COS20007

Object Orientated Programming

Learning Summary Report

Matthew Coulter

102573957 ID

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: Matthew Coulter

[illegible]

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS20007 Object Orientated Programming to a **High Distinction** level.
The following learning outcomes were required of this unit:

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

I have demonstrated by understanding of the key object orientated programming principles within the tasks of:

7.1P Object Oriented Principles

This task shows a thorough understanding as it present both a written and verbal version of a clear understanding about each concept listed.

7.2C Object Oriented Programming Concept Map

This task highlights the relationships between all of the concepts through a large relationship diagram.

8.1P Semester Test

The test requires us to both write the code and explain how each concept contributes to flexible and extensive software.

10.3HD High Distinction Project

My High Distinction video places me as a teacher where I am educating another person about each of the programming principles as well as Object Pooling.

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

Throughout this whole semester, we have used the associative class library SwinGameSDK to aid our development within C#. This is demonstrated through the following tasks:

Drawing Program (2.1P, 3.1P, 4.1P, 5.2C, 5.3D)

The drawing program was an interface that allows the user to draw the shapes of Rectangle, circle and line. Then, the user can select these shapes, delete them and save the current compilation of layers to a txt file, to load them back later on. This was a good example of object orientated programming with the implementation of SwinGameSDK.

Swin-Adventure (2.2P, 4.2P, 5.1P, 6.1C, 7.3D)

Swin-Adventure worked through a series of iterations to develop a console program that would accept some commands and guide the user on an adventure where they can pickup tools, drop tools, examine objects and places and work their way to some destinations.

9.2D Case Study Take, GUI and DLL

Task 9.2D required us to make our own library, which we implemented into a windows form for swin adventure. This essentially creates a reference to all of the classes that were written which means we only need to gamemain.cs to run the program.

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

We were introduced to unit testing this unit. Throughout all of the development this semester, I have implemented unit tests to validate the code that I am writing. This was demonstrated through the following tasks:

Swin-Adventure (2.2P, 4.2P, 5.1P, 6.1C, 7.3D)

The submissions for these iteration required that we implemented unit tests for basically every single method in the classes. This is why it is a good example for the unit testing learning outcome; it demonstrates how unit tests are very concise and test a single piece of code. This also allowed for many different asserts to be used.

Custom Program (6.3D)

In my 6.3D custom project, I have also written unit tests for all of the different components within my game. Whilst this is a requirement that tests are written, I found that it was necessary to get my program to work. For example, I was not able to run breakpoints to test intersections because the vehicles are always moving. Hence, I could position two rectangle overlapping and call one of their test intersection methods. This verified that my code was actually working properly.

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

We have actively read and constructed both UML class Diagrams and Flow diagrams to present the structure of our programming. This has been predominantly done through the following tasks:

Swin-Adventure (2.2P, 4.2P, 5.1P, 6.1C, 7.3D)

For the first few iterations, it required that we analyse and interpret the diagrams that were provided to us to form the spine of the class constructions. Throughout the later iterations, we actually had to create our own diagrams for the submission.

Custom Program (6.3D)

I have also made these diagrams in my custom program. This is an extensive ~20 class diagram for my custom projects which contains every class that my software utilized. Furthermore, I have made some flow diagrams to demonstrate they way key functions of the software operate.

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

I have learned about how to make efficient code and demonstrated my understanding through the following tasks:

8.1P Semester Test

In the last section of the test, we described how the good practices we implemented in the code leads to extensible and flexible software. Furthermore, my 10.3HD High Distinction Project explains additional good practices such as object pooling to further accomplish and ‘good object-oriented solution’. In this explanation, this was linked back to the way the code for the bee console app was written

10.3HD High Distinction Project

In my video, I thoroughly explain how object-oriented principles lead to efficient solutions. Furthermore, this introduces how object pooling leads to effective and efficient solutions. As two examples are used as evidence of their effectiveness, that satisfies this learning outcome.

Whilst I have demonstrated the requirements and completion of each learning outcome to be worthy of a High Distinction mark, I believe I am a strong candidate for this mark because I have invested a lot of time into learning about the concept of object pooling to apply in both my Distinction custom program and High Distinction Project. Furthermore, I believe the quality of work I have submitted for these is of a very high level. My Custom Program is a creative and thoughtful game that satisfies the requirements of OOP requirements as well as the object pooling. I have produced a very detailed and thorough video whereby I educate my brother “Shane” about Object Pooling, as evidence of my understandings. All of my submissions are presented neatly and consistently as a common format to highlight the effort I have put into this unit.

Reflection

The most important things I learnt:

Coming into this unit, I was not sure what object-orientated programming even meant. But now I have a very clear definition of what it is and why it is so important. In general, that understanding is one of the most important things I have learnt. In addition, the specific methods of testing, creating UML diagrams and implementation of Object Pooling are what IU have also found to be really important.

The things that helped me most were:

Attending the help desk during Arafat's time was most helpful. He was not my tutor this year, however I found that he was able to explain things to me in a way that other tutors could not. I also did a lot of research in my own time, referencing online sources, particularly stack overflow to help solve problems that other users also had problems with.

I found the following topics particularly challenging:

Trying to conclude the best and MOST efficient way is sometimes hard. This is because there are always multiple ways of implementing something so a lot of thought needs to be considered.

I found the following topics particularly interesting:

I am really interested in polymorphism, particularly because it is the heart of reusing code. As pa person, I am obsessed with having things work as efficiently as possible and this concept in programming was quite enthralling. Extending on this idea of efficiency, the object pooling design pattern was very interesting as it was not something I had originally thought about. It is quite cool that there are common ways of implementing things universally between different types of games.

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

I feel like a really grasped a full understanding of polymorphism and how to construct classes leading to really efficient code. Subsequently, my understanding of inheritance, abstraction and encapsulation are very good. Throughout the various ways I implemented object pooling for both the traffic controller and the gun shooting program (10.3D) that has given me a very clear understanding of what object pooling is and the benefits of implementing it.

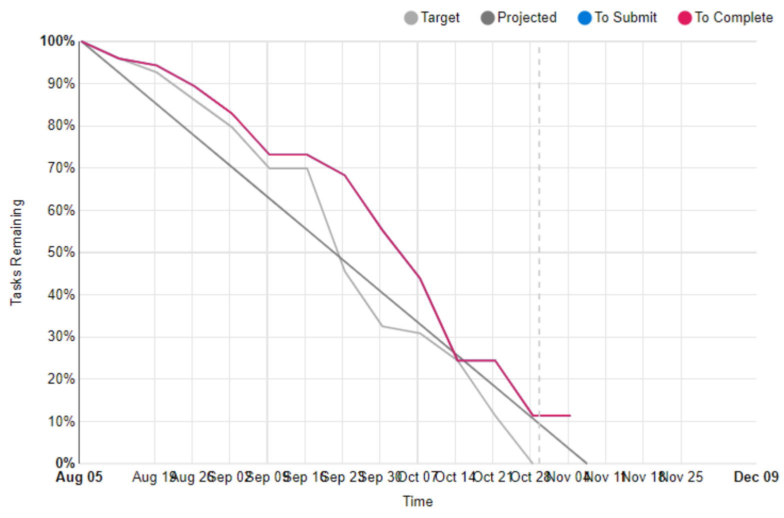
I have learned how to properly test code through the use of unit testing in both Visual Studio IDE and also Chai and Mocha for JavaScript.

I have learned how to present the structure of software in a concise way through the use of UML diagrams and UML flow diagrams.

I still need to work on the following areas:

A fuller understanding of how interfaces work.

My progress in this unit was:



Throughout this unit I had a relatively steady rate of progress. Whilst most of the time the ‘To Complete’ was above the ‘Projected Line’, this is not so significant because each task does require a different amount of time to complete. Whilst there were a few hiccups around the beginning of semester, I was able to catch up and actually submit my distinction custom project in the week that it was due. This was good as later on it allowed me to make improvements to reach the grade of HD. Had I not got ahead of time; I would not have been able to submit iteration 2 of my Custom Program.

This unit will help me in the future:

This unit covers the fundamentals of how games are developed. As I am undertaking a games development major, this unit helped me through a very thorough understanding of how to structure a game by incorporating the key concepts of Polymorphism, Abstraction, Encapsulation and Inheritance. Furthermore, the additional concepts that I was encouraged to research such as the object pool design pattern will go along way in terms of developing efficient games.

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

I would begin to invest more time into revising the niche topics such as interfaces. I would also be looking over more object design patterns beyond object pooling and try and develop using contemporary engines such as unity.

Other Comments:

I really believe my programming has improved a lot from this unit. Coming into this unit, I scraped by with a 57 in Introduction to Programming which I was really disappointed with. That is why coming into this unit I have tried really hard to change that.