COS20007

Object Oriented Programming

Learning Summary Report

Klim Huynh
101634015

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		67		

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: **Klim Huynh**

Portfolio Overview

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

I am aiming for a final grade of 67 because I have earned this grade from satisfying the following:

- Passed the semester test
- Completed all pass tasks
- Completed all credit tasks
- Completed all distinction tasks (excluding custom program)
- On time submissions for most of the semester

By completing the above, I have been able to demonstrate a good level of understanding when it comes to:

- The principles of object-oriented programming paradigms including abstraction, encapsulation, inheritance, and polymorphism (Refer to task 7.1P and 10.2D)
- Use of an object-oriented programming language such as C#, and associated class libraries (Swin-Games and NUnit Framework), to develop object-oriented programs.
- Design, develop, test and debug programs using object-oriented principles with an integrated development environment (Visual Studio 2017).
- Constructing UML and sequence diagrams to communicate an object-oriented solution

Reflection

The most important things I learnt:

For starters, I learnt how to code in C# and make it object-oriented instead of procedural. I also learnt about what object-oriented programming is, and why is it used over procedural programming. I learnt that the object-oriented paradigm makes modifying and maintaining a program easier. This is especially true with the iterations we had to do on a weekly basis. The complexity gradually increased and the program being object-oriented made it easier to implement new changes, without breaking everything.
Not only did I learn about the four major principles of object-oriented programming, but I also learnt about the importance of other concepts: cohesion, coupling, roles, collaborations, and responsibilities. I also learnt about C# programming artefacts: classes, objects, interfaces, methods, fields (Refer to Task 7.1P).

The things that helped me most were:

The things that helped me most were the lectures and tutorials. The lectures covered the theory and explained why we did things. The theory covered in the lecture was relevant content, I needed in order to complete the Doubtfire tasks. The tutorial reinforced my understanding of what was covered in the lecture that week and any topics that I didn't quite understand or got wrong were usually resolved in the tutorials. In addition to that, I knew that the programming helpdesk was always an option.

I found the following topics particularly challenging:

One of the topics I found challenging were interfaces. When we were first introduced to interfaces, I didn't understand it at all. But after completing iteration 4, where we were required to implement an interface called "IHaveInventory". An interface was essentially inheritance but because Bags and Players were already inheriting from GameObject, and C# doesn't allow multiple inheritances, an interface was required. By implementing this interface, Bags and Players were treated as containers. By turning Bags and Players into containers, I was able to implement the Look Command.
The other difficult topic was unit testing, or at least coming up with ideas on what tests to run. The first couples of iterations specified what tests needed to run but those specifications gradually disappeared. So, I needed to think of my own tests and how to test them.

I found the following topics particularly interesting:

The interesting topics that I went through were the ones regarding other languages. This included tasks 10.1 – "Clock" and 10.2 – "Other Languages". For 10.1, I wrote a clock program using JavaScript. Unlike C#, I learnt that JavaScript didn't use classes, even though the keyboard "class" existed, it was basically a function.
With 10.2, I discussed the differences between C# and JavaScript when it came to inheritance. C# would inherit from either a class or interfaces, while JavaScript had something called "prototypes". By using prototypes, every object wouldn't need to have their own set of methods but instead, they shared it. An additional benefit with prototypes was that it would allow you to add functions to external libraries, something that isn't available to C#.

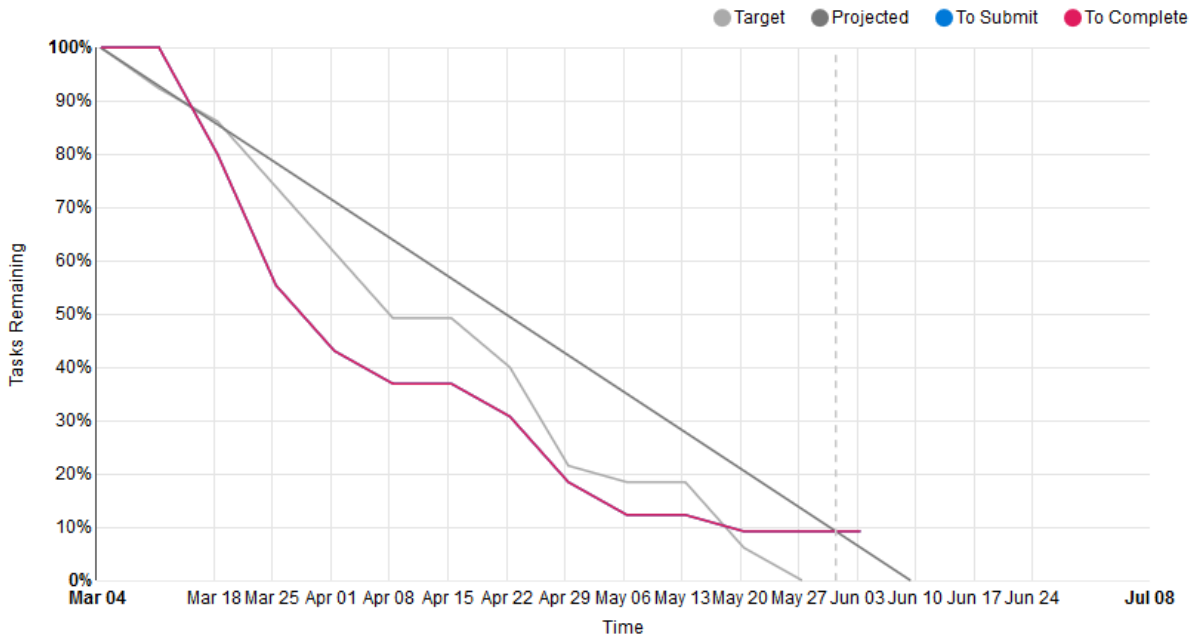
I feel I learnt these topics, concepts, and/or tools really well:

I feel like I did well with all the concepts in object-oriented programming, including the four principles. I was able to demonstrate this through the Doubtfire tasks, especially the final iteration (Refer to 9.2D).

I still need to work on the following areas:

I still need to work on designing a good program. Although I understand the object-oriented programming paradigm, designing a good program is another matter. By completing this unit, I feel like the software I'm able to produce are better than they would have been without the object-oriented paradigm, it is still far from good programs. I usually can't come up with any good designs until I've coded a portion of the program first.

My progress in this unit was …:



My progress throughout this unit has been pretty good. The progress graph above shows that for the most part, I was on task except for the last week or two. This was due to my commitments with another portfolio unit.

This unit will help me in the future:

Completing this unit will prepare me to work on more complex programs in the future whether it's at university or during my career. It will also hopefully help me get closer to creating good software, which is low coupling and high cohesion. This unit is a prerequisite for one of my final year units, "Software Engineering Project A", so I hope I can demonstrate what I've learnt from this unit there.

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

If I were to redo this unit, I would work on the Doubtfire tasks just like I did this semester, in addition to that, I would have started working on my custom program much earlier. Of course, that would have meant that I would need to do a lot of researching on design patterns, on my own. But ultimately, it would have given me more time to work and deliver a custom program and possibly a research project as well.