The Fundamentals of C#

Classes and Objects

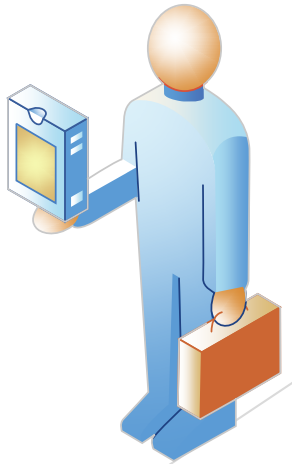
Objectives:

- What is an object?
- What is a class?
- What is a constructor method?
  - Multiple constructors

What is an object?

An object is a programming construct
which encapsulates data (state)
and methods to process that data (behavior)

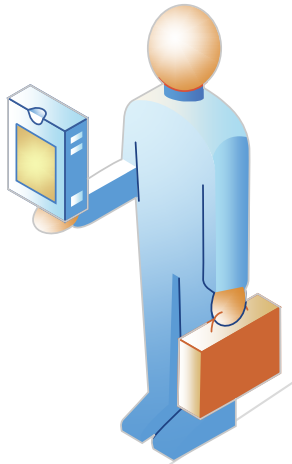


What is an object?

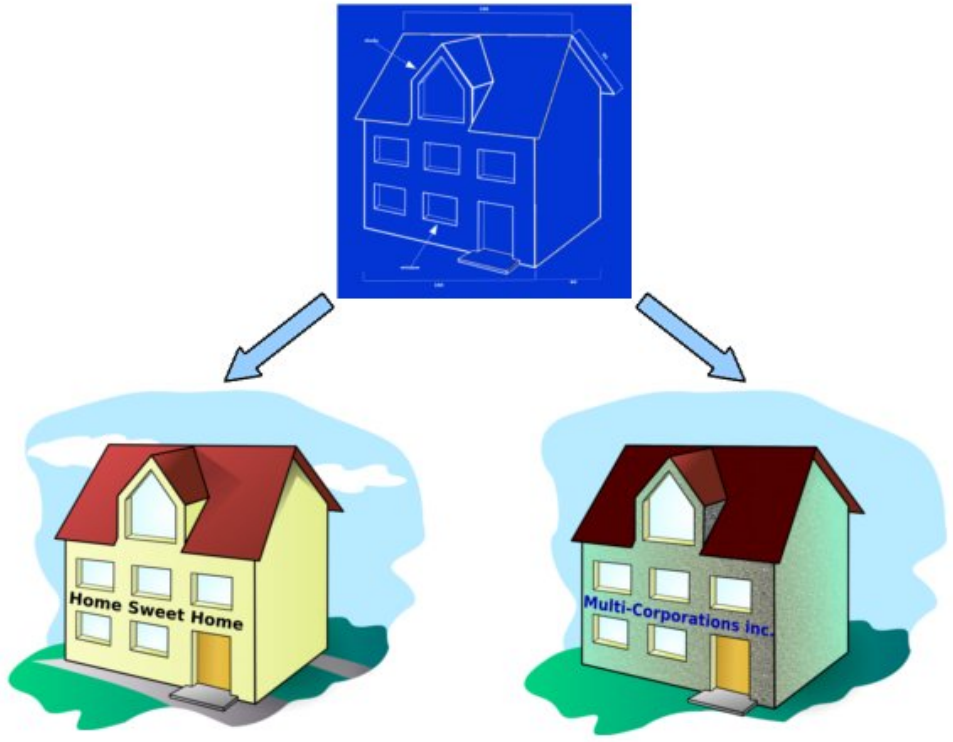
Objects allow us to represent real-world concepts more easily in our code.



What is a class?

In order to create an object we first need a blueprint. The class is our blueprint, specifying the data and behavior an object will have



Example

In a very simple example, we are interested in the following aspects of employees.

Data (State)
name
hourly pay rate

Behaviour
calculation of weekly pay

Classes and Objects

Each employee is an object. If we had 10 employees, we'd have 10 objects, 1 to represent each employee.

The employee class is a template for all employees.

An employee object is sometimes called an instance of the employee class.

Classes

Classes are made up of **Fields** and **Methods**. These are known as a classes' **Members**

Fields are variables that can have values. These hold the object's data (state).

Methods describe the behaviour associated with an object.

Employee fields

name

payRate

How we write it in C#

```
public class Employee
{
    public string name
    public double payRate
……
……
}
```

A field is a variable declared in a class definition (rather than in a method)

Employee method

We need a method to calculate an employee's pay.

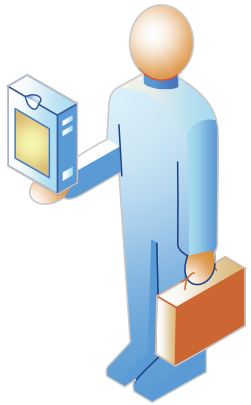
We will assume that all employees work 40 hours a week.

The method will have no parameters and it will return a double value.

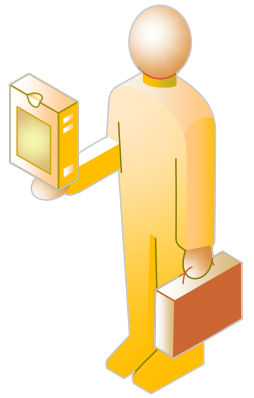
C# code for the Employee class

```
public class Employee
{
    public string name;
    public double payRate;

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

The class will be used to create multiple employee objects with different values in their fields.



Name: Carl
Pay Rate: $35.00



Name: Andrew
Pay Rate: $75.00

Employee class

Employee.cs

```
public class Employee
{
    public string name
    public double payRate

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

Can be compiled.

Cannot be executed.
Why not?

Employee class

- Can be compiled : YES
- Cannot be executed
  (because it has no **Main() method**)
- To use the Employee class
  we must create an Employee **object**
  (from another class)

Objects

An object is created using the **new** keyword and a **constructor method**.

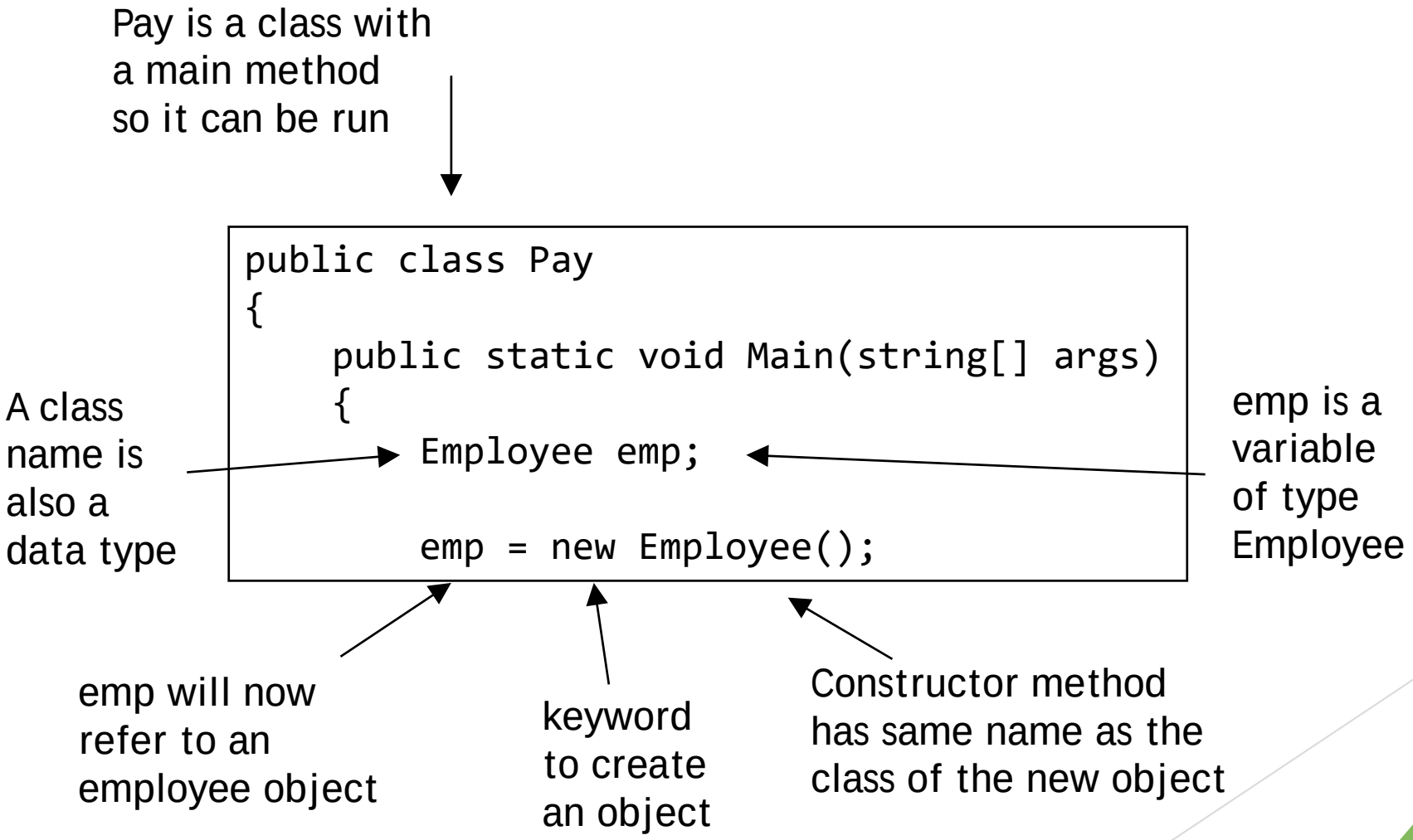
The name of the constructor method is **always the same as the class name.**

An object exists in memory and exists only while a program is executing.

A variable is required to store the object in the same way variables are required for numbers.

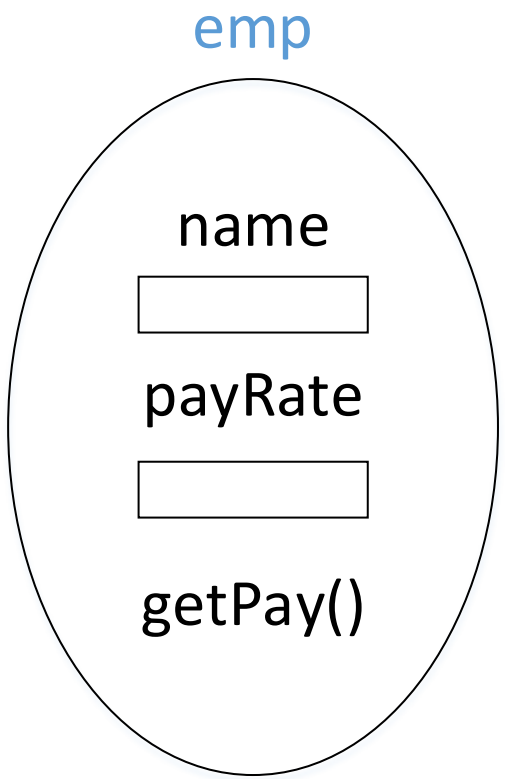
The type of the object's variable is the class name.

Creating an employee object



Creating an employee object (Continued)

Employee emp = new Employee()




```
public class Employee
{
    public string name
    public double payRate

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

Accessing an object's fields and methods

Dot notation can be used to access an object's fields or methods

syntax:

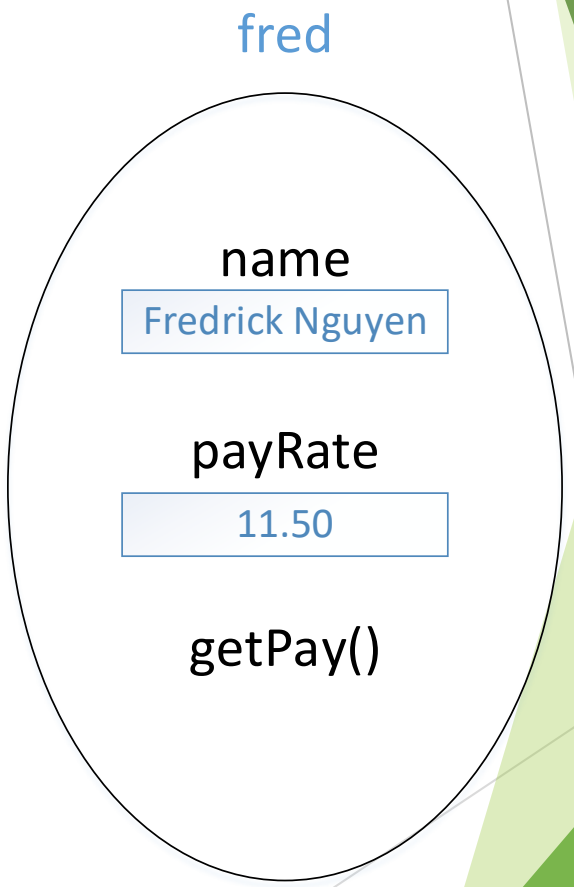
```
objectVariable.field

objectVariable.Method()
```

example:

```
emp.name

emp.GetPay()
```

Setting Object fields

```
public class Pay
{
    public static void Main(string[] args)
    {
        Employee fred = new Employee();

        fred.name = "Frederick Nguyen";
        fred.payRate = 11.50;

        double pay = fred.GetPay();
        …
```



Example

```
public class Pay
{
     public static void Main(string[] args)
     {
          Employee x = new Employee();
          Employee y = new Employee();
          x.name = "Michael Dempsey";
          y.name = "Lily Chang";
          x.payRate = 7.50;
          y.payRate = 12.00;
          Console.WriteLine(x.name + "\t" + x.GetPay());
          Console.WriteLine(y.name + "\t" + y.GetPay());
     }
}
```

Constructor method

A constructor method is a special kind of method used to create an object. As such, it has a special kind of **signature**.

It always has the same name as the class.

No return type is specified (It returns an object of the class).

Every class has a constructor method.
If you don't write one, C# creates one.

Constructor method (Continued)

When a constructor method is called, it creates an object. The constructor should initialize the state of the object.

Because of this you'll often want to write your own, custom constructor to configure the initial state of your fields to the way you want.

Remember, a constructor is about **initialization**, **NOT** behavior. That should be left to the methods.

Employee constructor

```
public class Employee
{
    public string name;
    public double payRate

    public Employee(string n, double rate)
    {
        name = n;
        payRate = rate;
    }

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

Example

```
public class Pay
{
    public static void Main(string[] args)
    {
        Employee x = new Employee("Michael Dempsey", 7.50);
        Employee y = new Employee("Lily Chang", 12.00);
        Console.WriteLine(x.name + "\t" + x.GetPay());
        Console.WriteLine(y.name + "\t" + y.GetPay());
    }
}
```

Class with default constructor

```
public class Employee
{
    public string name;
    public double payRate;

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

```
public class Employee
{
    public string name;
    public double payRate;

    public Employee()
    {
    }

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

If you don't write a constructor, it is the same as having one with no parameters and no processing

Class with no constructor

```
public class Employee
{
    public string name;
    public double payRate

    public double GetPay()
    {
        double pay;
        pay = payRate * 40;
        return pay;
    }
}
```

Employee x;

x = new Employee();✓

x = new Employee("Fred", 7.50);✖

Class with constructor

```
public class Employee
{
   public string name;
   public double payRate;
   public Employee(string n, double rate)
   {
      name = n;
      payRate = rate;
   }
   public double GetPay()
   {
      double pay;
      pay = payRate * 40;
      return pay;
   }
}
```

Employee x;

x = new Employee(); ✖

x = new Employee("Fred", 7.50); ✔

Class with two constructors

```
public class Employee
{
   public string name;
   public double payRate;
   public Employee()
   {
   }
   public Employee(string n, double rate)
   {
      name = n;
      payRate = rate;
   }
   public double GetPay()
   {
      double pay;
      pay = payRate * 40;
      return pay;
   }
}
```

Employee x;

x = new Employee();✓

x = new Employee("Fred", 7.50); ✓

Demonstration

- What is an object?
- What is a class?
- What is a constructor method?
  - Multiple constructors