**Software Development 2**

**Part 1: Object Orientation**

Last week our practical focused on using C# and .NET . In the lecture we covered the basics of Object Oriented software. This practical will take you through some background reading and strengthen your knowledge of OO theory.

We will be using the free online book “Object Oriented Programming using C#” available from BookBoon.Com. Please register and download a copy from :<http://bookboon.com/uk/textbooks/it-programming/object-oriented-programming-using-c-sharp>

*The tasks and exercises below will take more than the 2hrs that you have available to you in this practical, don’t worry about this, just finish them off in your own time. If you have any problems or questions email your tutor.*

We will refer to this book during the module.

1. Download the ClassExamples.ZIP file and unzip it. It should contain 10 C# projects. From within Visual Studio make sure that you select File->Open -> Project / Solution and then select “Class Examples\Classes1\Classes.sln”. All 10 example projects should load.
2. **EXERCISE 1**

Examine each project in turn, carry out the tasks contained in the comments within the code.

1. Starting on page 16 of the E-Book read section 1.4 and undertake activities 4-7, write down your answers and discuss them with your tutor.
2. Now read section 1.5 and work through activities 8-11
3. **EXERCISE 2**
   1. Within an auction a number of items may be sold, each item has a lot number (unique id), a description, its reserved price and the current price. When bidding is opened on an item, bidders may submit a bid. At some point the auctioneer will close the bidding on the item and the item will sold to the highest bidder.
   2. Identify the objects in the above scenario.
   3. For each object identified in part b, suggest attributes and operations (methods).
   4. Discuss your answers to b & c with your tutor
4. Read section 2.2- This section introduces a concept called Unified Modelling Language (UML). UML is collection of techniques for the analysis and design of software systems. At this stage we’re not going to cover the UML process, but we will be covering Class Diagrams which are used to show the classes contained within an OO system. Work through activity 1.
5. **EXERCISE 3**

Use the UML notation to draw the classes identified in exercise 1, discuss this with your tutor.

1. Now read section 3.3 undertake the activities, discuss the answers with your tutor.
2. **EXERCISE 4**

Look at the following scenario, identify the classes within it and draw the appropriate class diagram. Make use of inheritance.

*ENU Properties have a number of properties that they rent out. All properties have an address (number, street, postcode) all properties have rent collected each month. Residential properties also have a landlord registration number associated with them, as well as a number of bedrooms. Residential properties with more than 2 bedrooms as classed as multiple occupancy and have a fire inspection id and renewal date associated with them. Commercial properties are classed as retail, warehouse or office and have size (square meters associated with them).*

**Part 2: Methods**

1. Work through the example projects in the Methods solution supplied in the .ZIP with this document. Open the solution file “\methods\ConsoleApplication1\ConsoleApplication\_Methods.sln”, which contains 5 projects (Method 1 to Method 5).
2. Make sure that you know how to use the Debugging tools in Visual Studio to “Step Into” the code and see the order that each line of code is run in. Press F11 to Step Into the Code line by line. Right Click on Variables to “Add Watch” and see how each variable's values change as the program runs (and their scope). Watch the video Video3-1.wmv for a demo.
3. In the Solution Explorer Window right click on the Method 1 project and select “Set as startup project”. In the file M1.cs add a few more SayHello(); lines to demonstrate stepping into

Note: The main method always executes first

Then it calls the SayHello() method

The SayHello() method just produces a console with the message “hello”

Control then returns to the main method

1. Select Method 2 as the startup project, in M2.cs add a few more myName = "Jack"; and SayHello(); lines to demonstrate multiple calls to SayHello();

Within the SayHello() method, change the value of what is stored in Name to show that this does not affect the value of myName and step into the code again

e.g. Use Name = Name.ToUpper();

1. Select Method 3 as the startup project andchange the values of x and y in the Main() method to show that the program works.

Amend the program so that the user enters the values of x and y in the console. You can read a string from the console like this:

string myString = Console.ReadLine();

You will need to convert from a string to an int. HINT: you converted from string to double in the first practical.

Change to a windows application and use to compare the size of two labels

the size of the labels should be controlled by the user via slider components.

(use the Visual Studio Help to find out how to use slider controls and change the size of labels – or Google if you prefer)

* Remember you do not declare methods as being static when including them in a Windows Form Class

1. We can pass arrays as arguments, select Method 4 as the startup project and use the debugger to place a Watch on both myArray and anArray

Note Arrays are different...

When an array is passed to a method only a **reference** to that array is passed

The array is NOT copied.

This means that any changes to the array in the method WILL change the original array!

Create and test a method called multiplyByTwo which doubles each value in the array.

1. We can return an array from a method, to see how, select Method 5 as the startup project. The getData() method must be declared to return an array, the array to be returned must be created inside the method. Then the array is returned from the method.

Use the Visual Studio Debug to place a Watch on both myData and anArray.

Change this GetData() method to get data (5 numbers) from the console.

Write a method to display the contents of myData instead of the lines following '// output to console' in Main.

1. Create a new console application. Create a class called application with a main() method. Create a class Person in a separate file. Person should have the following properties:

name : string

age : integer

address: string

In c# you can create properties using the following syntax:

private string m\_name = string.Empty;

public string Name

{

get

{

return m\_name;

}

set

{

m\_name = value;

}

}

Add name, age and address properties to Person using the above syntax. Add the following methods to Person:

whoami: takes no parameters, returns no value, but prints name, age and address to the screen

say: takes a String parameter and prints it to the screen

birthday: takes no parameters, but adds 1 to age and returns the new value

Modify your main() method to create several Person objects and call their various methods.

1. Add a constructor to Person that takes, name and age as its parameters.
2. Refer back to the e-book Object Oriented Programming Using C# (<http://bookboon.com/uk/textbooks/it-programming/object-oriented-programming-using-c-sharp>) .
   1. Start working through chapter 6, read sections 6.1 -> 6.8, work through the activities contained therein.
   2. Consider the following scenario:

*The Besco Supermarket offers a home delivery service. Each participating customer places an order to be delivered to their address. Each order consists of a number of items, each item being identified by a stock code. Associated with each order is a receipt containing item descriptions and prices. Orders are allocated to delivery schedules, each day a group of schedules will be allocated to a driver, who is in turn allocated to a van. Each schedule has an associated time and millage, to allow the daily millage for each van to be calculated and the hours worked for each driver to be calculated.*

In order to commence constructing a system to manage the home delivery service please Identify classes, properties and methods in the above scenario using the techniques suggested in chapter 6.

* 1. Implement two the classes identified in part b using (e.g. Customer and Receipt). Separately from each class implement a c# Windows form that allows an instance of the class to be created and then allows the methods in the class (including properties) to be tested.