Programming 1: Tutorial 10

The aims for this tutorial are simple.
There is a small exercise that aims to introduce you to random numbers, once you have finished this you are then advised to start on Assessment 1.

The Correct Project Type

Don't forget to create the correct project:

Visual C++ Projects -> Win32 Console Project -> Name your project and select the relevant directory then click finish -> Click the applications tab and select 'Blank Project' option -> Click finish

Don't forget the Basic Structure of a C++ Program

```
// header files first
#include <iostream>
// namespaces next
using namespace std;
// globals
const unsigned int PI = 3.14;
// function prototypes
int GetAge();

int main () // then main
{
	// stuff in here

	return 0; // at the very end just before...
} // .. the terminating parentheses

// function definitions down here
int GetAge()
{
	int userAge = 0;

	cout << "Your age please ";
	cin >> userAge;

	return userAge;
}
```

Examples

Don't forget to look at the example code supplied at the end of this tutorial. Compile and run the examples and make sure you understand what is occurring.

Exercises

To complete these exercises you may need to reference the notes, either online or use the printed ones provided.
There is also sample code available, it has been included in hard copy format at the back of the tutorial notes.

Classes: Exercise 1

The aim here is introduce you to classes by getting you to follow an example.

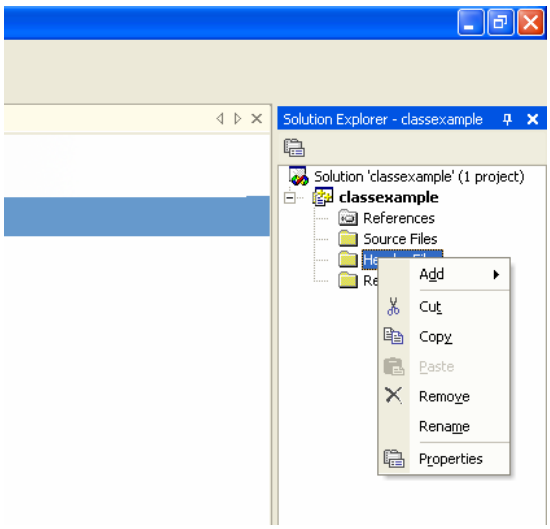
We are going to be coding and then using the Player class. The class will be defined in one file that will then be included in our main file.

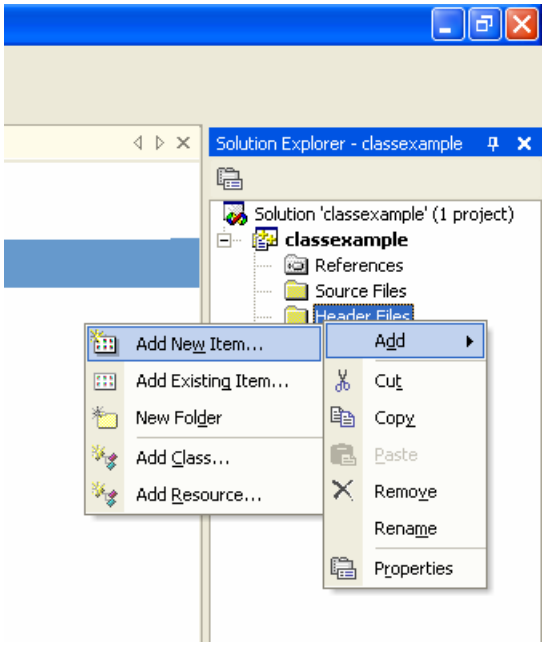
Starting Up

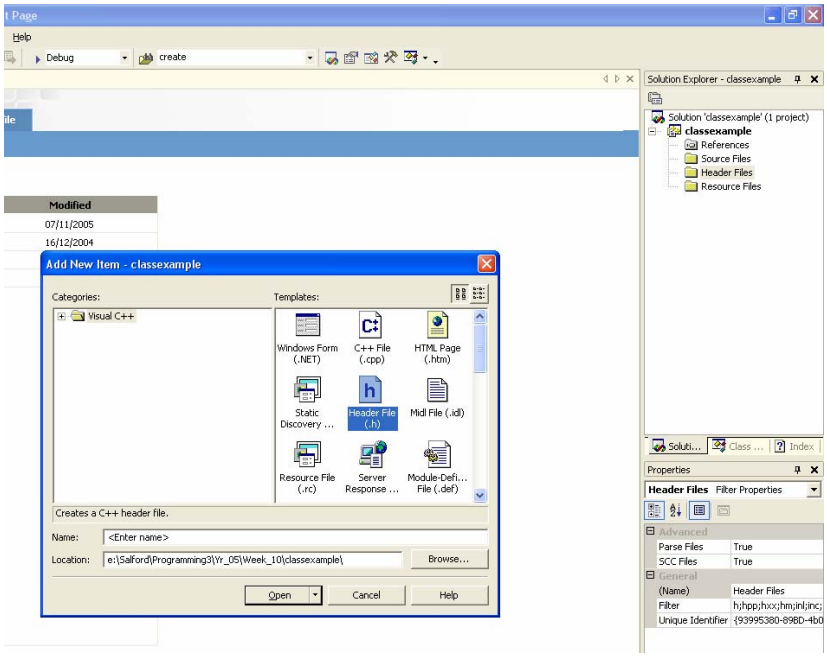
The first thing to do is to make a new project as normal.

Adding a Header File

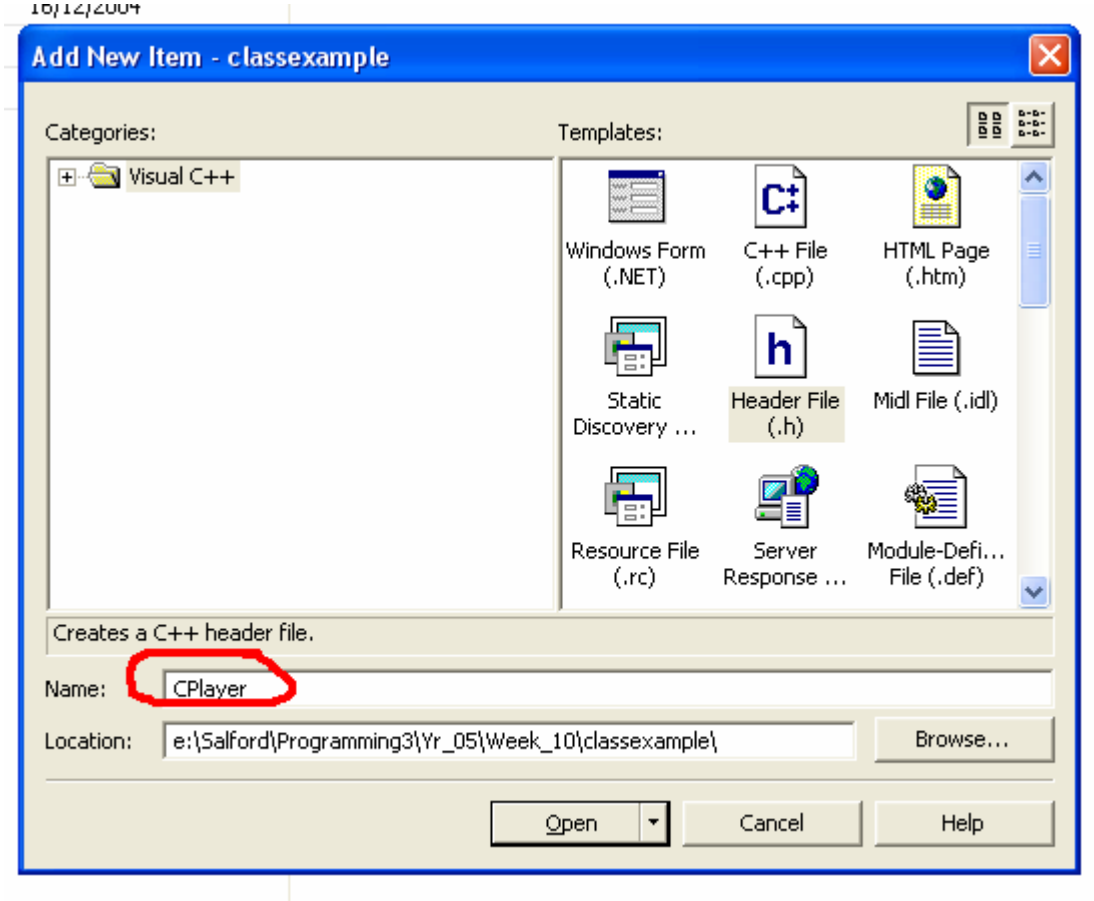
Then we add our header file:







Don’t forget to give your header file a name. As the header file is containing a class the name it has should be the name you are going to assign your class. In our case this is CPlayer. The files full name will be ‘CPlayer.h’.



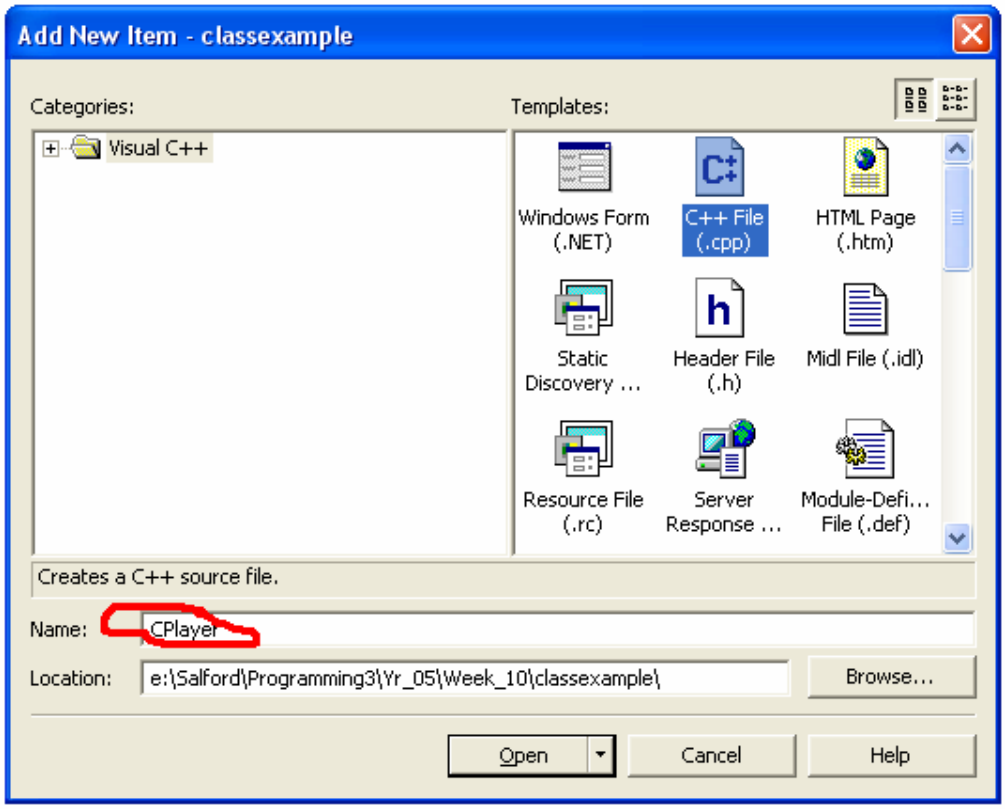
Coding our Class
We now add the following code to CPlayer.h, which is our CPlayer header file – the one created above.

```
class CPlayer
{
public:
	CPlayer(); // constructor
	// member methods
	void AlterHealth(int H);
	void AlterArmour(int A);
	int GetHealth() const;
	int GetArmour() const;
	void DisplayStats() const;

private:
	// member data
	int m_iHealth;
	int m_iArmour;
};
```

We now have a class that has 2 member data and that has a few member functions.

Adding a source file
We now want to add a source file. We do this as we would normally add a source file; however we want to call this file CPlayer as well. The files full name will be ‘CPlayer.cpp’.



Including Relevant Header Files

In our newly created source file we now need to define our classes functions. First though we need to tell the source file which header files it is going to need.

At the top of the CPlayer source file add the following lines:

```
#include <iostream>
// user defined header
#include "CPlayer.h"
```

With out including the iostream header file we won't be able to perform any input/output operations.
We need to include CPlayer.h so that the compiler associates the two files and is able to correctly define the class.

Defining our Class Methods

Now we have included relevant header files we need to define our class methods. Class methods are generally defined in the order they appear in the class declaration.

The constructor

This is generally defined at the top of the source file. We want ours to initialise the member data, health to 100, armour to 0, and to print our something that shows that the values have been initialised.

```
CPlayer::CPlayer()
{
```

```
	m_iHealth = 100;
	m_iArmour = 0;

	std::cout << "\nConstructor Initialisation:\n"
		<< "\nHealth = " << m_iHealth
		<< "\nArmour = " << m_iArmour << std::endl;
}
```

The remaining class methods
Our first class method looks like this:

```
void CPlayer::AlterHealth(int H)
{
	m_iHealth += H;
}
```

The method `AlterHealth(int H)` alters health my the amount provided when the function is called.

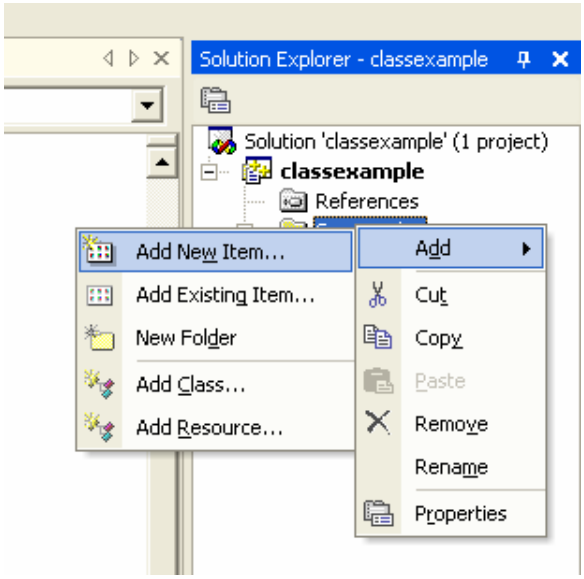
Code the rest
It is now up to you to code the rest of the methods. They all go in the source file we are currently writing in. They all have there return type specified and then the class name, `::`, and finally the rest of the function prototype.

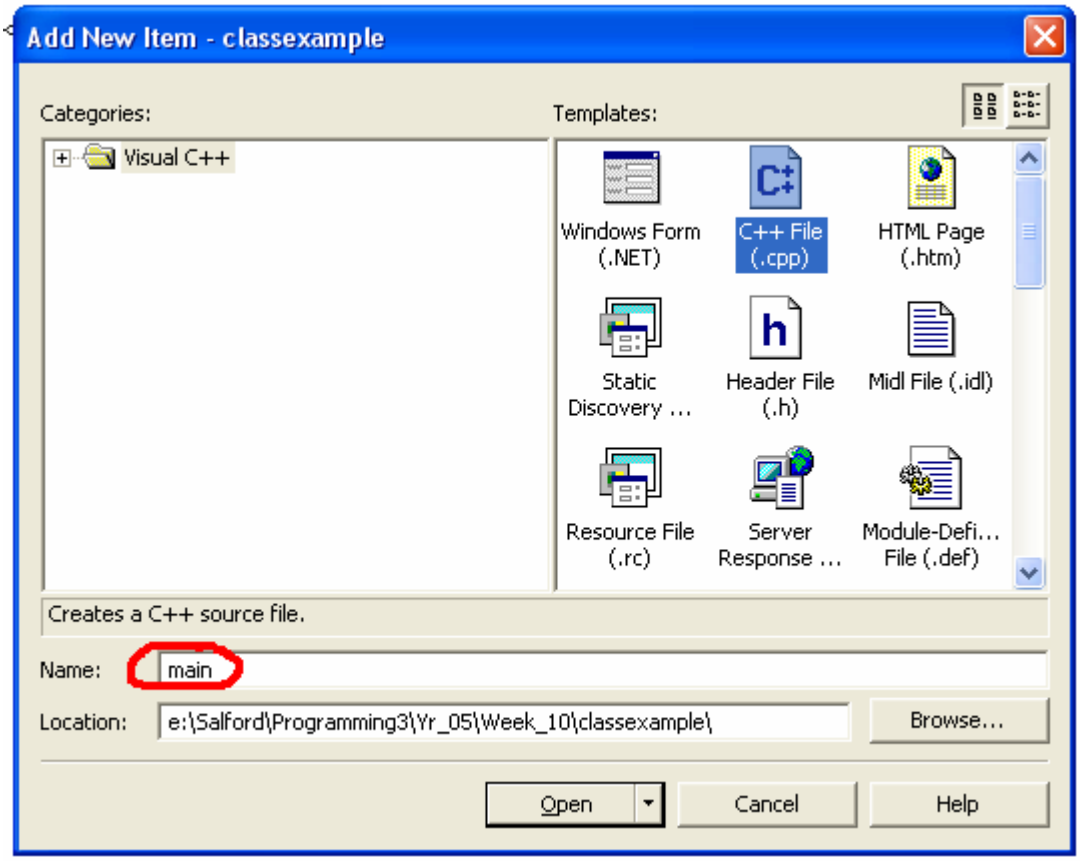
AlterArmour will be very similar to the AlterHealth method above.
The remaining functions are all `const` and therefore do not alter any of the member data.

The Get functions will simply return the values they are supposed to retrieve.

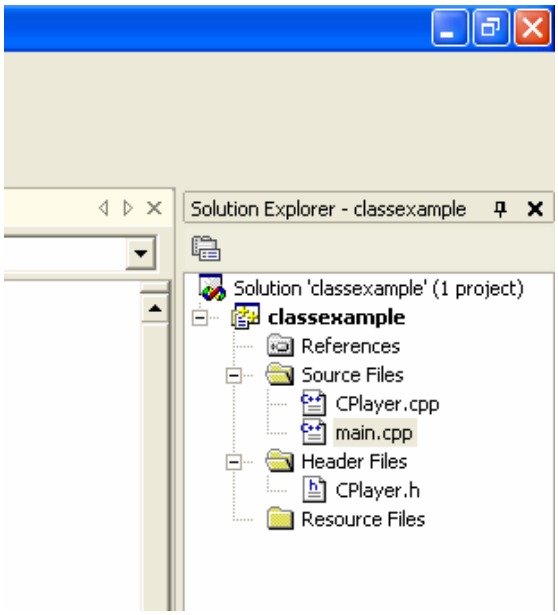
The final function, DisplayStats, simply prints the players health and armour values to the screen.

Adding our main code file
Finally we add our main code file as we normally would.





Our project should now have three files in the solution explorer. It should look like this:



Coding our main code file

Now we get round to polishing off the project by coding our main function. This main code is going to be nothing more than a driver program, one that simply tests our class.

```
#include "CPlayer.h"

int main()
{
	CPlayer	playerobject;

	playerobject.AlterHealth(-23);
	playerobject.AlterArmour(50);

	playerobject.DisplayStats();

	return 0;
}
```

All we are doing here is instantiating a member of our class. Our class, CPlayer, is effectively a new data type. Instantiation is therefore pretty much the same as making a variable of a simple data type.

We then call some of the member functions, altering the data stored in our object (object being an instance of a class). Before finally printing out the stored data via the member function DisplayStats.

Adding our Functionality

The final part of this tutorial sees you adding your own class method to CPlayer.
You should code a method called:
bool TakeDamage(int D)
This method should reduce the objects health by the specified amount and return true if the objects health drops below 0.

Call this method in main, testing it's return value in an if-statement and print out a relevant response.