COS20007

OBJECT ORIENTED PROGRAMMING

Learning Summary Report

Le Nho Bach
103487884

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: **Le Nho Bach**

Portfolio Overview

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

I think I deserve an HD level in this course, thanks to my overall accomplishment. For more details, I have completed all the tasks from week 1 to week 11, including the research and custom program at a high level. Besides, I reached all unit learning outcome requirements and can structure as well as create a program using OOP. For more details:

- About Pass tasks, I have completed all of them, from P tasks from Swin Adventure plan, Counter Clock, to Shape Drawer using Splashkit. I tried to implement mostly everything I have learnt from the class to accomplish those tasks. I have done the Semester Test successfully, and know how to apply OOP in Python.
- I also completed all Credit tasks. They are iterations from the Swin Adventure plan. Despite having some troubles in having a good design, I think that the knowledge from the lecture is still enough to overcome those tasks.
- About Distinction and High Distinction tasks, they took me much more time and required me to find more information on the internet. I completed the custom program at a high level with the feedback and recommendations from the lecturer. I also accomplished the research report about design patterns with high complexity and detail.
- After the course, I am now having a deep understanding of principles. I can explain clearly and implement abstraction, polymorphism, inheritance, and encapsulation in my code. I also know how to apply OO principles in other programming languages.
- I also gain many skills, including communication skills and digital literacies. This course also improves my design, test, and debug skills. Moreover, I can construct and present diagrams and UMLs, which help me in designing programs.

Reflection

The most important things I learnt:

I think that the most important thing for me in this course is how to apply the OO principles in improving my program design. After understanding deeply abstraction, encapsulation, polymorphism and inheritance, I see that my code is shorter and clearer. For me, these four concepts are the most remarkable definitions throughout the course and have the biggest impact on changing my design thinking. Besides these principles, I think other aspects like objects, pointers, interfaces or delegations, libraries, and design patterns are also useful and play a significant role in improving my coding skill.

The things that helped me most were:

As I mentioned above, the four principles: abstraction, encapsulation, polymorphism and inheritance are the most vital knowledge I have gained from the course. They simplify my code, make it clearer and shorter, and save me a lot of time each time I need to debug. From the materials from the course and outside the schools, these principles are significant for my future thanks to their influence. Having a deep understanding of these concepts can enhance the security, extensibility, and reusability of my program

I found the following topics particularly challenging:

In my HD-level custom program, my lecturer asked me to make the program able to save the game progress. At that moment, I wonder how can I save the data of many objects, with different enumerations and structs. If I translated all of them into text and save them, the structure would be extremely complicated. I read about the serialization but it was not effective, too. After all, I decided to make a dictionary to translate the object to text through its name, and it worked. Besides, I think understand deeply about the design patterns and the differences between different OOP language like C#, Java and Python are also a bit challenging.

I found the following topics particularly interesting:

When I read about design patterns, I struggled with the number of patterns. I was perplexed about patterns' similarities and concepts. However, I also saw that these patterns were potentially useful and yes, their applications are significantly interesting. Singleton seems to be the most simple pattern, but I was impressed about its impact in designing. It improved my custom program a lot. I were also interested in Bridge pattern or Template Method pattern.

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

I think that I use the abstraction principle quite good. I also like this principle the most compared to the three others. I used it a lot in simplified my code and made it easier to debug. Besides, I also learned more about the OOP in Python due to my career orientation, and now I can apply it very well. Structuring UML diagrams now is also one of my strengths, after I can understand the OO principles and know how to draw UML diagrams by using Plantuml.

I still need to work on the following areas:

In my point of view, I still need to practice more design patterns to fully understand their merits and demerits. Besides, due to the number of patterns, I also need more time to know when should I use patterns and which pattern should I use. Besides, I need to try to see more similarities and implement more inheritance principle in programming.

This unit will help me in the future:

I think knowing how to write clear code is never redundant in any field of IT. The principles also improve the extensibility, security and flexibility of my code in the future. The design patterns are also remarkable, but I need more time to practice them. Besides, skills like digital literacy, presentation skills and communication skill are also important and help me in every field in the future.

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

I think that I did quite well in this course. However, if I needed to do this course again, I would spend more time discovering design patterns because they are a bit perplexing. Otherwise, I would ask my lecturer and my peers more about my custom code to have more and more ideas.

Other…:

I believe that this course contains core knowledge about programming, and separates the new programmers and old programmers. Understanding deeply the principles and practising more and more are keys for me to thrive more in the future. To overcome the involved difficulties, I need to spend more time on coding as well as reading books and materials.