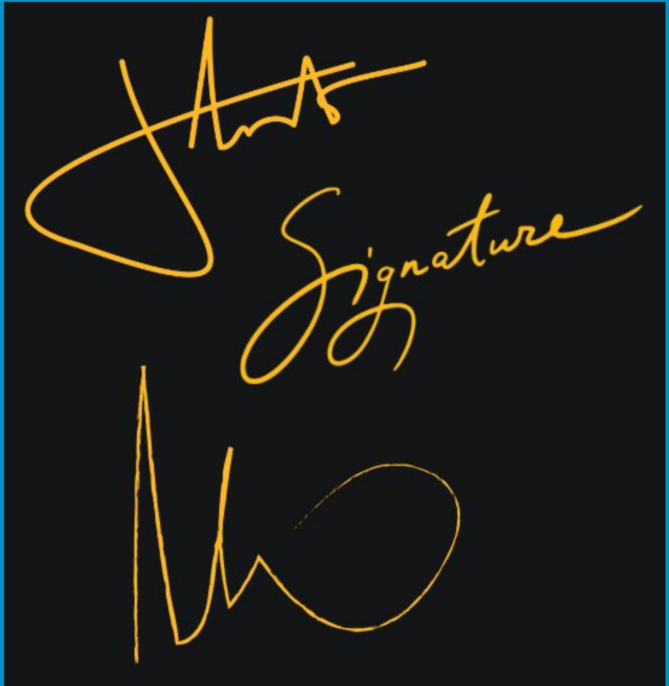
Method Overloading

Same Method Name

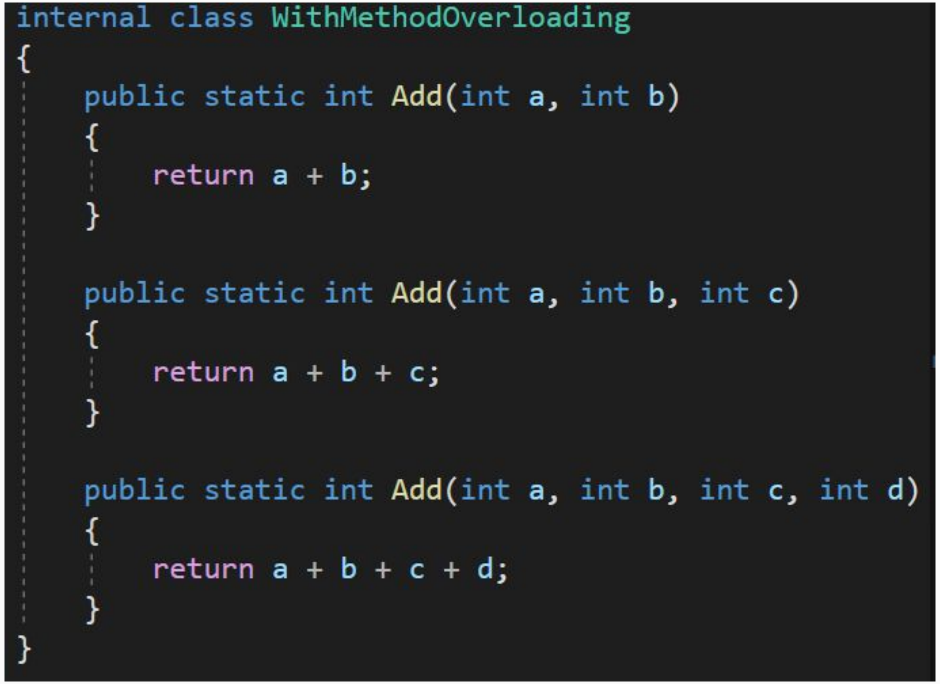
Method Overloading

Same method name, but different signatures



Method Signature

- A method’s signature is based upon the type and number of parameters

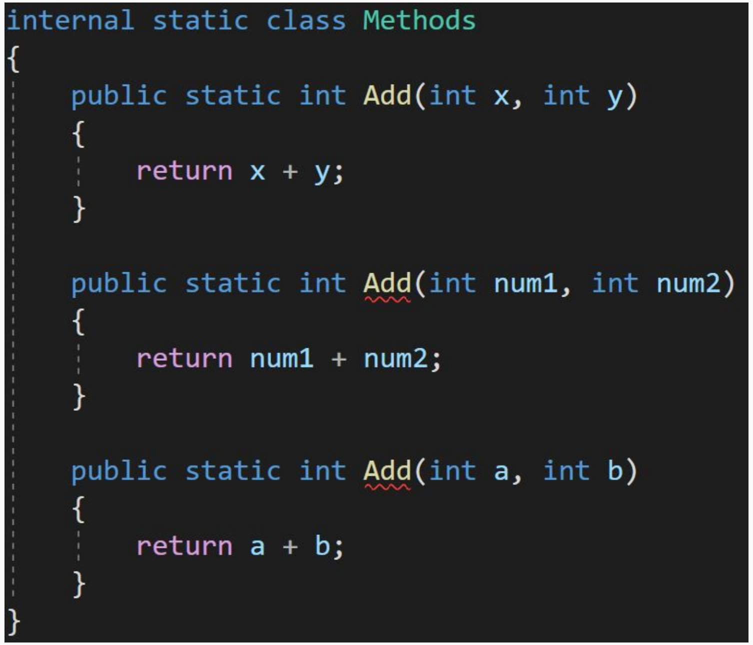


```
internal class WithMethodOverloading
{
    public static int Add(int a, int b)
    {
        return a + b;
    }

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

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

How NOT to overload methods

A method CANNOT be overloaded by changing the parameter's names!

Invalid

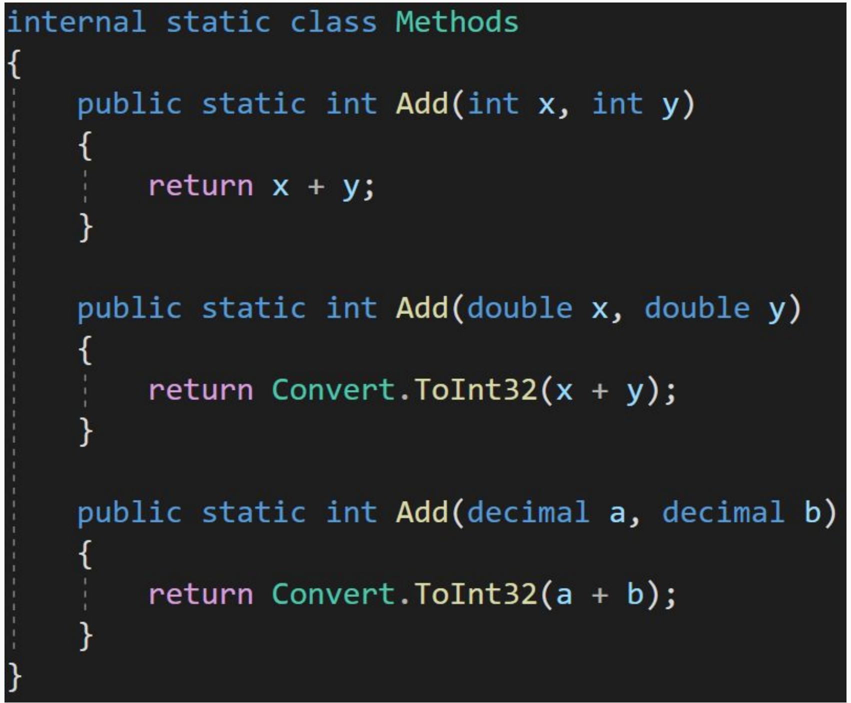


```
internal static class Methods
{
    public static int Add(int x, int y)
    {
        return x + y;
    }

    public static int Add(int num1, int num2)
    {
        return num1 + num2;
    }

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

A Method's Signatures

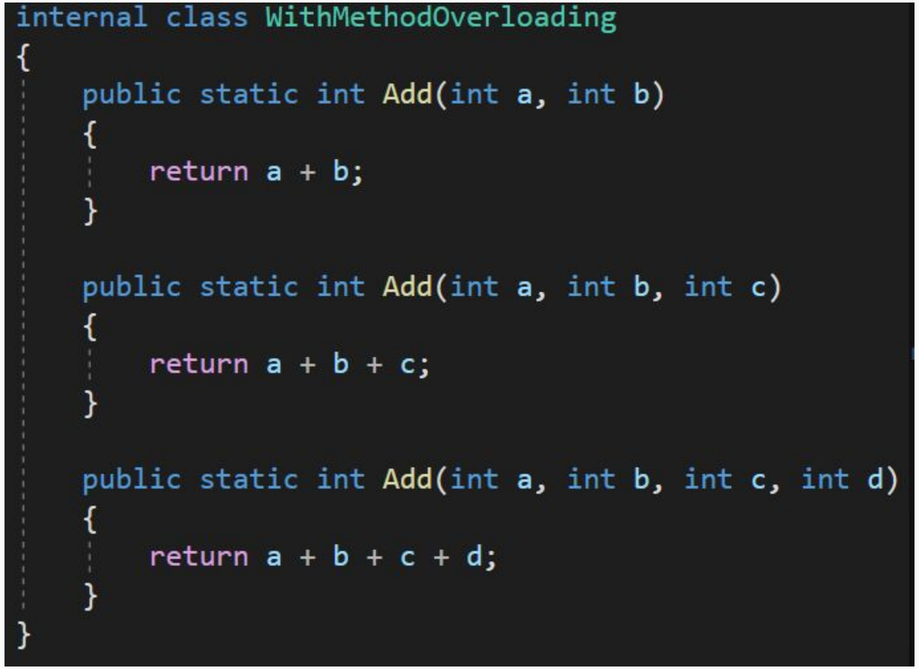
The signature is determined by:

a) The **number** of parameters and
b) The **data type** of the parameters


```
internal static class Methods
{
    public static int Add(int x, int y)
    {
        return x + y;
    }

    public static int Add(double x, double y)
    {
        return Convert.ToInt32(x + y);
    }

    public static int Add(decimal a, decimal b)
    {
        return Convert.ToInt32(a + b);
    }
}
```

A Method's Signature continued

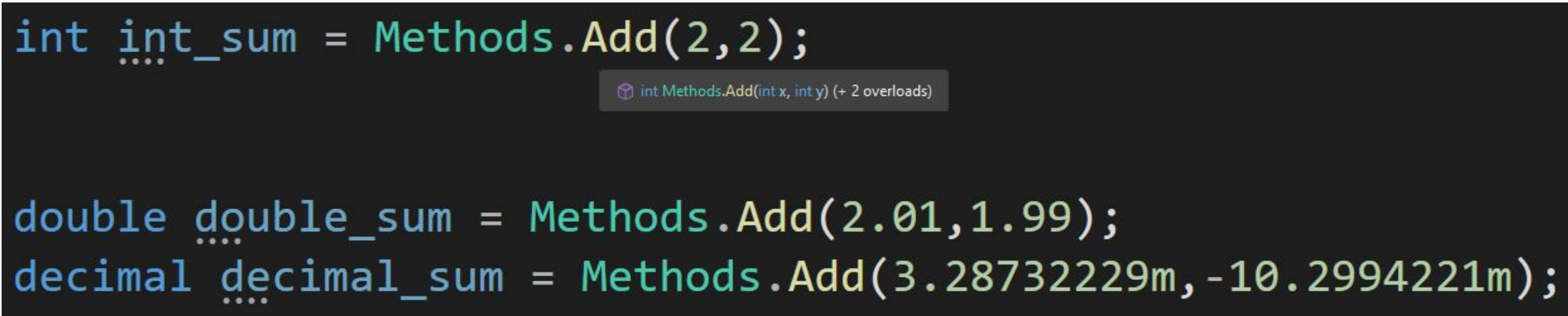
Note: It's best practice to use the same name and position for all of your parameters!



```
internal class WithMethodOverloading
{
    public static int Add(int a, int b)
    {
        return a + b;
    }

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

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

Method Resolution

When you **invoke** or call a method the correct method to run will be determined by the arguments that get passed in a process known as **method resolution**.



```
int int_sum = Methods.Add(2,2);
               int Methods.Add(int x, int y) (+ 2 overloads)

double double_sum = Methods.Add(2.01,1.99);
decimal decimal_sum = Methods.Add(3.28732229m,-10.2994221m);
```

Method Overloading Takeaways

- To overload a method, you keep the name but change the signature
- The signature is determined by:
    - The **number** of parameters and
    - The **data type** of the parameters
- Parameter names should remain the same and in the same position/pattern
- Any **modifiers keywords** or the method's **return type** are not considered as part of the signature when method overloading