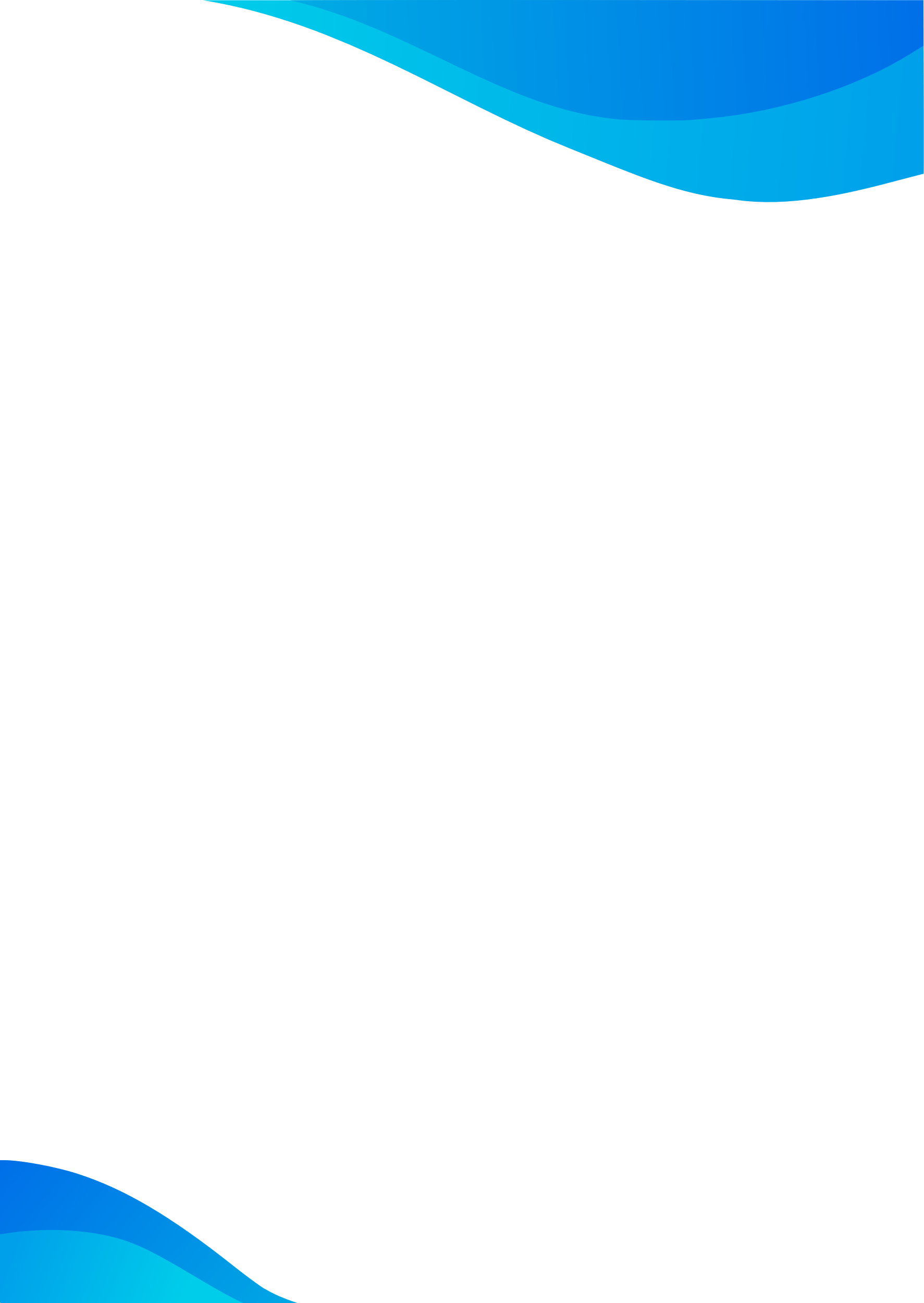
Creating an Object Oriented Program Workbook

**Name: Geraint Waddon**

**Student Number:**

**Module Code:** CL44CY001

**Module Title:** Level 4 Designing and Developing Object Oriented Computer Programs

I hereby declare that this workbook contains only my own work.

| Signature: | Geraint Waddon |
| --- | --- |
| Date: |  |

# Learning Outcomes

| 1. Design object-oriented programs to address loosely-defined problems. | 1. Identify a set of classes and their interrelationships to address the problem. |
| --- | --- |
| 1. Make effective use of encapsulation, inheritance and polymorphism. |
| 1. Select and reuse pre-existing objects and templates specialising as required. |
| 1. Structure the design so that objects communicate efficiently. |
| 1. Specify the properties and behaviour of classes to allow efficient implementation, selecting appropriate data types, data and file structures and algorithms. |
| 1. Record the design using well-established notations. |
| 1. Produce a working object-oriented program which meets the design specification. | 1. Make effective use of basic programming language features and programming concepts to implement a program that satisfies the design specification. |
| 1. Make effective use of the features of the programming environment. |
| 1. Make effective use of user interface components in the implementation of the program. |
| 1. Make effective use of a range of debugging tools. |

| 1. Develop object-oriented programs that reflect established programming and software engineering practice. | 1. Apply standard naming, layout and comment conventions. |
| --- | --- |
| 1. Apply appropriate data validation and error handling techniques |
| 1. Develop test strategies and apply these to object-oriented programs. | 1. Develop and apply a test strategy consistent with the design identifying appropriate test data. |
| 1. Apply regression testing consistent with the test strategy. |
| 1. Use appropriate tools to estimate the performance of the program. |
| 1. Develop design documentation for use in program maintenance and end-user documentation. | 1. Record the final state of the program in a form suitable for subsequent maintenance. |
| 1. Provide end-user documentation that meets the user’s needs. |

# Part 1 - Planning your

# Program

You have been tasked with creating a text adventure game in C# using object oriented techniques. The game can have whatever theme and narrative you like, the only requirements for the game are:

* There is at least one win condition
* When the game is finished, the player can opt to play again without reloading the program
* The game has a launch window with multiple UI elements
* The code is backed up by a robust testing strategy that includes regression testing
* The code is written in a consistent style and well documented

| 1. **Briefly describe your game’s premise and structure.** | **1.1** |
| --- | --- |
| The game will be a murder mystery text adventure game with strong Welsh themes. The player will begin the game in a room with three other characters and a body. The player will be given various choices throughout the game which will lead to either the next stage or the game will end.   The player will need to collect clues about who the murderer is by interacting with the other characters and moving around the room before the police arrive. The game will contain items that might have been used for the murder or they may be a red herring. The game will ask the player if they would like to make a guess at various points within the game however if the player guesses incorrectly the game will end.  The aim of the game is to use the clues found in the game to determine who the murderer is. The player will win the game if they guess correctly.. | |

| 1. **Describe the classes your game needs to address its technical requirements. Describe the properties, attributes and behaviour of each class.**   *Justify your choices* | **1.1, 1.5** |
| --- | --- |
| A person class - this will contain properties such as Name, Age, Height, Build, Health. These will be needed for the clues on what happened. This class will contain methods that all people in the game can do e.g a method that all people will be able to greet the player when talking to them. (public abstract string Greet();)  A character class - this will inherit from the person class but will contain properties of that specific character e.g Bryn will have a specific greeting.   Item class - this will contain properties such as Size, Weight, Description,ContainsBlood. This will help the player determine if the item they’ve found could be used as a weapon or a clue for example. This will also contain an abstract method called Clue that the items that inherit from this class will be able to override.  Various items class - these will inherit from the items class but will contain an override method for Clue such as a boolean that contains blood splatters or not for example. | |

| 1. **Draw a UML diagram to represent the classes and behaviours described in question 2.**   *Explain how your design allows objects to communicate efficiently* | **1.4, 1.6** |
| --- | --- |
| Person class will contain all people in the game including the player. All characters and the player will inherit from the Person class and the characters will use polymorphism to access the Greet method where they will be able to override the Greet method to return a custom value for each character. | |

| 1. **Explain how your coding style is standardised.**   *Include naming conventions, layout and comments* | **3.1** |
| --- | --- |
| TBD | |

| 1. **Using your answers to questions 1 and 2, create a comprehensive test strategy.**   *Your strategy should detail how you test your classes and methods and what tools you will use.* | **4.1, 4.2** |
| --- | --- |
| TBD | |

# Part 2 - Reviewing your program

| 1. **Describe the three main control structures in programming.**   *Provide screenshot examples of each from your solution* | **2.1** |
| --- | --- |
| The first main control structure is the sequence on how the code is run. For instance, c# is read sequentially (top to bottom) so it will run the first line before moving on top the next line. E.g:  The second main control structure in programming is the selection structure. For instance, if you wanted to run a block of code if a certain criteria is met then you would use an if statement. E.g:    The third main control structure in programming is the repetition structure. This is used when you want to execute a block of code x amount of times or until the break clause condition is met. E.g: | |

| 1. **Provide examples of pre-existing objects and methods you have included in your solution.** | **1.3** |
| --- | --- |
|  | |

| 1. **Explain how you incorporated encapsulation, inheritance and polymorphism into your solution.**   *Provide screenshot examples* | **1.2** |
| --- | --- |
| I provided encapsulation in my solution by E.g:    I provided inheritance in my solution by creating parent classes (People and Items) then created child classes (Bryn, fishing rod…), that inherit all the attributes and methods from their parent classes. E.g:    I provided polymorphism in my solution by creating a custom abstract method called Greet() in the People class. This is then used by the characters classes to greet the player in a unique output when that method is called via character.Greet() E.g: | |

| 1. **Describe the user interface components you implemented in your solution.**   *Provide a screenshot example* | **2.3** |
| --- | --- |
| Maui? | |

| 1. **Explain how you incorporated error handling into your solution.**   *Provide screenshot examples.* | **3.2** |
| --- | --- |
| I used a try catch statement within a for loop to handle exceptions in case the player used illegal characters when asked for an input. If the user did enter an illegal character it would loop back to the main menu for the player to try again E.g: | |

| 1. **Illustrate how you made use of debugging tools when developing your solution.** | **2.4** |
| --- | --- |
| I used the debugging tool in Visual Studio throughout the build of the solution for multiple reasons. The main reason I used the debugging tool was to find out what would stop my solution from building so I could make amendments. I also used the debugging tool to test-run my solution and check the output for any unexpected bugs. | |

| 1. **Explain how you implemented your test strategy from part 1.**   *Detail how you enacted regression testing.* | **4.1, 4.2,** |
| --- | --- |
| TBD | |

| 1. **Provide examples of test data you used when enacting your test strategy.**   *Justify your choices* | **4.1** |
| --- | --- |
| TBD | |

| 1. **Explain how you measured the performance of your program and what your findings were.** | **4.3, 2.2** |
| --- | --- |
| TBD | |

| 1. **Write developer documentation for your solution.**   *Include a sentence explaining how your documentation allows your solution to be maintainable.* | **5.1** |
| --- | --- |
| TBD | |

| 1. **Write end-user documentation for your solution.** | **5.2** |
| --- | --- |
| TBD | |