Learning Summary Report

	Pass	Credit	Distinction	High Distinction
Self-Assessment (please tick)				✓

Self-assessment Statement

	Included (please tick)
Learning Summary Report (This document)	✓
Semester Test with all corrections	✓
At least 50% of Pass Tasks signed off as Complete	✓

Minimum Pass Checklist

	Included (please tick)
All of the Pass requirements	✓
100% of Pass Tasks signed off as Complete	✓
Evidence of credit tasks that have been completed	✓

Minimum Credit Checklist, in addition to Pass Checklist

	Included (please tick)
All of the Credit requirements	✓
100% of Credit Tasks signed off as Complete	✓
Code for your program that demonstrates good OO design *including the use of polymorphism, implementing an interface and managing collections of objects*.	✓
UML class diagrams and UML sequence diagrams that communicate the design your custom program	✓
Description of what your program does and screenshots of your program in action	✓

Minimum Distinction Checklist, in addition to Credit Checklist

	Included (please tick)
All of the Distinction requirements	✓
Research report and associated pieces	✓

Minimum High Distinction Checklist, in addition to Distinction Checklist

Introduction

This section summarises what I learned about Object Oriented Programming. It includes a justification of the assessment pieces included in the portfolio, details of the coverage of the unit learning outcomes, and a reflection on my learning.

Overview of Pieces Included

Briefly describe what you have included in each section of the report.

1. Pass Task 3: a console project which contains a Resource class and a Resource Category enumeration.
2. Pass Task 7: develop a unit test for a C# program regarding showing id and status of the resource class using NUnit.Framework
3. Pass Task 8: documenting code with XML documentation
4. Pass Task 10: design software using UML class diagram and develop console program regarding add or remove resource from a project based on it.
5. Pass Task 12: design software using UML class diagram and develop a console program regarding add or remove resource from a project with abstraction, inheritance, and polymorphism principles with abstract, virtual, and override keywords
6. Pass Task 13: develop a console program for administrator usage based on real industry case written with abstraction, encapsulation, inheritance, and polymorphism principles.
7. Credit Task 1: constructing a class diagram for software regarding student grading system with abstraction, encapsulation, inheritance, and polymorphism principles
8. Credit Task 2: using Netbeans to develop a c++ program for student grading system based on the proposed UML class diagram
9. Distinction Task 1: construct a UML class diagram for my custom program: A Day in Candy Shop.
10. Distinction Task 2: construct UML sequence diagram for my custom program: A Day in Candy Shop.
11. Distinction Task 3: develop unit test for my custom program: A Day in Candy Shop.
12. Custom Program's user manual: describe the gameplay of A Day in Candy Shop.
13. Custom Program: A Day in Candy Shop.
14. High Distinction Research Report: research paper about refactoring by using design pattern in software development.

Coverage of the Intended Learning Outcomes

This section outlines how the pieces I have included demonstrate the depth of my understanding in relation to each of the unit's intended learning outcomes.

ILO 1: Object Oriented Principles

Explain the principles of the object oriented programming paradigm specifically including abstraction, encapsulation, inheritance and polymorphism.

The following pieces demonstrate my ability in relation to this ILO:

- Pass Task 3,7,8,10 demonstrate my ability to develop a console program written with encapsulation principle.
- Pass Task 12 demonstrate my ability to develop a console program regarding add or remove resource from a project written with abstraction, encapsulation, inheritance, and polymorphism principles.
- Pass Task 13 demonstrate my ability to develop a console program for administrator usage based on real industry case written with abstraction, encapsulation, inheritance, and polymorphism principles.
- Credit Task 1 demonstrate my understanding in constructing a class diagram for software regarding student grading system with abstraction, encapsulation, inheritance, and polymorphism principles.

- Distinction Task 1 demonstrate my understanding in constructing a class diagram for my custom program with object-oriented principle.
- High Distinction Research Report demonstrate my deep understand in object-oriented principles.

ILO 2: Language Syntax

Use an object oriented programming language, and associated class libraries, to develop object oriented programs.

The following pieces demonstrate my ability in relation to this ILO:

- Pass Task 3 demonstrate my ability to develop a console program regarding showing id and status of the resource class using C#. I learned what access modifiers are and enum keyword to create an enumeration class.
- Pass Task 7,8,10,12 demonstrate my ability to develop a unit test for C# program using NUnit.Framework. I learnt what use of [TestFixture()] and [Test()].
- Pass Task 8,10,12 demonstrate my ability in understanding XML documentation.
- Pass Task 10 demonstrate my ability in understanding the practical use of indexer in C#.
- Pass Task 12,13 demonstrate my ability in understanding the meaning and practical use of abstract, virtual and override keywords.
- Credit Task 2 demonstrate my understanding in c++ syntax by developing a console program for student grading system based on the proposed UML class diagram.

ILO 3: Writing Programs

Design, develop, test, and debug programs using object oriented principles in conjunction with an integrated development environment.

The following pieces demonstrate my ability in relation to this ILO:

- Pass Task 3 demonstrate my ability to develop a console program regarding showing id and status of the resource class using C#. I learned how to use access modifiers to write a console program with encapsulation principle. I also used enum keyword to create an enumeration class.
- Pass Task 7 demonstrate my ability to develop a unit test for the C# program regarding showing id and status of the resource class using NUnit.Framework. I learned how to indicate and develop test class by [TestFixture()] and indicate and develop test method by [Test()].
- Pass Task 8,10,12 demonstrate my ability in documenting code with XML documentation.
- Pass Task 10 demonstrate my ability to develop a console program regarding add or remove resource from a project according to the UML class diagram with the help of Xamarin Studio.
- Pass Task 12 demonstrate my ability to develop a console program regarding add or remove resource from a project with abstraction, inheritance, and polymorphism principles with abstract, virtual, and override keywords.
- Pass Task 13 demonstrate my ability to develop a console program for administrator usage based on real industry case according to the proposed UML class diagram.
- Credit Task 2 demonstrate my understanding in using Netbeans to develop a c++ program for student grading system based on the proposed UML class diagram.
- Custom Program demonstrate I have the ability to develop software by object-oriented approach.

ILO 4: Object Oriented Design

Construct appropriate diagrams and textual descriptions to communicate the static structure and dynamic behaviour of an object oriented solution.

The following pieces demonstrate my ability in relation to this ILO:

- Pass Task 10 demonstrate my ability in designing object-oriented program regarding add or remove resource from a project by using UML class diagram. I learned how to present aggregation and dependency in the UML class diagram.
- Pass Task 12 demonstrate my ability in designing object-oriented program with encapsulation, abstraction, inheritance, and polymorphism by using UML class diagram. I learned how to present aggregation and inheritance in the UML class diagram.
- Pass Task 13 demonstrate my ability in designing object-oriented program with encapsulation, abstraction, inheritance, and polymorphism by using UML class diagram. I learned how to present strategy design pattern in UML class diagram.
- Credit Task 1 demonstrate my understanding in constructing a class diagram for software regarding student grading system with abstraction, encapsulation, inheritance, and polymorphism principles.
- Distinction Task 1 demonstrate my ability in constructing a class diagram for my custom program with object-oriented principle.

ILO 5: Program Quality

Describe and explain the factors that contribute to a good object oriented solution, reflecting on your own experiences and drawing upon accepted good practices.

The following pieces demonstrate my ability in relation to this ILO:

- Pass Task 10 demonstrate my ability in designing software regarding add or remove resource from a project with correct relationship between classes which would lead to better program quality.
- Pass Task 12 demonstrate my ability to develop high flexibility software with runtime polymorphism.
- Pass Task 13 demonstrate my ability to develop high flexibility and high modularity software for administrator usage with design pattern.
- Credit Task 1 demonstrate my understanding in designing high flexibility and high modularity software for student grading system with an abstract factory pattern, observer pattern, template pattern, and singleton pattern.
- Distinction Task 1, 2, 3, Custom Program and HD research report demonstrate my understanding in the factor that contributes to a high flexibility and modularity software.
- High Distinction Research Report demonstrate my deep understanding of what constitutes a good object-oriented solution.

Reflection

The most important things I learned:

The most important things I learned is how grit helped myself success in this unit. For me talent does not matter much to become a moderate level programmer. Interest on a subject, willing to learn and hardworking contribute most for successful learning. Moreover, to learn from experienced teacher is the crucial way to gain an insight into the field.

While taking two portfolio unit together, I gain experience in time management. I feel more agreed with the quotes "Whatever your life's work is, do it well. A man should do his job so well that the living, the dead, and the unborn could do it no better." – by Martin Luther King, Jr.

By doing all the pass task and credit task myself, I managed to pass my unit test with good result. “There is no substitute for hard work.” – Thomas A.Edison.

I also improve my ability to find reference on Google by searching with good keywords while trying to complete my pass task and credit task.

Another important thing I learned is that the unit plan by university definitely make sense and thus spending effort in it would not only get me a higher GPA but also act as a stepping stone for harder difficulty unit.

I appreciate more the value in previously taken unit e.g. Introduction to Programming which provides me a basic to object-oriented programming. Thus, I should work harder in all unit.

The most important technical stuff I learned is the four pillar of object-oriented programming. The first principle I learned is encapsulation. Encapsulation is the binding of object state(fields) and behaviour(methods) together. Encapsulation help increase the chance to discover bug and information hiding.

The second principle I learned is abstraction. Abstraction is a process of hiding unnecessary details of an object from the client. Abstraction helps simplify the complexity of the software.

The third principle I learned is inheritance. One class could acquire the properties and behaviour of another class by inheriting it. Inheritance helps reduce the effort needed in developing software by reuse of code. However, inheritance should not be the purpose for reuse of code as inheritance provides tight coupling between classes. In most scenario, reuse code by composition is a better strategy than inheritance.

The fourth principle I learned is polymorphism. Polymorphism is one of the crucial features of object-oriented programming. The same method could have different behaviour based on the actual type of object. This provides great flexibility in software and could be an alternative to if-else conditional statement.

The things that helped me most were:

Lecturer Miss Foo helped me greatly when I faced difficulty in the subject. I also would like to express my deep gratitude to Design Patterns: Elements of Reusable Object-Oriented Software’s author (Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm) and Head First Design Patterns’ author (Eric Freeman, Kathy Sierra, Bert Bates, Elisabeth Robson).

Furthermore, stackoverflow.com should be on the list as I managed to gain much insight regarding object-oriented programming from senior software engineer. Swingame api and its website developed by Swinburne University of Technology provide an easy way for me to develop mine custom program. The open source project developed using Swingame api on Github helped me to understand more about Swingame api.

I found the following topics particularly challenging:

The most challenging topic is Design Patterns. To master all the 23 design patterns listed in Design Patterns: Elements of Reusable Object-Oriented Software and applied it in the correct context would need enormous effort.

While doing my credit task 2 which require implementing by C++, I discovered generic programming. Although I could just use the library provided by other senior software engineers, writing my own one would be exciting and challenging.

The topic of resource management is challenging. In this topic, I learned the difference between stack and heap. The most challenging part of this topic is the failure of resource management will lead to memory leak which could cause the crash of software.

I found the following topics particularly interesting:

I found the topics regarding design pattern very interesting. Complex software would benefit greatly by applying design pattern correctly. I managed to applied state pattern in my custom program to help me construct the program with high flexibility and extensibility. State pattern is too interesting and useful thus my HD research report is about refactoring by using state pattern.

The topic regarding event-driven programming is very interesting. This topic introduces Qt library to me. I would like to try to develop some software with it.

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

I feel I learned the topic regarding inheritance and polymorphism really well. I gain insight about that inheritance may not be a good solution for code reusing as inheritance results in tight coupling between base class and derived class. This could lead to higher difficulty to modify and extend the software when new features are required.

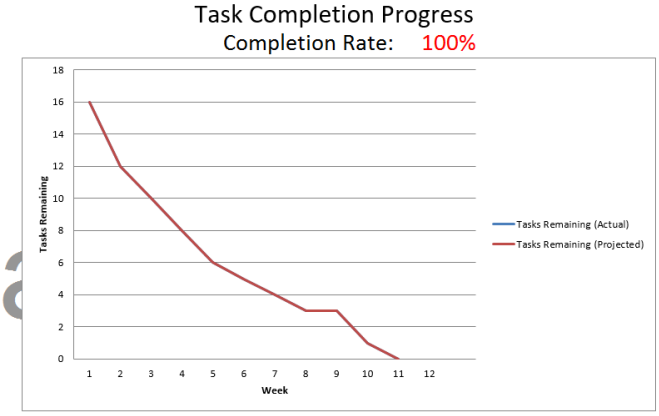
Aggregation or composition would be a better choice since inheritance reduces modularity of the software. However, inheritance plays a crucial role for dynamic binding.

I still need to work on the following areas:

I still need to work more on the skill in architecting software. Although I could implement refactoring by state pattern correctly, I believe I still need to improve my ability in writing a clear report to convey concise message.

My progress in this unit was …:

Week / Lab	Week Begin	Week End	Number of Tasks Assigned	Tasks Completed	Tasks Remaining (Actual)	Tasks Remaining (Projected)
1	4-Mar	10-Mar	4	4	16	16
2	11-Mar	17-Mar	4	4	12	12
3	18-Mar	24-Mar	2	2	10	10
4	25-Mar	31-Mar	2	2	8	8
5	1-Apr	7-Apr	2	2	6	6
6	8-Apr	14-Apr	1	1	5	5
7	22-Apr	28-Apr	1	1	4	4
8	29-Apr	5-May	1	1	3	3
9	6-May	12-May	0	0	3	3
10	13-May	19-May	2	2	1	1
11	20-May	26-May	1	1	0	0
12	27-May	2-Jun	0	0		
Total			20	20		
Completion Rate:				100%		



Note: Fill in cell **F3 till F13** ONLY, other cells and the graph will be calculated automatically
Amend the Graph title; "**Completion Rate**" as accordance to value calculated in cell **F17**

This unit will help me in the future:

This is my second portfolio unit taken, thus managed to complete this unit will help me gain experience and gain confidence in future portfolio unit. By learning how to focus on the lecture and learn hard in the tutorial session, I think I could learn well in harder difficulty unit.

Moreover, this unit will help me gain confidence in other units. This unit delivered most of the foundation of software developing. By paying attention in the lecture and learn hard in the tutorial session, I get more prepared for harder unit e.g. Software Development for Mobile Devices and Data Structure and Patterns.

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

I would learn Swingame api at the start and finish my custom program before week 7 so I could have an easier life at the end of the semester.

Other…:

Concluding Statement

In summary, I believe that I have clearly demonstrated that my portfolio is sufficient to be awarded a HD grade as I had completed all the requirement for HD grade.