Object Orientated Programming

What is OOP?

LEXICON

- An Object is a ”thing”
- Objects have properties and methods
- Methods effect or manipulate the data
- Objects have or implement certain characteristics –
- **Encapsulation**, **Polymorphism** and **inheritance**
- Accsess modifiers. Public & Private

```
public class MyClass
{
        string someInternalValue;

        public string SomeProperty { get; set; }

        public string SomeMethod (string possibleInputVar)
        {
                return “Some Value”;
        }
}
```

OOP

LEXICON

```
...
public SomeProperty
{
        get{  someInternalVar = value; }
        set{ return someInternalValue; }
}
```

public string SomeMethod()

public string SomeMethod(string SomeInput)

public void SomeMethod()

Encapsulation

LEXICON

- Ensures an object group all necessary properties and methods for a particular object. E.g a **Customer** might have **Name** and **Address** properties – abd these would be defined within the Customer object itself.

- Objects can contain other objects – for example a **Customer** might contain an **Address** object that has **Street Town** and **post code** properties.

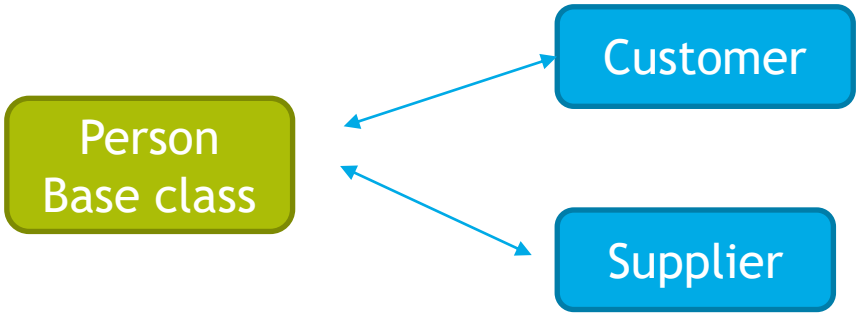
Address

Customer

Inheritance

LEXICON

- Objects can ’extend’ other objects
- A **Person** object may define common properties and methods. A **Customer** may inherit that Person object and add new properties and methods
- Resultant Object contains all properties and methods of both the ’base and the new object.




```
Public class Person{
   Public string name;
}

Public class Supplier: Person{
  Public string Website;
}
```

Polymorphism

LEXICON

- Two objects may have the same properties and methods but implement them diffrently
- Use of *interfaces* to define properties and methods – signature
- Any object that 'implement' the interface can be freely swapped in and out for each other without breaking the code

```
Public interface IResult{
  Int Compute (int first, int second)
}

Public class AddFunction : IResult{
  Int Compute (int first, int second){
    Return first + second;
  }
}

Public class SubtractFunction : IResult{
  Int Compute (int first, int second){
    Return first - second;
  }
}
```

```
IResult r;

If(mode=="Add"){
        r = new AddFunction();
}
If(mode=="Subtract"){
        r = new SubtractFunction();
}
Int answer = r.Compute(5, 5);
```

Namespace

LEXICON

How do a post man find his address?

35 Blogger Street,
Some town,
England

Using Objects / Scope

LEXICON

- MyClass myvarref = new MyClass();
- Garbage collector

- Scope:

```
GlobalObject globalObject = new globalObject();

If(a==b)
{
        ScopeObject scopeobject = new ScopeObject();
        scopeobject.candosomething();
        globalobject.candosomthing();
}
```

Constructors and Deconstructors

LEXICON

- Contructor:

```
Public class MyClass{
        Int globalvar;
        Public MyClass(int someparameter)
        {
                this.globalvar = someparameter;
        }
        ...other methods and properties
}
```

- Deconstructors:

```
Public class MyClass{
        ~ MyClass()
        {
                //any internal cleanup here
        }
}
```