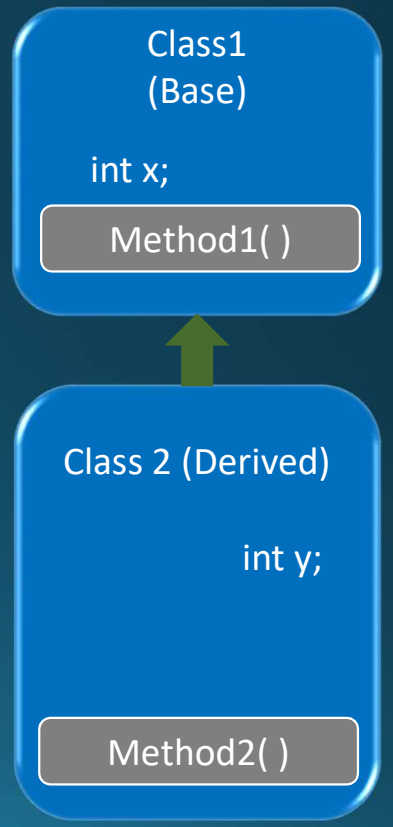
Inheritance

- Declare a class to be inherited from another class

```
public class Class1
    {
        public int x;
        public void method1()
        {
            Console.WriteLine("x={0}", x);
        }
     }
    class class2:Class1
        {
            public int y;
            public void method2()
            {
                method1();
                Console.WriteLine("y={0}", y);
            }
        }
```



Inheritance (cont.)

- Access Modifiers effect
  - protected access modifier

```
public class Class1
{
   protected int x;
}
public class class2:Class1
{
    public void method2()
    {
            x=10;
    }
}
….
static void Main(string[] args)
{
    Class1 c1 = new Class1();
    c1.x = 10;//error
```

Inheritance (cont.)

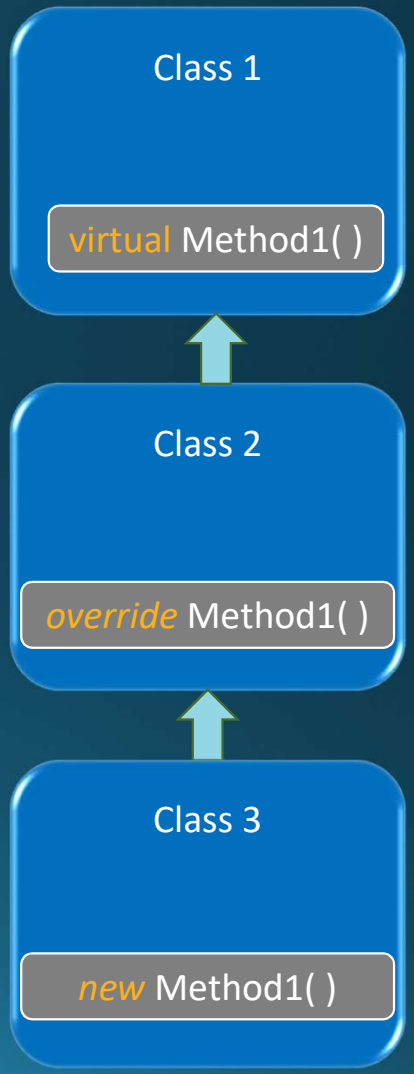
- Sequence of constructors
- *base* keyword
  - Sequence of constructors to be called
  - Explicitly calling specific constructor through base
- *sealed* keyword

Inheritance(cont.)

- *Reference* To parent class and object to child class

```
class1 c1 = new class2();
class1 c1 = new class3();
```

- Method Override (Polymorphism)
  - *virtual* modifier
  - *override* modifier
  - *new* modifier (hide parent function)
- is , as operators w21



w21

```
if (emp is Human)
{
}
```

wael, 1/13/2017

Inheritance(cont.)

- Object Class methods

Method	Description
public virtual bool Equals (object o)	if reference type check reference equality if value type check value(if different type return false even value is equal)
public Type GetType()	object type not reference type
public virtual string ToString()	Return a string (default return type as a string)
public virtual void Finalize()	implemented through destructor
public static bool ReferenceEquals (object a , object b)	check reference equality

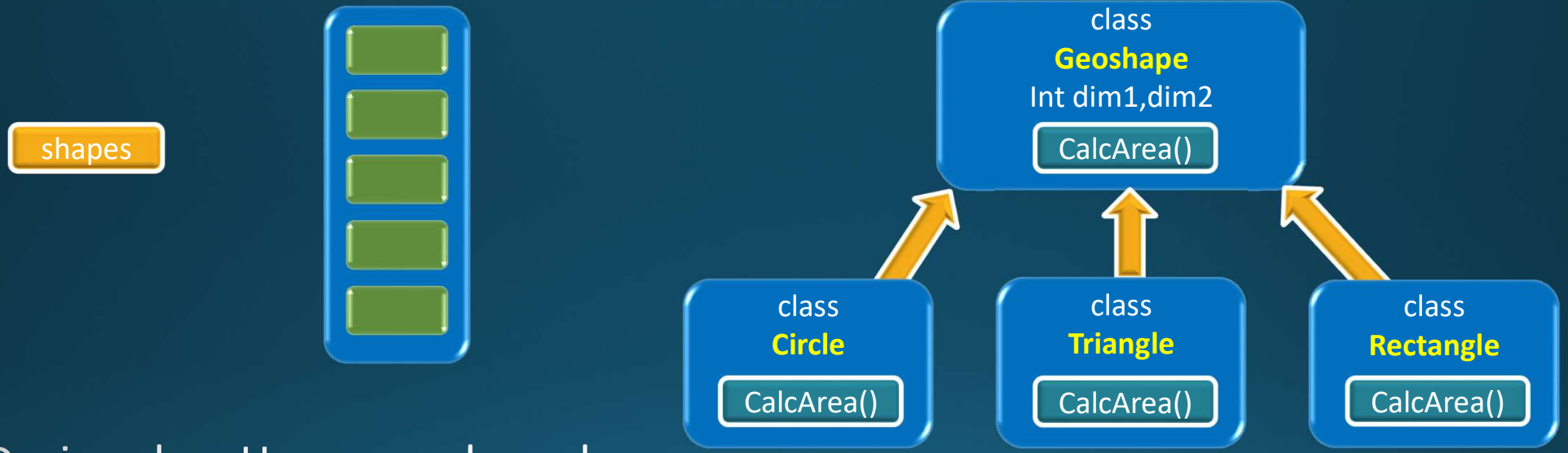
Extension Methods

- Adding methods to existing types
- Define and call Extension Method
  - Define static class
  - Define static Method
    - First parameter ➔ type that it will operate on preceding with ***this*** keyword
  - Call the method as if it is member Method (through object of the type)

```
Using myextensions
string s = "123";
int x = s.returnInt32();
Extensionclass.returnInt32(s);
Console.WriteLine(x);
```

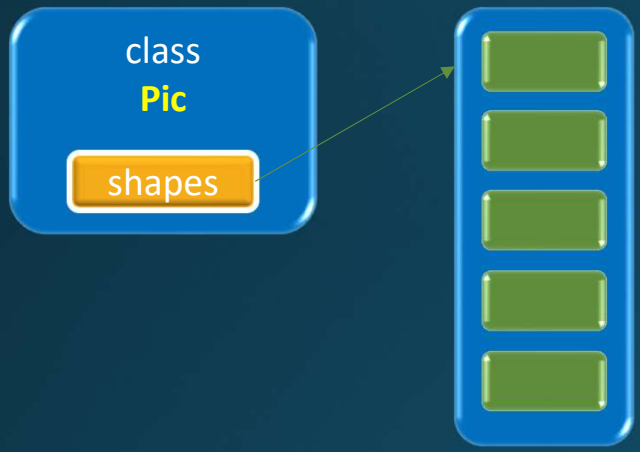
```
namespace myextensions
{
 static class extensionclass
 {
   public static int returnInt32(this string s)
    {
            int r = int.Parse(s);
            return r;
    }
 }
}
```

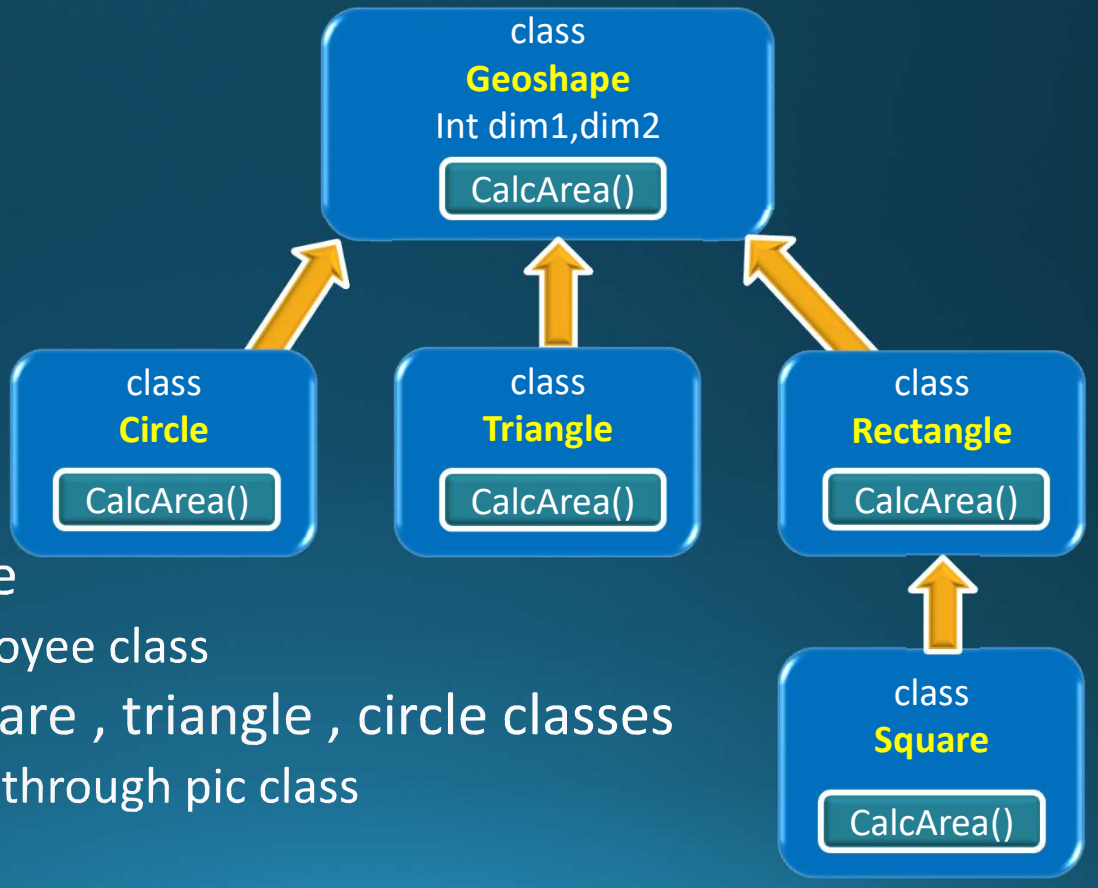
Assignments



- Design class Human and employee
  - Override tostring( ) method in employee class
- Design GeoShape , rectangle , square , triangle , circle classes
  - And calculate areas individual then through Array

Assignments





- Design class Human and employee
  - Override tostring( ) method in employee class
- Design GeoShape , rectangle , square , triangle , circle classes
  - And calculate areas individual then through pic class