

# Polymorphism

## 1. MathOperation

Create a class **MathOperations**, which should have 3 times method **Add()** . Method **Add()** have to be invoked with:

- Add(int, int): **int**
- Add(double, double, double): **double**
- Add(decimal, decimal, decimal): **decimal**

You should be able to use the class like this:

Startup.cs
<pre>public static void Main() {     MathOperations mo = new MathOperations();     Console.WriteLine(mo.Add(2, 3));     Console.WriteLine(mo.Add(2.2, 3.3, 5.5));     Console.WriteLine(mo.Add(2.2m, 3.3m, 4.4m)); }</pre>

## Examples

Output
5 11 9.9

## Solution

Created MathOperation class should look like this:

```
public int Add(int a, int b)
{
    return a + b;
}

public double Add(double a, double b, double c)
{
    return a + b + c;
}

public decimal Add(decimal a, decimal b, decimal c)
{
    return a + b + c;
}
```

## 2. Animals

Create a class Animal, which hold two fields:

- name: string
- favouriteFood: string

Animal have one virtual method **ExplainSelf(): string**.

You should add two new classes **Cat** and **Dog**. There **override ExplainSelf()** method by adding concrete animal sound on new line. (Look at examples below)

You should be able to use the class like this:

Startup.cs
<pre>Animal cat = new Cat("Pesho", "Whiskas"); Animal dog = new Dog("Gosho", "Meat");  Console.WriteLine(cat.ExplainSelf()); Console.WriteLine(dog.ExplainSelf());</pre>

## Examples

Input	Output
	I am Pesho and my fovourite food is Whiskas MEEOW I am Gosho and my fovourite food is Meat DJAAF

## Solution

```
public class Animal
{
    2 references
    public string Name { get; protected set; }

    2 references
    public string FavouriteFood { get; protected set; }

    2 references
    protected Animal(string name, string favouriteFood)
    {
        this.Name = name;
        this.FavouriteFood = favouriteFood;
    }

    4 references
    public virtual string ExplainSelf()
    {
        return $"I am {this.Name} and my favourite food is {this.FavouriteFood}";
    }
}
```

```

public class Cat : Animal
{
    0 references
    public Cat(string name, string favouriteFood) : base(name, favouriteFood)
    {
    }

    4 references
    public override string ExplainSelf()
    {
        return base.ExplainSelf() + Environment.NewLine + "MEEOW";
    }
}

```

### 3. Shapes

Create class hierarchy, starting with **abstract** class Shape:

- **Abstract methods:**
  - CalculatePerimeter(): double
  - CalculateArea(): double
- **Virtual methods:**
  - Draw(): string

Extend Shape class with two children:

- **Rectangle**
- **Circle**

Each of them need to have:

- **Fields:**
  - height and width for Rectangle
  - radius for Circle
- **Encapsulation for this fields**
- **Public constructor**
- **Concrete methods for calculations (perimeter and area)**
- **Override methods for drawing**