COS20007

Object-Oriented Programming

NGUYEN DUC MANH

Lecturer: Nguyen Dang Khoa

Self-Assessment Details

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

In addition to the checklists, please append the following supporting appendices to your portfolio:

- **Appendix I:** A list of screenshots showing each task title and the feedback from your tutor upon submission.
    - ➢ I have submitted this on an independent .zip file.

- **Appendix II:** A summary of task corrections. For each task, if you have made any amendments or corrections that have not been previously assessed by your tutor, please summarize these changes. If there are no corrections or updates, indicate that the task submission from previous weeks (up to Week 12) remains unchanged.
    - ➢ [7.1P - Case Study - Iteration 5 - Console Application typing prior iterations] I added the missing screenshort of the running program.
    - ➢ The other tasks remain unchanged.

- **Appendix III:** A list of your up-to-date **corrected** task submissions in PDF format. These should reflect any changes mentioned in Appendix II. If your submission has already been assessed by your tutor on a weekly basis and there have been no changes, you do not need to resubmit the PDFs, as we already have them.
    - ➢ The correction has already been assessed as “completed” by the tutor.

- **Appendix IV:** The source code of all your previous task submissions in compressed zip format. Regardless of whether you made corrections or not, you must include all source code for your task submissions. Compress the source code into one or more zip files and submit them to Canvas along with your portfolio report. For example, the first .zip file can contain all the source code for the Hello-World program, The Counter, and Clock projects. The second .zip file should contain all the C# source code (.cs files) and test case implementations for your task submissions related to the Shape Drawing project. Finally, the third .zip file should include all the C# source code (.cs files) and test case implementations for your task submissions related to the Swin-Adventure case study.
    - ➢ I have submitted this on independent .zip files.

- **Appendix V.** If you repeatedly receive minor/major revision feedback after the T1-1 - Semester Test Fix and Resubmit, you can still resubmit your corrections in this appendix. You need to summarize how you addressed the feedback and submit your full test solution again. However, this will result in a mark deduction from your final grade.
- Alternatively, if you failed to submit the T1-1 - Semester Test Fix and Resubmit by the deadline, you can still submit it in this appendix. However, this will result in a **FAIL grade** for the unit if your test resubmission is incorrect.
    - ➢ My T1-Semester Test marked as completed.

Remarks:

Failure to provide the source code for any task submission will result in that task not being assessed, even if the source code is printed and included in the PDF submission. The teaching team needs the source code to execute it and verify correctness. Additionally, the source code will be used for plagiarism detection and academic integrity checks.

Self-Assessment Statement

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

Minimum Pass Checklist

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

Minimum Credit Checklist (in addition to Pass Checklist)

	Included
All Credit Tasks are Complete	✓

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
Custom program meets Distinction criteria & Interview booked	✓
Design report has UML diagrams and screenshots of program	✓

Minimum Low-Band (80 – 89) High Distinction Checklist (in addition to Distinction Checklist)

	Included
Custom project meets HD requirements	✓

Minimum High-Band (90 – 100) High Distinction Checklist (in addition to Low-Band High Distinction Checklist)

	Included
Research project meets requirements	✓

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. Failure to meet this requirement will result in a failing grade for the unit.
Failure to provide the source code for any task submission will result in that task not being assessed, even if the task is included in PDF format.

Signature: **Nguyen Duc Manh**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieved all Unit Learning Outcomes for COS20007 Unit Title to a **HD** level.

I think my work for this unit should be at the HD level because:

- I’ve finished all the lower-level tasks of this unit and all of them have been approved as “complete” by the tutor.
- I’m currently doing my Custom Program and try my best to meet the HD requirements. To be more specific, I’ve implemented some of the design pattern principles from the book provided by Swinburne online library (***Design Patterns: Elements of Reusable Object-Oriented Software***).
- For the Custom Program idea, I chose to make a “chess-based” game because it is the primary example of how different “objects” on the board interact with each other. But I think the pure chess game is quite normal and has been made and utilized by most of the students. Therefore, I chose a chess-like game with much more difficult rules and logic to challenge myself.

Task Summary

To demonstrate my learning in this unit, I would like the following tasks to be considered part of my portfolio:

1. Coding Tasks: I've completed all the **Pass** tasks but here I chose all the **Credit** tasks to show that I have done harder tasks
    - 5.3C - Drawing Program — Saving and Loading with Customized Payload **(completed)**
    - 7.2C - Case Study — Advanced Iteration 6: Locations **(completed)**
    - 9.2C - Case Study - Advanced Iteration 7: Paths **(completed)**
    - 10.1C - Case Study - Advanced Iteration 8 - Command Processor **(completed)**
    - 11.1P - Clock in Another Language with Practical Benchmarks **(completed)**
2. Report Tasks:
    - 6.3D - Custom Program Initial Plan **(submitted)**
    - 6.4D - D Level Custom Program: UML Design Level, Implementation, and User Documents **(submitted)**
    - 6.5HD - HD Level Custom Program Initial Plan **(submitted)**
    - 6.6HD - HD Level Custom Program **(submitted)**
    - 9.3HD - Research Project Initial Plan **(submitted)**
    - 9.4HD - Research Project **(submitted)**

Reflection

The most important things I learnt:

This unit saw me gain a definite understanding of the Object-Oriented Programming (OOP) principles of encapsulation, inheritance, polymorphism, and abstraction. They were unfamiliar to me, but they were essential for writing maintainable and reusable code. I also gained an understanding of how design patterns could be used to structure code more efficiently, which taught me how professional software is written. Apart from the actual technical content, I also enhanced my problem-solving skill and planning and organizing of larger coding projects.

The things that helped me most were:

- The hands-on coding exercises that allowed me to apply OOP as soon as I had studied it.
- The tutor's explanations and examples shown, especially during code reviews.
- The Custom Program assignment, which enabled me to explore a more complex subject and see how different classes and objects communicate.
- The design patterns book, introducing me to out-of-the-box thinking of how to approach problem-solving via code.

I found the following topics particularly challenging:

Personally, the following subjects proved to be quite challenging for me:
I found polymorphism and abstract classes a bit confusing at first, especially when combining them with inheritance. It took me a few practical tasks to really understand how they work and when to use them. The concept of cohesion and coupling in class design also required more time to fully grasp.

I found the following topics particularly interesting:

Design patterns were the most fascinating to me. I wanted to understand how the pre-existing solutions could simplify complex programming problems. The MVC (Model-View-Controller) pattern was also fascinating because it showed me how large programs such as games or web apps are organized.

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

- Personally, I think I learnt the concept of Abstraction and Polymorphism the best. I feel like I can easily grasp those ideas and philosophies quickly.
- At the end of the unit, I’m confident that I can use Visual Studio 2022 at a decent level because I’ve spent so much time with it.

I still need to work on the following areas:

I think I still need to learn to outline the whole project ideas and progress because many times I felt lost and didn’t really know what I was supposed to do to complete the project. The second thing is the UML since it is quite complicated for me.

My progress in this unit was …:

Overall, my performance during this unit was consistent. I finished all the work required by the tutor within deadlines and continued receiving feedback from the tutor. I fully engaged in feedback comments from the tutor and used that feedback to improve later work. Finishing the Custom Program also provided me with an opportunity to reflect on how I approach large projects, such as time management, debugging, and testing.

This unit will help me in the future:

What I learned this unit will apply to a lot of my course and professional work. A lot of current programming is used OOP principles, and this will be the foundation for the rest of my learning in web development, software design, and mobile app development. Knowing how to initialize the idea, following UMLs, implementing OOP principles will be especially helpful when you are working on group projects or on bigger software programs.

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

If I had the opportunity to redo this unit, I would begin learning design patterns sooner and use them from the outset of the Custom Program. I would also spend more time working out the organization of my classes prior to diving in and coding them, which would minimize refactoring in the future. For future units, I will also try to test code more along the way, rather than waiting until the end of the unit.

Other…:

None.