Learning Summary Report

Object Oriented Programming

COS20007

2021

Written by Marella Morad

Student ID: 103076428

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment				✓

Self-Assessment Statement

	Included
Learning Summary Report	✓
Test is Complete in Doubtfire	✓
C# programs that demonstrate coverage of core concepts	✓
Explanation of OO principles	✓
All Pass Tasks are Complete on Doubtfire	✓

Minimum Pass Checklist

	Included
All Credit Tasks are Complete on Doubtfire	✓

Minimum Credit Checklist (in addition to Pass Checklist)

	Included
Distinction tasks (other than Custom Program) are Complete	✓
Custom program meets Distinction criteria & Interview booked	✓
Design report has UML diagrams and screenshots of program	✓

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
HD Project included	
Custom project meets HD requirements	✓

Minimum High Distinction Checklist (in addition to Distinction Checklist)

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: **Marella Morad**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieved all Unit Learning Outcomes for COS20007 Object Oriented Programming to an **HD** level.

Summary of the Tasks Included:

This semester, I completed the following tasks:

- Pass Task 1.1 - Object Oriented Hello World
- Pass Task 1.2: C# Reference Sheet
- Pass Task 2.1: Counter Object
- Pass Task 2.2: Drawing Program – Shapes
- Pass Task 2.3: Case Study Iteration 1
- Pass Task 3.1: Drawing Program – Drawing
- Pass Task 3.2: Clock
- Pass Task 4.1: Drawing Program - Different Shapes
- Pass Task 4.2: Case Study Iteration 2
- Pass Task 5.1: Case Study Iterations 3 and 4
- Credit Task 5.2: Drawing Program – Saving
- Distinction Task 5.3: Drawing Program - Flexible Saving
- Credit Task 6.1: Case Study - Iterations 5 and 6
- Credit Task 6.2: Custom Program Plan and Design
- High Distinction Task 6.4: HD Custom Program – **Super Mario 29.11**
- Pass Task 7.1: Object-Oriented Principles
- Credit Task 7.2: Object-Oriented Programming Concept Map
- Distinction Task 7.3: Case Study - Iterations 7 and 8
- Semester Test
- Distinction Task 9.1: Case Study - Take, GUI and DLL
- Pass Task 10.1: Clock in Another Language
- Distinction Task 10.2: Other Language Report

Evidence of Covering the Learning Outcomes:

My purpose in this section is to showcase how my pieces illustrate my understanding of each of the unit's learning outcomes.

Learning Outcome 1

Explain the principles of the object-oriented programming paradigm specifically including abstraction, encapsulation, inheritance, and polymorphism.

I have demonstrated a deep understanding of the object-oriented programming paradigm which is evident in tasks 7.1P, 7.2C, 8.1P, 5.3D, 9.1D and 6.4HD. In task 7.1P, I included a description of each of the principles along with an example showcasing how to apply them. I created a concept map in 7.2C that explains the relationships, correlations, and descriptions of Object-Oriented Programming in general and the four OOP concepts in particular. Furthermore, in task 8.1P, the semester test, I covered abstraction and polymorphism in relation to the Library solution I created for the task. Moreover, in task 5.3D, I applied inheritance in the child classes that derive from the abstract Shape class (MyRectangle, MyCircle, and MyLine). Other than explaining each of the principles, I practised applying them in the programs I created, beginning with the 1.1P task, which demonstrates encapsulation and progressing to SwinAdventure and Super Mario 29.11 (9.1D and 6.4HD), in which I applied the four principles with increasing complexity.

Learning Outcome 2

Use an object-oriented programming language, and associated class libraries, to develop object-oriented programs.

For this unit, we used C# as the primary programming language to investigate OO programming. Coming from a C# background, it was fascinating to see the other approach, the OOP approach, to developing applications. With the language syntax being the same, it was easier for me to implement the programs in each of the given portfolio tasks, given that I did not need to learn the language from scratch. I started with a simple program, task 1.1P, where I used C#'s System library to print messages on the application Console. As we progressed through the unit, we started using other libraries such as System.Collections.Generic to be able to use Lists in C#. Furthermore, we used an external library called SplashKit to draw on the screen. I used this library in all the Drawing tasks (2.2P, 3.1P, 4.1P, 5.2C and 5.3D) as well as my custom program, 6.4HD, where I learnt and applied new commands from the library. Furthermore, I developed a C# reference sheet that contains the majority of C# language syntax used in this portfolio (Task 1.2P). I also explored OOP in C# further when I researched the similarities and differences between C# and Python as OOP languages in task 10.2D as well as the concept map in 7.2C where I explored the use of classes to represent objects in OOP and how to encapsulate methods and fields inside a class which is accomplished using variable and method scope in C# (public, private, static, protected, etc....). In addition to exploring Python as an OOP language, I implemented OOP in Java for task 10.1P.

Learning Outcome 3

Design, develop, test, and debug programs using object-oriented principles in conjuncture with an integrated development environment.

I have designed, developed, tested, and debugged programs using OO principles in combination with Visual Studio as my IDE. I explored some different types of solutions that can be created using Visual Studio, such as Console and Windows Form applications. I mainly used a console application to develop SwinAdventure classes and associations, which I then built as a dll and included as a part of my 9.1D task, which is a WinForm application demonstrating the game, SwinAdventrue. SwinAdventure is also where I applied testing skills using NUnit testing (using C#'s NUnit.Framework) to ensure each of the classes is performing as expected. I have also done a testing phase for my custom program, Super Mario 29.11, where I tested all the functionalities and fixed the ones that had bugs. During this phase, I used the debug option to be able to spot the bug and follow the stack trace to resolve it. Lastly, I used an online compiler to write the Clock code in Java for task 10.1P.

Learning Outcome 4

Construct appropriate diagrams and textual descriptions to communicate the static structure and dynamic behaviour of an object-oriented solution.

Throughout the unit, I was able to develop and demonstrate advanced design skills in the design and documentation of my programs. This is mainly evident in SwinAdventrue iterations, which is where I applied design principles to design the required classes starting from iteration 5 and onwards (including tasks 6.1C, 7.3D and 9.1D). I also applied design skills in re-designing the library program for task 8.1P, to include an abstract class that other classes can inherit from (applying polymorphism). Moreover, I demonstrated advanced design skills in my custom program, where I included new design patterns such as singleton and strategy to make my classes more coherent and specific. To do all of that, I relied on UML class diagrams, which represent the encapsulation of classes and their relationships with other classes. Moreover, I used UML Sequence diagrams to illustrate certain functionalities of classes and how they interact with each other to achieve the requested functionality. These diagrams can be seen as part of tasks 3.2P, 6.1C, 7.1P, 7.3D, 8.1P and 6.4HD). In terms of documentation, I demonstrated XML documentation skills in task 5.3D, where I stored objects in a text file, which can then be read and re-drawn on the screen using SplashKit.

Learning Outcome 5

Describe and explain the factors that contribute to a good object-oriented solution, reflecting on your own experiences and drawing upon accepted good practices.

After undergoing this unit, I have learnt that a good object-oriented solution applies all four principles correctly. Object-oriented programs are based on creating specialised classes and defining the associations between them to develop the final product. Other than having one class responsible for all the functionalities in the program, roles and responsibilities are broken down into smaller roles that single classes are responsible for. I have applied this observation in my custom program, as I designed and implemented a class called Resize, and another class called Direction which are responsible for enlarging, shrinking and identifying the direction the player is facing (see Super Mario 29.11 for further details). Initially, I planned to include all these roles into the Player class, however, I then decided to refactor these roles into smaller classes that can work in association with the player class. Another important feature of OOP that I realized is that using the concept of inheritance, for example, makes the program highly modular, allowing other classes with similar roles to be added and inherit from the parent class without the need to rewrite large areas of the program.

Reflection

The most important things I learnt:

The most important thing that I learned from this unit is how applying the four object-oriented principles can significantly decrease, if not remove, the need to duplicate code. In the past, I applied procedural programming using C# in which was challenging to avoid code duplication. However, in object-oriented programming, we can apply the inheritance concept to allow multiple classes to have the same property or even override some properties, if necessary, without the need to duplicate code. Another main thing I learned from this unit is the interpretation and design of UML class and sequence diagrams, and how extremely important it is to thoroughly think about a program before beginning to write its code. Even if the plan is incomplete, it is better to have a starting point, which is what I had when I started developing my custom program. Even though my UML class and sequence diagrams look a lot different from my initial plan, having that plan is what allowed me to get to this stage. Lastly, I really found the test-driven programming approach helpful as I could define multiple test cases that revolve around the main functionality of a class and write the class from there using the assistance from the test, NUnit tests in this unit.

The things that helped me most were:

- Lectures (live and recordings): these are where I gained the most knowledge about OOP from. Lectures and lecture notes provided me with a great reference to OO principles which I carried out and applied throughout the semester.
- Lab demonstrations: throughout the semester, my tutor demonstrated key concepts from the unit, especially NUnit testing and UML diagrams and associations which I found very helpful given that I had no prior experience with object-oriented programming.
- Live Consultations: these define the best experience I had in a unit at university as I have access to tutors almost all the time throughout the week for consultation regarding my progress through the unit, my programs, any errors that I could not figure out. The feedback I received from tutors/unit convenor really helped me develop a better understanding of the OOP application as well as following food practices in designing and developing my programs.
- Google, Stack Overflow, and other online resources: I was able to use these resources to find examples that apply similar ideas to what I needed to develop. One main thing that I research and found really helpful online resources about was applying physics to make my character jump (projectile motions, using speed and gravity), which I was then able to apply in my custom program with the help of my tutor.

I found the following topics particularly challenging:

- At the start of this semester, I found NUnit testing, and UML diagrams interpretation very challenging. Initially, I had to keep looking at existing diagrams, and test cases to be able to write my own, however, as time went by, and I got to practice writing my own tests and drawing my own diagrams more, I have become more confident in these two areas.
- Overall, I found shifting from a procedural programming approach to an object-oriented programming approach really challenging as I usually tend to write large classes with small methods/procedures rather than having separate classes performing more specified functionalities.

I found the following topics particularly interesting:

- I found game development, which I applied in my custom program, very interesting. The fact that I had to think about every single movement the player does was frustrating at times, however, very interesting and very rewarding when I was able to make the functionality work.
- I also found NUnit testing interesting along with the fact that we can compare our expected results to the actual results.

I feel I learnt these topics, concepts, and/or tools really well:

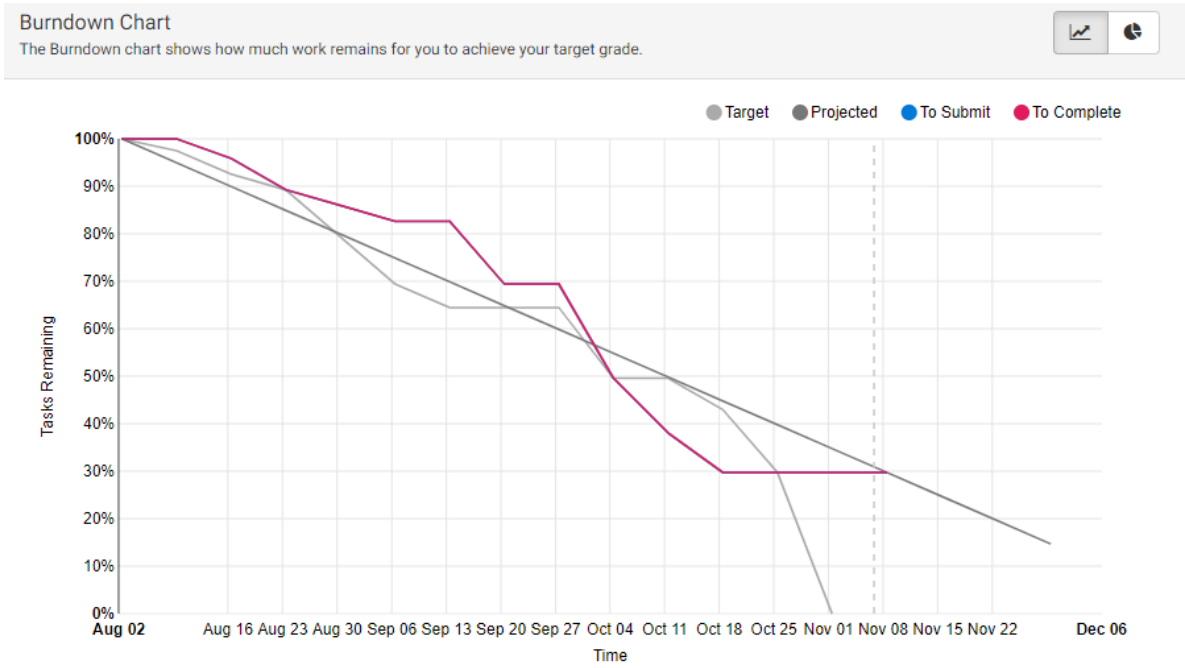
- The four object-oriented principles including encapsulation, abstraction, inheritance, and polymorphism, which I have applied in almost all my tasks as mentioned earlier in Learning Outcome 1.
- Draw.io tool which I extensively used to design my concept map, UML class and sequence diagrams for the tasks mentioned in Learning Outcome 4.
- I also learnt about good OO programming practices in C#, especially how a foreach loop is favoured over a for or a while loop and I experienced the difference in my custom program. Initially, I was trying to implement jumping functionality using a for loop, which did not work as when a for loop is run, other parts of the code cannot run with it simultaneously, resulting in the game stopping for a while to do the jump, then running again for other actions.

I still need to work on the following areas:

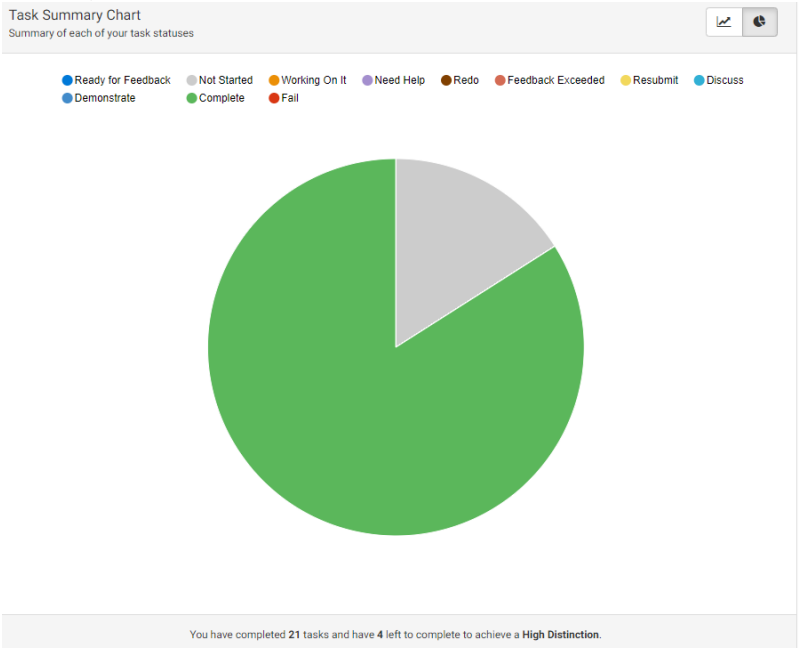
- Writing programs using a fully object-oriented approach: as I mentioned earlier, I have the habit to follow procedural programming which causes my classes to get larger, however, I am working on making them more specific and creating new classes/interfaces that other classes can inherit and implement features from (for example having multiple block classes that inherit from an abstract Block class with each having its specific bitmap and type)
- Drawing accurate UML class and sequence diagrams: I think now I am at a good stage with drawing my diagram, nevertheless, I believe more practice is required to understand the relationships more deeply and be able to apply them.
- Design patterns: I think putting the extra time into learning and applying new design patterns is essential to be able to produce a truly object-oriented program. I have already started learning about new patterns such as singleton and strategy which I applied in my custom program, but I think in the future, some other patterns can also be helpful in this context.

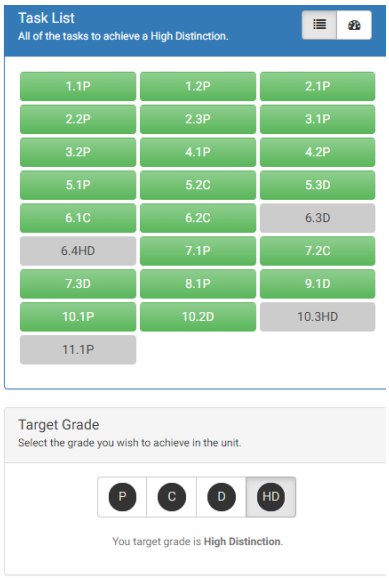
My progress in this unit was:

My progress in this unit was consistent, as I was up to date with the tasks, having almost all the tasks done by the target date. After completing SwinAdventure, I shifted my focus to my custom program and started the design stage of it, whilst working on the conceptual tasks such as 7.1P, 7.2C, 8.1P and 10.2D. Overall, I believe I gave myself a good amount of time to complete each of the tasks, however, I could have used some more time on my custom program.



As shown in the pie chart below, I have 84% of my tasks marked as complete with only the custom program and the learning summary report draft missing. *I will not be submitting an HD project (task 10.3HD).*





This unit will help me in the future:

Having learnt and applied the object-oriented programming approach, I believe will really help me in future jobs as well as future units applying similar principles. This unit also showcased that programming does not only differ from a language to another but also there are different styles of programming which I am really keen to explore. One major thing that I think will be really helpful in the future is testing, and the approach of test-driven programming, as the testing phase is one of the most important phases in programming to ensure meeting required functionalities. As a future Software Engineer, having this skill will be very helpful to me.

If I did this unit again, I would do the following things differently:

- One thing that I will utilise more if I do the unit again is the help desk. This semester I approach the unit relying mostly on myself and the internet, which was time-consuming, and could have been done a lot quicker if I consulted with a tutor or even a classmate.
- I would also practice taking my programs to the next level of efficiency, applying the cores of object-oriented programming.

Conclusion

Overall, I feel I have demonstrated that my portfolio is adequate to receive an HD.