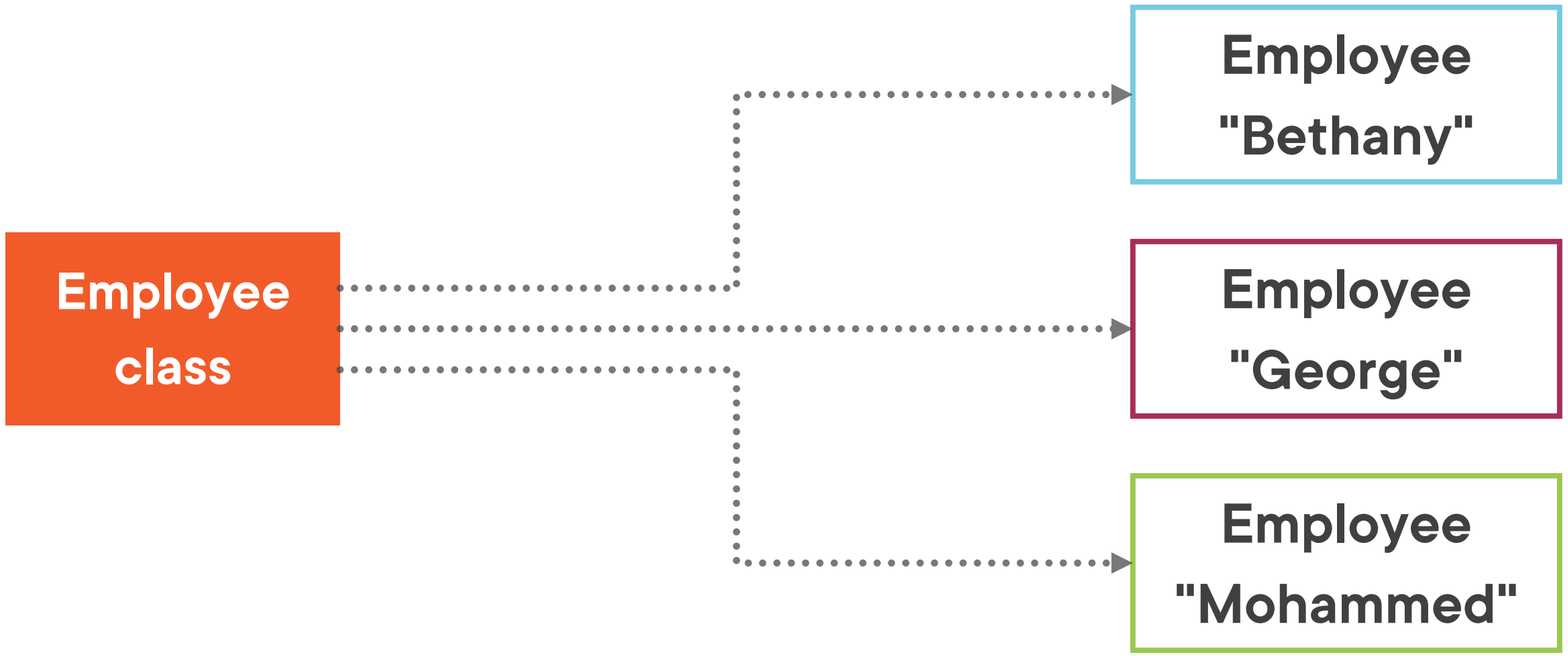
Used to set initial values

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!

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

◀ Instantiating the object

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

◀ Invoking a method

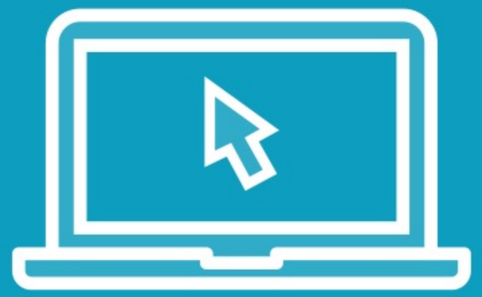
```
employee.firstName = "Bethany";
```

◀ Changing a field

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

◀ Returning a value from a method

Demo

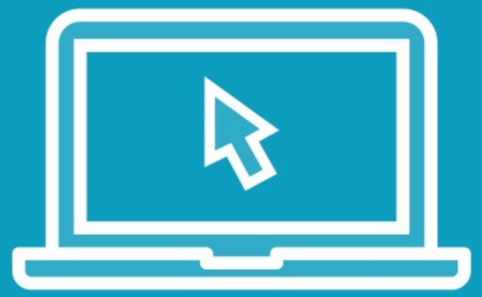


Adding a constructor

Creating an object

Using the dot operator

Demo

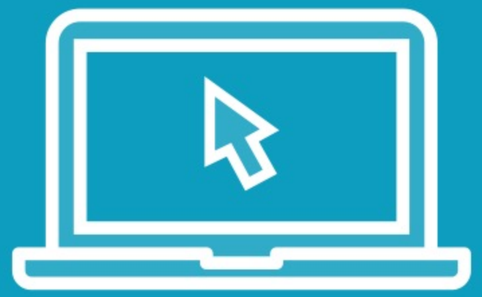


Working with several objects

Understanding Classes Are Reference Types

```
Employee employee1 = new Employee();

employee1.firstName = "Bethany";


Employee employee2 = employee1;

employee2.firstName = "George";


string check = employee1.firstName;//check will be George
```

Demo



Classes are reference types

Adding Properties

So Far, Our Data Is Stored in Fields

```
public class Employee
{

    public string firstName;

    public int age;



    public Employee(string name, int ageValue)
    {

        firstName = name;

        age = ageValue;
    }

}
```

```
Employee employee1 = new Employee();

employee1.firstName = "Bethany";
```

Manipulating a Class's Data

Other classes can directly change the field data

Access to class data

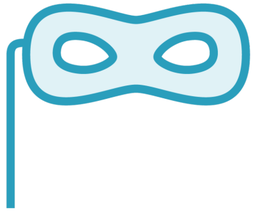
If data is public, everyone can change the data of an object

Introducing Properties

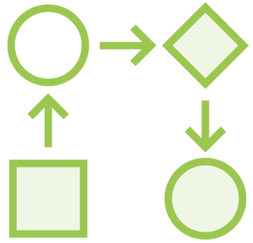
```
public class Employee
{
    private string firstName;

    public string FirstName
    {
        get { return firstName; }

        set

        {

            firstName = value;

        }

    }

}
```

C# Properties

Wraps data (fields) of a class



Hide implementation



Define get and set implementation

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

◀ Instantiating the object

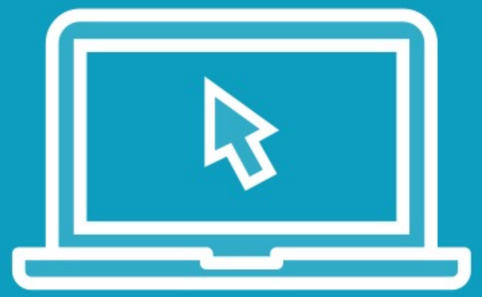
```
employee.FirstName = "Bethany";
```

◀ Setting a value through a property

```
int empFirstName = employee.FirstName;
```

◀ Getting the value through a property

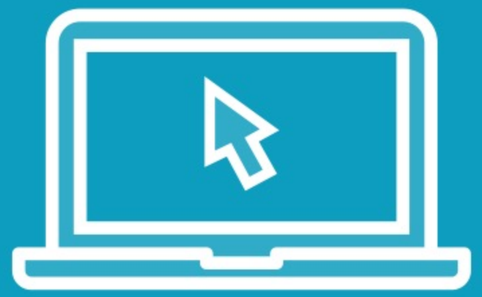
Demo



Adding properties on our class

Using the properties instead of the fields

Demo



A small employee application

Exercise

Assignment

- **Add an option to change the hourly rate for an employee**
- **Add an option to give a bonus and add that to the wage**

A solution is available with the downloads

Resources

Other relevant courses in the C# path:

- **Object Oriented development in C#**
  - **Deborah Kurata**
- **C# Events, Delegates, and Lambdas**
  - **Dan Wahlin**
- **Working with C# Records**
  - **Roland Guijt**

Summary

Classes are the main building block in C#

Define fields, properties and methods

Are the blueprint for creation of objects

- Constructors

Up next:

Doing more with classes and objects