Methods

C# Programming

Methods in C#

- We have been using methods in C# for some time

  `WriteLine`

  `ReadLine`

  `Parse`

- These are all methods which we have been calling to do tasks for us

UNIVERSITY OF Hull

What is a method?

- A method is a block of code that you can refer to by its identifier
- Whenever you refer to the method the block of code that is in the method is executed for you
- Methods let you break a solution down into smaller parts

Our first method

```
static void doit ()
{
    Console.WriteLine ("Hello");
}
```

- This method is called `doit`
- The body just prints a message
- It does not return anything - `void`
- It does not work on anything - ( )

Using a method

```
using System ;

class MethodDemo {
    static void doit () {
         Console.WriteLine ("Hello");
    }
    static void Main () {
         doit();
         doit();
    }
}
```

UNIVERSITY OF Hull

Method Header

```
static void doit ()
{
    Console.WriteLine ("Hello");
}
```

- The header at the top of the method tells the compiler about the method:
  - The type of the value the method returns
  - The items that the works on

University of Hull

Method Body

```
static void doit ()
{
   Console.WriteLine ("Hello");
}
```

- The body of the method is a block of code
- Methods can contain many statements
- When the block is finished the method ends and control returns to the caller

Method parameters

- You can give a method parameters if you want to pass some information into it
- We have been doing this with `WriteLine` and `Parse`
- However, not all methods need something to work on
- It depends on what the method is being used for

A method with a parameter

```
static void silly (int i)
{
   Console.WriteLine ("i is :" + i);
}
```

- This method has the identifier `silly`
- It is passed a single integer parameter
- Within the method we can use the parameter like we would any other `int` variable

Calling silly

```
static void Main ()
{
        silly ( 101 ) ;
        silly ( 500 ) ;
}
```

- The call of `silly` must now include a parameter value for it to receive
- This must be an integer, since that is what we have said `silly` will expect

Return values

- A method can also return information to the caller
  - We have seen this with the `ReadLine` method
- To tell the compiler this we must identify the type of data the method will return
- The method must also contain a statement which returns the result

A useful method

```
static int add (int i, int j)
{
    return i + j;
}
```

- This method has the identifier add
- It is passed two integer parameters
- It returns the result of the two integers added together

The method type

```
static int add (int i, int j)
{
    return i + j;
}
```

- Up until now all our methods have been void, which means they return nothing
- If a method has a type the method must return a value of that type

The return keyword

```
static int add (int i, int j)
{
    return i + j;
}
```

- The return keyword is how a method returns a value
  - Void methods do not need a return as they do not return a value

Calling add

```
static void Main()
{
  int result = add (2, 2);
  Console.WriteLine( result );
}
```

- The method is called with two integer parameters and returns the integer result

UNIVERSITY OF Hull

Naughty Calls?

```
int r1 = add (2.0, 2);
int r2 = add (2); // will not compile
add (2, 2);
```

- If we get the parameters wrong we will get a compilation error
- However ignoring the result returned by a method does not cause an error

A Useful Method

```
static int ReadNumber(string prompt, int min, int max)
{
    int result = 0 ;
    do
    {
        Console.Write(prompt);
        string numberString = Console.ReadLine();
        result = int.Parse(numberString);
        if (result > max || result < min)
            Console.WriteLine("Please enter a value in the range " +
                                   min + " to " + max);
        else
            break;

    } while (true);
    return result;
}
```

Using A Useful Method

```
NoOfPlayers = ReadNumber("Enter number of players : ", 2, 4);
```

- This method can be used to read a value
- It is given a prompt string and the min and max values of the range
- It then returns with a result in that range

Summary

- Methods allow us to break a program into a smaller chunks and reuse code
- They are created outside the Main method
- A method can receive values to work on (parameter) and return a result
- The compiler will make sure that a method is called correctly when it is used