COS20007

Object-Oriented Programming

Learning Summary Report

JOSHUA SANJAY KING
105282612/J23039507

Self-Assessment Details

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

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.

Signature: **JOSHUA SANJAY KING**

Portfolio Overview

Dear lecturer,
I am writing to justify why my portfolio demonstrates that I have achieved a Distinction level for the COS20007 unit. The work included in this portfolio demonstrates that I fully achieved the unit learning outcomes (ULOs) and therefore providing evidence that I have the ability to apply object-oriented programming (OOP) principles effectively.

Throughout the unit, I have demonstrated a strong grasp of abstraction, encapsulation, inheritance, and polymorphism. For example, my implementation of the **ShapeDrawer** project showcased these principles through clearly defined classes, effective data encapsulation and polymorphic methods that allowed dynamic behavior in the program. Using these principles have developed a program that uses scalable and reusable code structures.

I have successfully developed a text-based game using an object-oriented language with C# named **SwinAdventure**. During development, many classes where reused virtual methods from a parent class “GameObject”. In addition, many classes and unit testing were developed along the process, which demonstrated modular programming. I utilized NUnit to create unit tests for critical methods, allowing myself to test certain methods before making the full functional program. Having unit testing has helped fixed subtle errors in both logic and implementation. This project reflects a clear understanding of advanced OOP concepts.

Besides finishing the tasks given by this subject, I had also developed a 2D game which also uses C# for my distinction task. In addition to using every object-oriented principle such as inheritance, polymorphism, encapsulation and abstraction, I also learnt sprite designing and basic animating concepts needed to make the game more engaging and fun for the players. All research was done independently as additional concepts needed for my custom program is beyond the scope for this subject.

I have consistently reflected on my work, identifying areas for improvement and aligning my practices with industry standards. My projects had a clear and clean coding implementations, meaning documentation and thoughtful design decisions that balance functionality with maintainability.

In conclusion, my projects have shown that I have meet the distinction criteria. My projects demonstrate not only technical competency but also creativity, problem-solving and a commitment to excellence. I believe this portfolio is a strong representation of my ability to apply all ULOs effectively at a distinction level.

Task Summary

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

- 2.2P - Counter Class – COMPLETED
- 2.3P - Drawing Program - A Basic Shape – COMPLETED
- 2.4P - Case Study Iteration 1 - Identifiable Object – COMPLETED
- 3.1P - Clock Class – COMPLETED
- 3.2P - The Stack and Heap - COMPLETED
- 3.3P - Drawing Program - A Drawing Class – COMPLETED
- 4.1P - Drawing Program - Multiple Shape Kinds (UML Diagrams updated) - COMPLETED
- 4.2P - Case Study - Iteration 2 - Players Items and Inventory – COMPLETED
- 5.1P - In Person Check-in 2 - Drawing Program – COMPLETED
- 5.2P - Case Study - Iteration 3 – Bags – COMPLETED
- 5.3C - Drawing Program - Saving and Loading – COMPLETED
- 6.1P - Case Study - Iteration 4 - Look Command – COMPLETED
- 7.1P - Case Study - Iteration 5 - Tying it Together – COMPLETED
- 7.2C - Case Study - Iteration 6 – Locations – NEED UML FIX & MINOR CORRECTIONS
- T1-1 - Semester Test Fix and Resubmit – COMPLETED
- 9.2C - Case Study - Iteration 7 – Paths – NEED UML FIX
- 10.1C - Case Study - Iteration 8 - Command Processor – NEED UML FIX
- 11.1P - Clock in Another Language – LATE SUBMISSION

The list above is evidence that I understand the **OOP concepts** in order to complete all of the **Pass tasks** and most of the **Credit tasks.** The tasks above are step-by-step to create object-oriented programs. In this semester, I have created a clock counter program, a shape drawer program that can draw three different shapes, and a text-based game that has many in depth details such as being able to open inventory, look around, move, and look at things. Completing all these tasks has shown that I am able to understand how to handle object-oriented programs.

Reflection

The most important things I learnt:

The most important things I learnt in this unit were the core principles of object-oriented programming, such as abstraction, encapsulation, inheritance and polymorphism. These concepts helped me understand how to structure programs in a way that is modular, reusable and easy to maintain. Additionally, I learned the importance of testing debugging by using tools like NUnit. Unit testing is important as it allows programmers to test individual classes and methods before developing the full program. This subject has also taught me problem-solving skills and independent learning.

The things that helped me most were:

1. Practical Assignments: Working on object-oriented programs such as **ShapeDrawer** and **SwinAdventure** helped apply concepts in a practical setting.
2. Check-in sessions with Lecturer: Having these check-in sessions has shown myself confidence in OOP concepts as I was able to explain them clearly. In addition, the lecturer also pointed out some minor flaws in the program and gave valuable feedback to help improve the quality of my **SwinAdventure**.
3. Peer collaboration: As this is the first semester working with OOP, I needed help for certain tasks from peers that are experience in OOP.

I found the following topics particularly challenging:

1. UML diagrams: Creating accurate UML diagrams required lots of attention to detail, specifically the diagram symbols.
2. Polymorphism: I was able to understand partially on how to apply it practically but it was difficult for me to fully understand.

I found the following topics particularly interesting:

1. Developing text-based game: creating SwinAdventure was an engaging and fun process.
2. Abstraction and encapsulation: These principles helped hide complexity while focusing on essential functionalities.

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

1. Inheritance and Reusability – **SwinAdventure** project showed my strong understanding of reusing parent class methods and extending functionality through virtual methods.
2. Unit testing – creating unit testing helped ensure that classes were working as intended
3. Splashkit – I learnt how to use the Splashkit library for graphics programming.

I still need to work on the following areas:

1. UML diagrams – I still struggle in creating accurate UML diagrams
2. Optimizing code for performance: my code could improve efficiency by better understanding algorithm design and optimization techniques.

My progress in this unit was …:

My progress in this unit was steady and consistent. I submitted almost every work on time, barely skipped any class and seek for guidance whenever needed. Collaborating with my lecturer and peers contributed to my learning. However, balancing multiple projects at

times was challenging, which pushed me to improve my time management skills. This experience taught me to plan tasks better and focus on incremental progress.

This unit will help me in the future:

Working on huge programs has helped me learned on how to start small and not get distracted at the workload. Additionally, the independent learning skills I developed will help me tackle new challenges in my academic and professional journey.

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

[List and explain, how will you approach learning in the future? What things worked well, but what could you change to make sure you did better next time?]

1. Start project earlier: beginning assignments earlier would allow more time for corrections and future tasks.
2. Participate more in class: engaging in classes could further enhance my understanding of complex topics

Other…:

This unit not only strengthened my technical skills but also taught me the importance of persistence and adaptability. Overcoming challenges in assignments showed me the value of seeking help and embracing feedback.