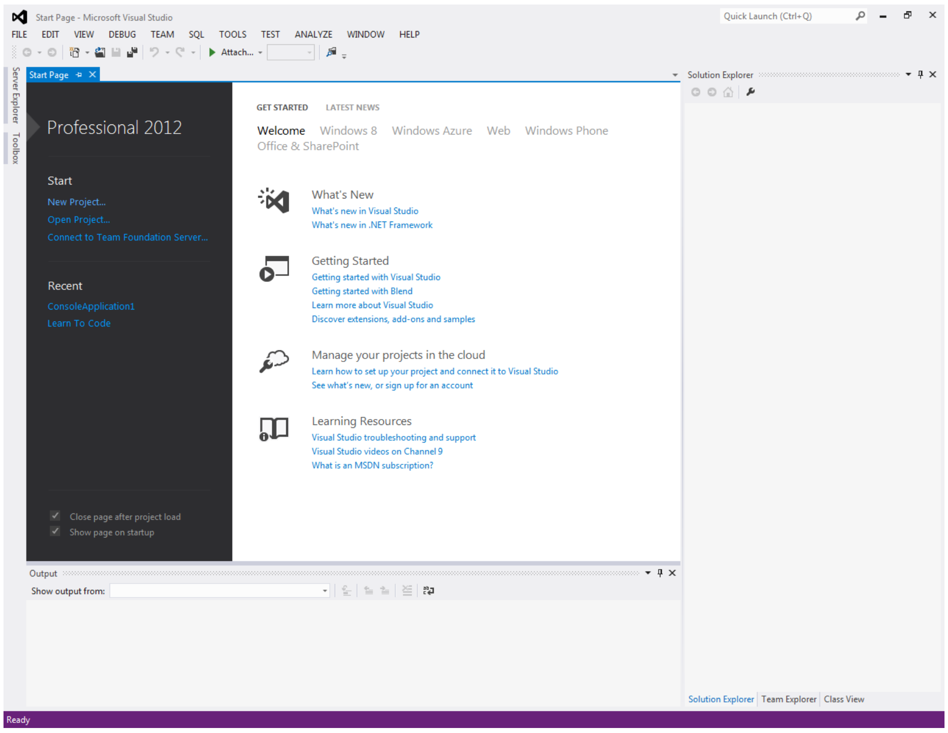
EXERCISE 1

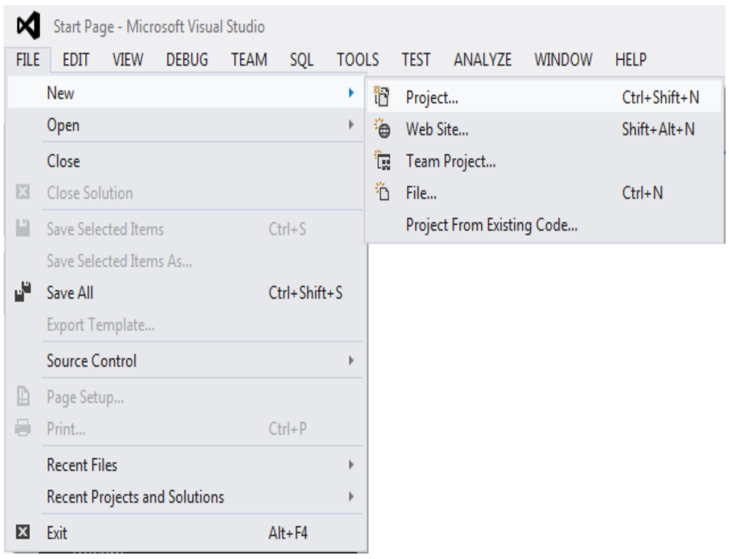
The tasks in the first part of this section are step by step instructions to guide you in the practical application of the theory of programming. This is followed by a number of tasks for you to complete by applying the knowledge that you have previously gained.

TASK 1: STARTING WITH VISUAL STUDIO: HELLO WORLD!

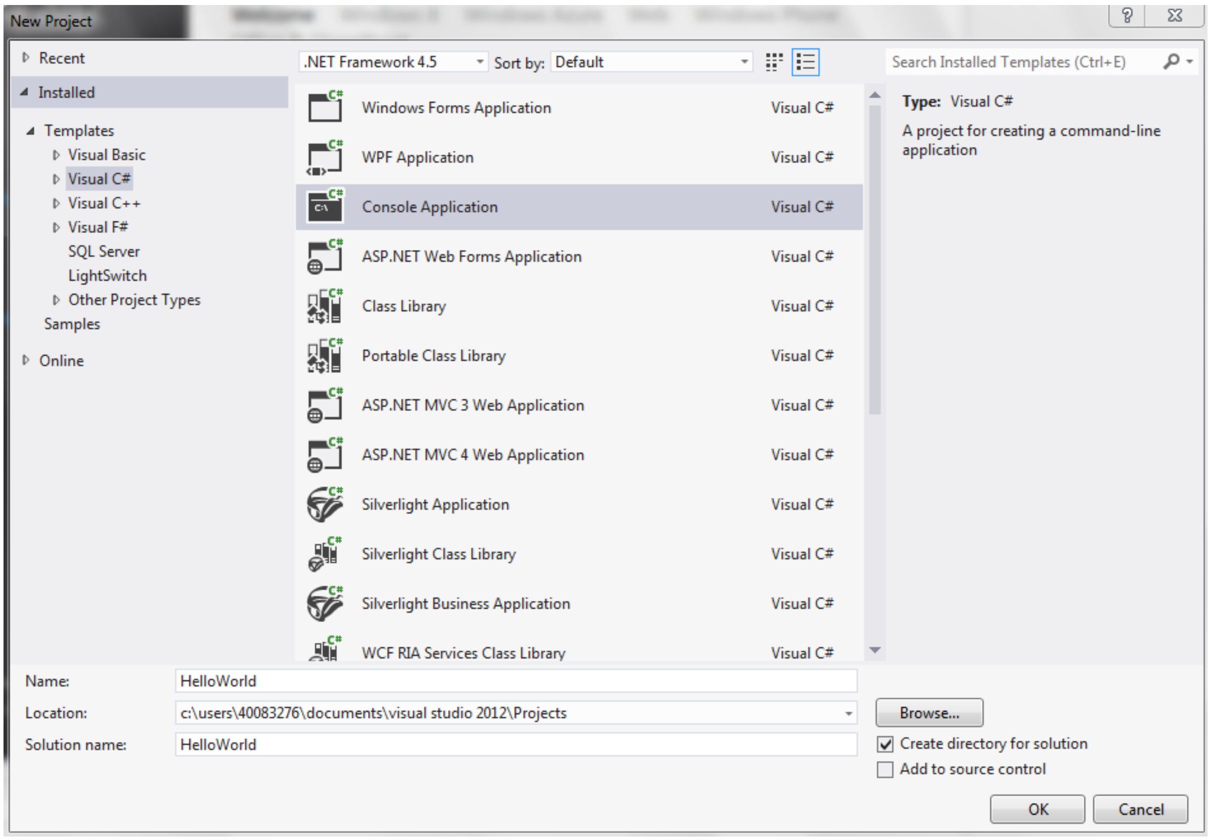
Step 1: Run Visual Studio. It will open to the start page from which you can create a new project.


Start Page - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Start Page
Solution Explorer
Professional 2012
Start
New Project...
Open Project...
Connect to Team Foundation Server...
Recent
ConsoleApplication1
Learn To Code
Close page after project load
Show page on startup
GET STARTED
LATEST NEWS
Welcome Windows 8 Windows Azure Web Windows Phone Office & SharePoint
What's New
What's new in Visual Studio
What's new in .NET Framework
Getting Started
Getting started with Visual Studio
Getting started with Blend
Learn more about Visual Studio
Discover extensions, add-ons and samples
Manage your projects in the cloud
Learn how to set up your project and connect it to Visual Studio
See what's new, or sign up for an account
Learning Resources
Visual Studio troubleshooting and support
Visual Studio videos on Channel 9
What is an MSDN subscription?
Output
Show output from:
Solution Explorer Team Explorer Class View
Ready

Step 2: Click on the **File** tab at the top left and then select **New > Project.** Note the key combination of Ctrl+Shift+N can also be used as a shortcut.


Start Page - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
New
Open
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project... Ctrl+Shift+N
Web Site... Shift+Alt+N
Team Project...
File... Ctrl+N
Project From Existing Code...

Step 3: A new pop up window will open in which we select our project type. On the left hand side from the Templates drop down menu choose **Visual C#** and then select **Console Application** as shown below. Then enter a suitable name for your project such as `Prac1Ex1` and click OK.



Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Forms Application	Visual C#
Class Library	Visual C#
Portable Class Library	Visual C#
ASP.NET MVC 3 Web Application	Visual C#
ASP.NET MVC 4 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
WCF RIA Services Class Library	Visual C#

Type: Visual C#

A project for creating a command-line application

Name: HelloWorld

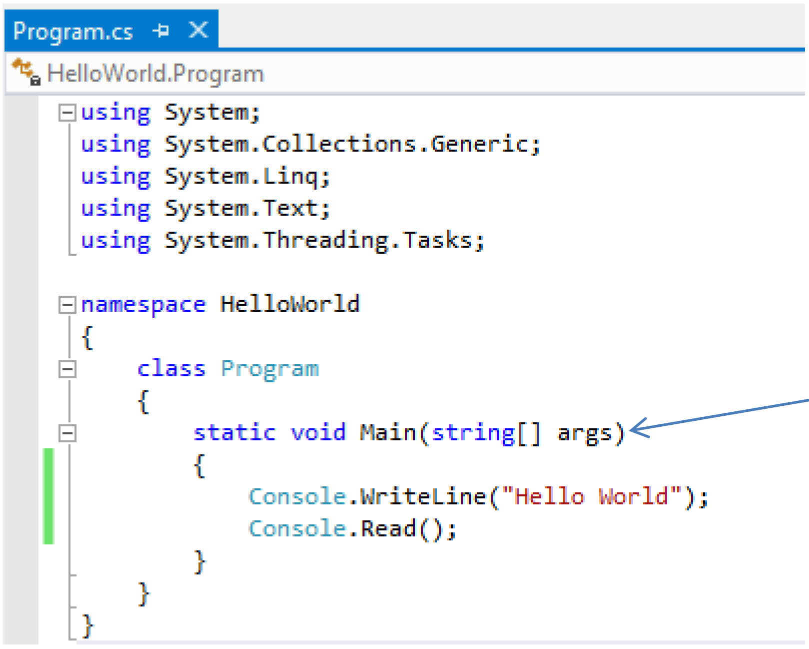
Location: c:\users\40083276\documents\visual studio 2012\Projects

Solution name: HelloWorld

☑ Create directory for solution

☐ Add to source control

Step 4: Your project has now been created and a class file called `Program.cs` will open. Note: there may already be some code in the window and you will only need to type two of the lines below. `Console.WriteLine` simply displays the text between the quotation marks in the console window.

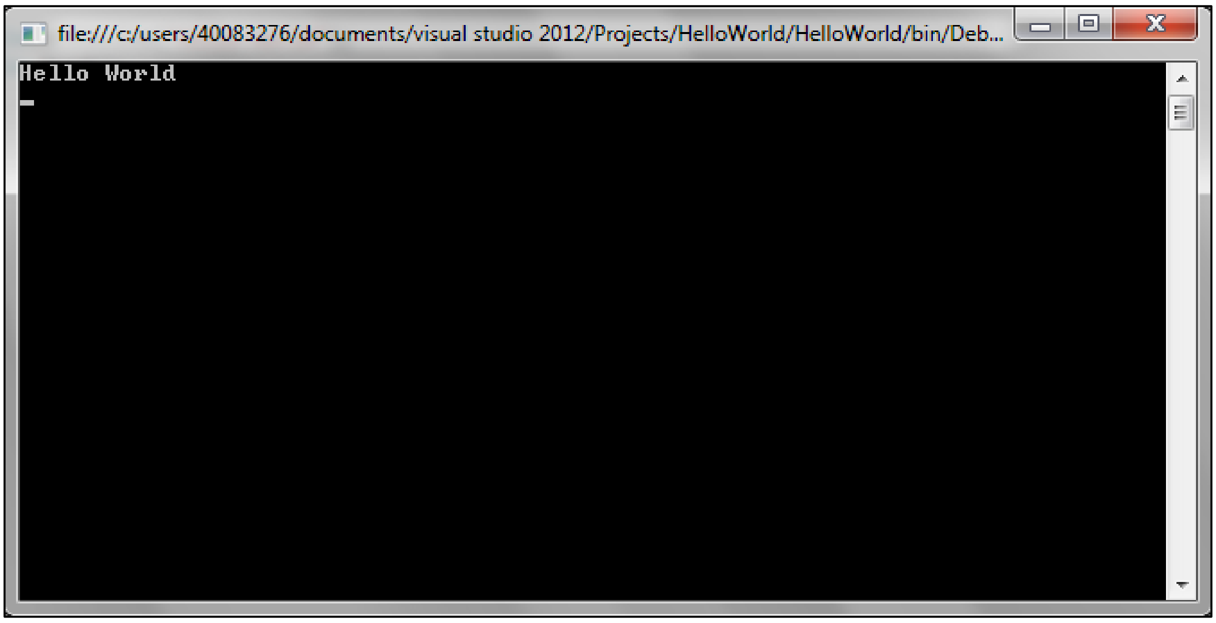


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
            Console.Read();
        }
    }
}
```

Note: This line indicates that this is our **main method,** it will allow our code within it to run.

Step 5: Save your work (CTRL+S) and then press the Start ▶ Start button on the toolbar. Your program will run and a console with open showing the results. Press any key to close the console.


file:///c:/users/40083276/documents/visual studio 2012/Projects/HelloWorld/HelloWorld/bin/Deb...

```
Hello World
_
```

You have created your first C# program!

TASK 2: PRINT OUT YOUR NAME USING THE CONSOLE WINDOW

We will update our Hello World program so that it takes input from the user, in this case your name, and displays it in the console output.

Step 1: Make an alteration to the message between the " " marks on the `Console.WriteLine` so that the user is asked for their name e.g.

```
Console.WriteLine("Hello, what is your name?");
```

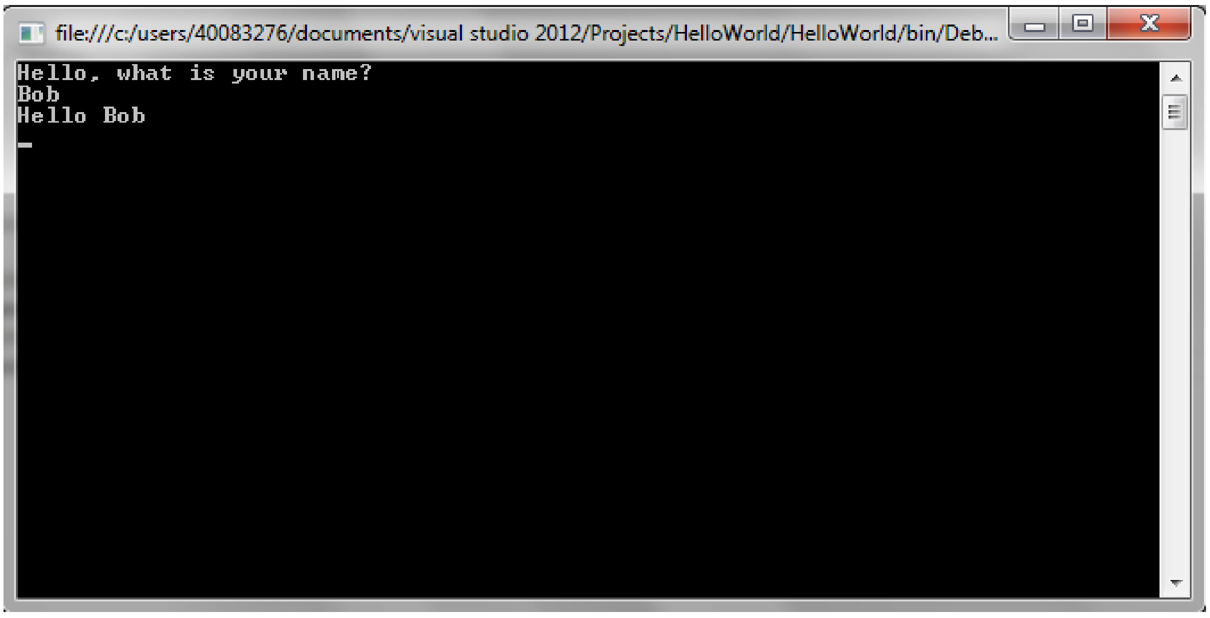
Step 2: Now you will need to declare a variable of type String to hold the user's name. It is good practice to give your variables names that accurately describe what they are to avoid any confusion. `Console.ReadLine` is used to take user input and will accept characters which are typed until the user presses enter. The characters are assigned to the variable `myName`.

```
String myName=Console.ReadLine();
```

Step 3: Using another instance of `Console.WriteLine` you can display your name by referring to the variable used to store it.

```
Console.WriteLine("Hello " + myName);
```

Step 4: Save your project (Ctrl+S) and click start ▶ Start to see the results. You will be prompted to enter your name. Press the enter/return key when you are finished entering.

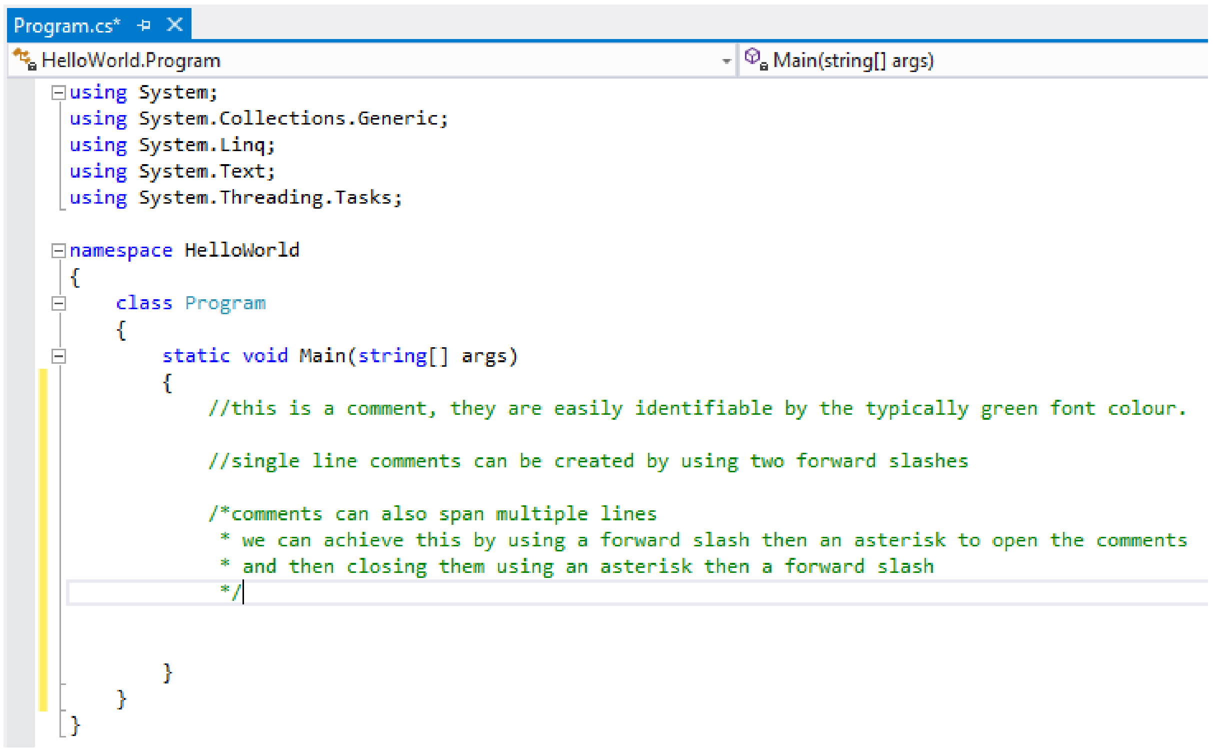

file:///c:/users/40083276/documents/visual studio 2012/Projects/HelloWorld/HelloWorld/bin/Deb...

```
Hello, what is your name?
Bob
Hello Bob
_
```

Now you have experience with basic user input and output.

TASK 3: COMMENTS

Comments are annotations or notes written by the programmer which explain the code to another user and make it easier to understand. They can be very significant to a human but are generally ignored by the computer in the compilation of the program. In terms of our first program, comments may not seem necessary but as projects get larger they become more important and it is good practice to get into the habit of using them. Examples are shown below but be sure to try it out for yourself.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            //this is a comment, they are easily identifiable by the typically green font colour.

            //single line comments can be created by using two forward slashes

            /*comments can also span multiple lines
             * we can achieve this by using a forward slash then an asterisk to open the comments
             * and then closing them using an asterisk then a forward slash
             */


        }
    }
}
```

EXERCISE 2

TASK 4: GETTING TO GRIPS WITH VARIABLES

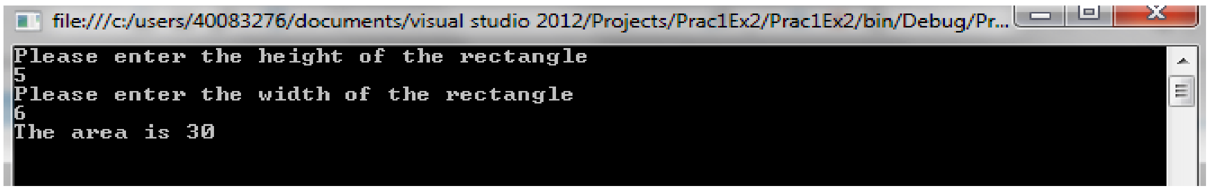
Now it is time to test some of the knowledge you have gained involving creating variables and data types.

Create a new project (C# Console Application) as shown in Exercise 1. Name it `Prac1Ex2` and click Ok.

In your program create three `int` variables and call them width, height and area. We will need to make use of the `Console.ReadLine( )` to initialise the height and width variables with values input by the user. However as the `ReadLine()` accepts characters rather than numbers we will need to convert our input to an `int` so that we can perform mathematical operations on it. We simply use a method that has been built in type conversion method called `Convert` followed by a period and '`ToInt32`' specifies what type we want to convert to.

```
width = Convert.ToInt32(Console.ReadLine());
```

This will assign the next number entered to our variable width. Do the same for height so both the variables have been assigned values. Find the area of the shape by multiplying the two variables and storing the result of the operation in the variable area. Finally print area out using `Console.WriteLine(area);`


file:///c:/users/40083276/documents/visual studio 2012/Projects/Prac1Ex2/Prac1Ex2/bin/Debug/Pr...

```
Please enter the height of the rectangle
5
Please enter the width of the rectangle
6
The area is 30
```

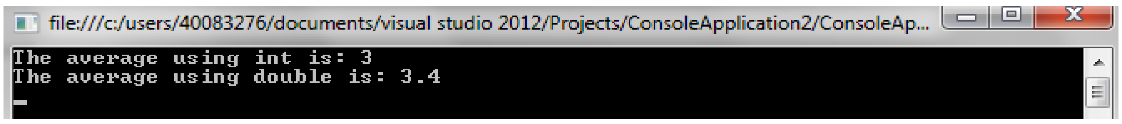
TASK 5: CASTING DATA TYPES

This exercise requires you to cast the result of an operation performed upon two integers to a double. Declare and initialise variables for two `ints` sum and count. Initialise these variables to 17 and 5 respectively. Then declare a double variable called `doubleAverage` and an `int` variable called `intAverage`.

Remember that the process for casting is to place the data type in parentheses before the operation that is being performed which in this case is sum/count. We will perform this operation and store the result in `intAverage` as well for reference. Using `Console.WriteLine(variableName)` write `doubleAverage` and `intAverage` to the output window to see the difference.

```
int sum = 17;
int count = 5;
int intAverage = sum / count;
double doubleAverage = (double)sum / count;
```

Example Output


file:///c:/users/40083276/documents/visual studio 2012/Projects/ConsoleApplication2/ConsoleAp...

```
The average using int is: 3
The average using double is: 3.4
_
```

TASK 6: FINDING THE SUM OF THE 10 POSITIVE INTEGERS

Write a program that prints the sum of the first ten positive integers: `1+2+…+10`.

TASK 7: MODIFIED HELLO WORLD PROGRAM

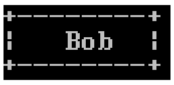
This is a slightly modified version of the `HelloWorld` program :

```
class HelloWorldError {
      public static void main(String[] args) {
            Console.WriteLine ("Hello World!);
            Console.WriteLine ("Hello","world");
      }
}
```

Each of the `Console.WriteLine` statements has an error. Fix the errors so that the program successfully compiles and runs. What were the errors?

TASK 8 : DISPLAYING YOUR NAME INSIDE A BOX

Write a program that displays your name inside a box on the screen, see example shown below.



```
+---------+
¦   Bob   ¦
+---------+
```

You should be able to do this with characters such as + – |

EXERCISE 3

TASK 9: METHODS

Step 1: Create a new project (C# Console Application) as shown in Exercise 1. Name it `Prac1Ex3` and click Ok.

Step 2: In the program class after the closing bracket for the main method you will need to define two methods called `messageOne` and `messageTwo`. These methods will be public and have no return type i.e. be void and accept no parameters. Each method will simply output a message to the console such as:

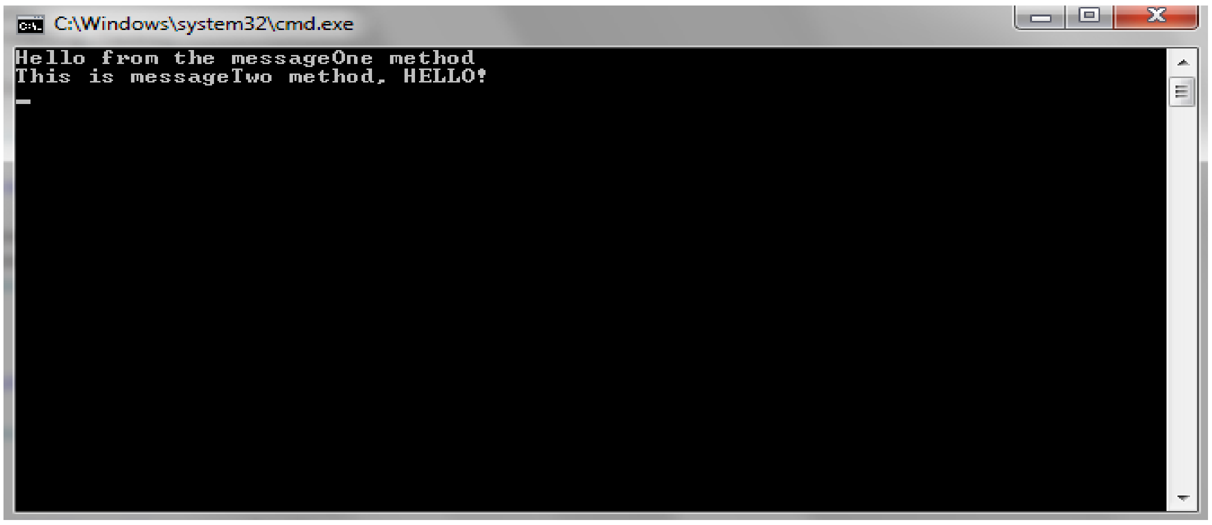
```
public static void messageOne()
{
    Console.WriteLine("Hello from the messageOne method");

}
```

The static keyword placed after the access modifier indicates that the method belongs to the class which defines it in this case our Program class.

Step 3: Now call your two methods in your main method by using their names followed by parentheses `()`.

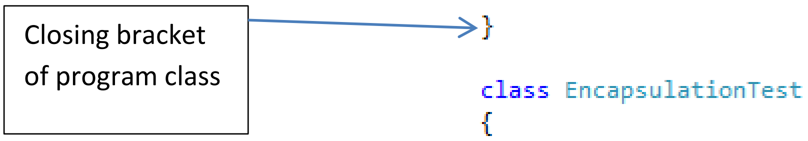
Example Output:



```
Hello from the messageOne method
This is messageTwo method, HELLO!
_
```

TASK 10: EXPLORING ENCAPSULATION

Step 1: Create a new class after the closing bracket for the program class and name it Encapsulation Test.



```
}

class EncapsulationTest
{
```

Step 2: Define two methods in the new class. The first will be a public method and have no return type or parameters, call it `publicMethod()`. The second method will be similar but will be private and thus called `privateMethod()`. Remember to add the static keyword after the access modifiers to indicate that the method belongs to the EncapsulationTest class.

```
<access modifier> static <return type> <method name> (parameters)

{

      Method body

}
```

Step 3: Output a message from each of your methods making sure to differentiate between which method is being called e.g.

```
Console.WriteLine("This is the public method from the EncapsulationTest class!")
```

Step 4: Call both of the methods in the main method. You should receive an error when trying to call the `privateMethod( )`. This is because of the accessibility level we assigned to the method by using the private keyword. How can this issue be solved without changing the access modifier of the method to public?

TASK 11: EXPLORING ENCAPSULATION

Write a program that will help the user decide whether they should drive their car to university or take the train. You know the one-way distance from your home to university and the fuel efficiency of your car(in miles per gallon). You also know the one way distance price of a train ticket. You assume the cost of petrol is £6.22 per gallon and car maintenance at 26.2 pence per mile. Write a program to decide which commute is cheaper. You should try to design this on paper first and then try to code it in C#.

Example for Mini Cooper:

- Fuel Efficiency: 52.3mpg
- Price of one way train ticket from Bangor to Belfast £5.40
- Cost of petrol per gallon £6.22 (136.9 pence per litre)
- Car maintenance 26.2 pence per mile
- Distance from home to university 15 miles
- Therefore the cost of driving should be £5.71(depending on rounding).