

# OO Principles

---

ELIZABETH BOURKE

# OO Principles

---

- ❑ A class describes a set of objects with the same data and behaviour.
- ❑ When you develop an object-oriented program, you create your own classes that describe what is important in your application.
- ❑ For example, in a student database you might work with Student and Course classes

```
namespace OOP1
{
    2 references
    class Product
```

```
{
    2 references
    private string Code { get; set; }
    2 references
    private string Description { get; set; }
    2 references
    private decimal Price { get; set; }

    0 references
    public Product() { }

    0 references
    public Product(string c, string d, decimal p)
    {
        this.Code = c;
        this.Description = d;
        this.Price = p;
    }

    0 references
    public void Print()
    {
        Console.WriteLine("Code" + this.Code);
        Console.WriteLine("Description" + this.Description);
        Console.WriteLine("Price" + this.Price);
    }
}
```

Data members - should be created private

Data Encapsulation

Constructor

Member function

# Data Members

---

An object stores its data in **data members**. These are variables that are declared inside the class.

When implementing a class, you have to determine which data each object needs to store. The object needs to have all the information necessary to carry out any member function call.

Private data members can only be accessed by member functions of the same class.

# Member Functions

---

The definition of a class declares its member functions. Each member function is defined separately, after the class definition.

```
public void Print()  
{  
    Console.WriteLine("Code" + this.Code);  
    Console.WriteLine("Description" + this.Description);  
    Console.WriteLine("Price" + this.Price);  
}
```

# Constructors

---

A **constructor** is a member function that initializes the data members of an object. The constructor is automatically called whenever an object is created. By supplying a constructor, you can ensure that all data members are properly set before any member functions act on an object.

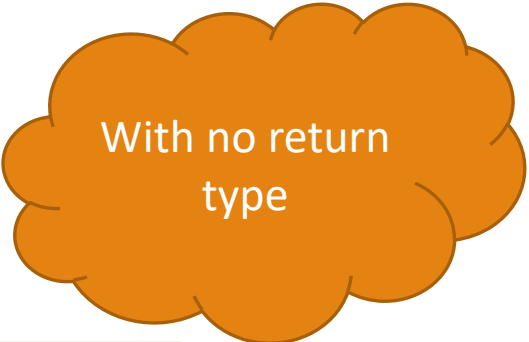
A constructor is called automatically whenever an object is created.

A default constructor has no arguments.

The name of a constructor is the same as the class name.

A class can have multiple constructors.

The compiler picks the constructor that matches the construction arguments.



With no return type

# Implicit and explicit parameters

---

The implicit parameter is a reference to the object on which a member function is applied.

`p.Print();`



Explicit parameters of a member function are listed in the function definition.

`Product p = new Product("DF123", "Power Cables", 12.99m);`



# Encapsulation

---

When you work with an object, you do not know how it is implemented. All you need to know is the member functions that you can invoke. The process of providing a public interface, while hiding the implementation details, is called encapsulation

Encapsulation is the act of providing a public interface and hiding implementation details.

Encapsulation enables changes in the implementation without affecting users of a class.



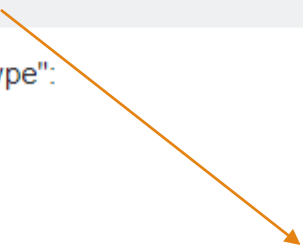
# Short cut {get; set;}

---

You could type "prop" and then press tab twice. That will generate the following.

```
public TYPE Type { get; set; }
```

Then you change "TYPE" and "Type":



You can also get the full property typing "**propfull**" and then tab twice. That would generate the field and the full property.

```
private int myVar;  
  
public int MyProperty  
{  
    get { return myVar;}  
    set { myVar = value;}  
}
```

## Class and object concepts

- An *object* is a self-contained unit that has *properties*, *methods*, and other *members*. A *class* contains the code that defines the members of an object.
- An object is an *instance* of a class, and the process of creating an object is called *instantiation*.
- *Encapsulation* is one of the fundamental concepts of object-oriented programming. It lets you control the data and operations within a class that are exposed to other classes.
- The data of a class is typically encapsulated within a class using *data hiding*. In addition, the code that performs operations within the class is encapsulated so it can be changed without changing the way other classes use it.
- Although a class can have many different types of members, most of the classes you create will have just properties, methods, and constructors.

## Exercise-The Student Class

---

The Student should have a K number, First name, Surname, Address, Registered course

- Create a class to store the above student data.
- Give the class full data encapsulation, and a method called print that will print the all the student details to the console.
- Instantiate 4 instances of the student class with test data and run the print method on each object.