COS20007

Object-Oriented Programming

JUSTIN NGUYEN

104309099

Learning Summary Report

# 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 |
| The custom program meets Distinction criteria; the Interview is booked | ✓ |
| The design report has UML diagrams and screenshots of the program | ✓ |

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

|  |  |
| --- | --- |
|  | Included |
| The custom project meets HD requirements |  |

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

|  |  |
| --- | --- |
|  | Included |
| The research project meets the requirements |  |

# Declaration

I declare that this portfolio is my 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: **Justin Nguyen**

# Portfolio Overview

This portfolio serves as a testament to my successful achievement of all Unit Learning Outcomes for the COS20007 Unit Title. It showcases my comprehensive understanding and proficient application of Object-Oriented Programming concepts, reaching a **Distinction** level.

My attainment of the Distinction level was a result of completing all the required Pass and Credit tasks, passing the Semester test, and having my custom program plan marked by my tutor.

In this unit, I have learned the main concepts of OOP, including abstraction, encapsulation, inheritance, and polymorphism, by doing tasks like ShapeDrawing and SwinAdventure. I also learned to use the SplashKit library to create a GUI in ShapeDrawing. Throughout the unit, I learned how to test my programs to check for functionalities and mistakes when writing unit tests for the programs. Also, before writing any programs, I am required to draw the UML diagrams to illustrate better how my programs work and which objects they will contain. It is a core unit of my course, so ensuring you learn correctly and apply what you have learned for future units and your future career is promising.

# Task Summary

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

* Pass tasks:

+ Drawing Shapes tasks (2.3P, 3.3P, and 4.1P): This helps me get accustomed to using the SplashKit library and the concept of OOP (inheritance, encapsulation, polymorphism, and abstraction).

+ SwinAdventure (2.4P, 4.2P, 5.2P, 6.1P, and 7.1P): First steps to create a console-based game with different categories (characters, inventory, simple look command, and a main program to operate).

+ OOP concepts and knowledge (1.1P, 3.3P, and 6.2P): a summary of what concepts and knowledge I have learned this semester and a review of the last semester in the Introduction to the Programming unit.

* Credit tasks:

+ Drawing Shapes tasks (6.3C): learning to save and load files.

+ SwinAdventure (7.2C, 9.2C, and 10.1C): more advanced categories (location, path, move command, and a command processor)

* Distinction tasks: the custom program (learning to use WPF to create a GUI and Bitmap to render images)

# Reflection

## The most important things I learned:

* OOP concepts: encapsulation, abstraction, inheritance, polymorphism, roles and responsibilities, etc.
* C# language: I learned C++ in high school, so it is good for me to understand the logic of C# language quickly.
* Testing: This is a crucial step to check for mistakes and ensure the programs work correctly.
* External library: SplashKit is used to create a GUI, and WPF is used to create a simple console.

## The things that helped me most were:

* Lectures: The lectures provide essential knowledge for me to finish the tasks.
* Help Desk: The help desk helps me catch up with the deadlines of the tasks and is the place to ask for things you want to clarify.
* Workshops: I attended the workshops frequently to sign off on my work and ask questions.

## I found the following topics particularly challenging:

* One of the significant challenges I encountered was with SplashKit, a widely used library for creating a GUI. I found it particularly challenging to use correctly, especially when determining the coordinates to draw lines and shapes. To overcome this, I decided to use WPF for my custom program, which allowed me to select the elements I wanted to include in the GUI, thereby reducing the amount of code I needed to write.
* Unit tests: Testing is an integral part of the programs, but sometimes, the tests come out of error due to minor mistakes (the change of objects, the length of the string, etc.).

## I found the following topics particularly interesting:

* SplashKit: While the library took me much time to understand its logic, seeing all the shapes drawn onto a GUI was enjoyable.

## I feel I learned these topics, concepts, and tools well:

* I learned the main concepts of OOP well and adapted my understanding to every task in the unit.

## I still need to work on the following areas:

* My focus for improvement lies in the GUI aspect. While both SplashKit and WPF can create GUIs, SplashKit demands extensive code for GUI generation, and WPF is limited to simple console-based GUIs. To develop more complex systems, I aim to explore diverse tools for creating visually appealing GUIs.

## My progress in this unit was:

* I submitted most of my work ahead of deadlines, except for some tasks that required a long period of coding and check-in, like Saving and Loading Shapes or SwinAdventure. Also, I have to do other work in other units, so sometimes, the tasks in this unit put me under lots of time pressure.

## This unit will help me in the future:

* This unit will help me understand the basic concepts of OOP and the steps to developing a program, from drawing UML diagrams to writing and testing programs. It is a foundation unit in my course, so it is essential for more advanced units in the course and in my future career.

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

* If I redid this unit, I would finish the work ahead of deadlines every time and distribute my time and effort concisely so as not to put myself under pressure.