University of Hull

# How Programs Run

C# Programming

# Programs

- In the practical session last week we created a number of programs
- In this session we are going to look at each program and see how it works/what it does
- We are also going to investigate how programs store and work with data

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

- What would this program print?

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

- The first statement makes a variable called `first` and stores 1 in it

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

second: 2

- The second statement makes a variable called second and stores 2 in it

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

second: 2

third: 3

- The third statement makes a variable called `third` and stores 3 in it

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

second: 2

third: 3

result: 3

- The fourth statement makes a variable called `result` and stores `first + second` in it

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

second: 2

third: 3

result: 9

- The fifth statement multiplies the variable `result` by the variable `third`

# Program Execution Test

```
static void Main()
{
    int first = 1;
    int second = 2;
    int third = 3;
    int result = first + second;
    result = result * third;
    Console.WriteLine(result);
}
```

first: 1

second: 2

third: 3

result: 9

- The final statement prints out the value in the `result` variable

# Variables

- There are four variables in the program
  - `first`, `second`, `third` and `result`
- Each of these variables can hold a single integer value
  - We can create variables that hold other kinds of data (for example strings of text) by declaring them differently
  - We used the string type in the Greeter program when we stored the name of the user

# Strings and numbers

```
string number1Text = Console.ReadLine();
int number1 = int.Parse(number1Text);
```

- This is part of the sums program
- It contains two variables, one called `number1Text` and one called `number1`
- You can think of these as two different kinds of boxes

# Strings and numbers

number1Text: "1"

```
string number1Text = Console.ReadLine();
int number1 = int.Parse(number1Text);
```

- If we run the program and the user types "1" then it creates a string variable called number1Text that holds the string "1"

# Strings and numbers

number1Text: "1"

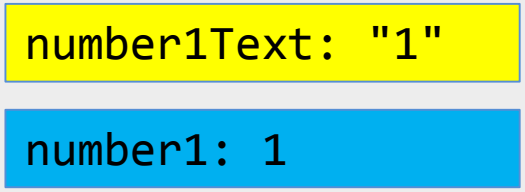
number1: 1

```
string number1Text = Console.ReadLine();
int number1 = int.Parse(number1Text);
```

- The second statement uses a magic method called Parse that converts a string of text into a numeric value
- The result is that we have an integer variable called number1 that holds the value 1

# Strings and Numbers

- At first glance this seems very confusing
- Both boxes seem to hold the same thing
  - They both hold 1
- The thing to remember is that one is text, and the other is a number



# Strings and Explosions

- The string box can hold any string of text
- For example you could type the word “one” into the program
- This gives the Parse method a problem, as it can’t convert this into a number

```
number1Text: "one"
```

```
number1: 1
```

# Strings and Explosions

- If the Parse method is given a string that contains a numeric value it is very happy and just converts it into a number
- If it is given anything else it will crash your program
  - We will see how to fix this later in the course

```
number1Text: "one"
```

```
number1: 1
```

# Why does printing just work?

```
Console.WriteLine(result);
```

- There seems to be an asymmetry about the way that strings and integers work
  - I need to use Parse to convert a string into a number, but I don't need to do anything to convert a number into a string when I print it out
- This is not because the conversion doesn't have to be performed, it is because the conversion is automatic

# Why does printing just work?

```
Console.WriteLine(result.ToString());
```

- All data items in C# provides a `ToString` method which will convert the item into a string version of themselves
- You can call this method yourself if you like – as shown above
- However, when printing it is called automatically

# Why isn't Parse called automatically?

- Now we have another mess
  - `ToString` is sometimes called automatically
  - `Parse` is never called automatically
- You could argue that this is wrong
  - You might be right (in Visual Basic this kind of conversion happens automatically)
  - This is just how C# works

# Summary

- The program is made up of the statements that get obeyed when it runs
  - Each statement is obeyed in turn
- We can make variables that hold data
- We have methods, such as Parse and ToSring, that can be used to convert from one type to another