Assessment Cover Sheet

This Assessment Cover Sheet is only to be attached to hard copy submission of assessments.

ASSESSMENT DETAILS

Unit title	Object Oriented Programming	Tutorial /Lab Group		Office use only
Unit code	COS20007	Due date	5th/Dec	
Name of lecturer/tutor	Fu Swee Tee			
Assignment title	Final Portfolio Submission (Pass | Credit)			Faculty or school date stamp

STUDENT(S) DETAILS

	Student Name(s)	Student ID Number(s)
(1)	Takeru SONODA	102784225
(2)		
(3)		
(4)		
(5)		
(6)		

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/tutor concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Takeru SONODA	(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment**

Copies of this form can be downloaded from the Student Forms web page at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment/how-to-submit-work.php**

Learning Summary Report

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

Self-assessment Statement

	Included (please tick)
Learning Summary Report (This document)	✓
Assessed Pass Tasks	✓
Semester Test	✓

Minimum Pass Checklist

	Included (please tick)
All of the Pass requirements	✓
Evidence of credit tasks that have been completed	✓

Minimum Credit Checklist, in addition to Pass Checklist

	Included (please tick)
All of the Credit requirements	✓
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.

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

Pass Task 12:

- Abstraction and Encapsulation: This task involved extending the Rack and Box classes to include different types of boxes (Cardboard, Plastic, Acrylic), each with unique properties and behaviors. The design of these classes demonstrates my understanding of abstraction by representing complex real-world objects in a simplified manner. Encapsulation is evident in how each class manages its own data and exposes functionality through methods.
- Inheritance: The creation of specific box types (Cardboard, Plastic, Acrylic) as subclasses of the Box class showcases my ability to use inheritance. This design allows for shared properties and behaviors in the Box class while enabling specific characteristics in the subclasses.
- Polymorphism: The subclasses override and extend the functionality of the Box class, such as the GenerateDetails and Update methods. This demonstrates my understanding of polymorphism, where subclasses can have their own unique behaviors while still conforming to the interface of the superclass.
- Practical Implementation: The implementation of these concepts in C# code, including the use of enumerations, abstract classes, and method overriding, shows my practical skills in applying object-oriented principles in a programming language.

ILO 2: Language Syntax

Pass Task 3: Box Class Implementation:

- Class and Object-Oriented Design: This task involved creating a Box class with specific attributes and methods. It demonstrates my understanding of class design in C#, including the use of constructors, properties, methods, and private fields.
- Enum Usage: The implementation of Type and Size enumerations showcases my ability to use enums in C# for defining a set of named constants, enhancing the readability and maintainability of the code.
- Method Implementation: The CalculateDimension and CalculateDimensionalWeightInKG methods illustrate my skill in implementing functionality that performs specific calculations based on object properties.
- Property Accessors: The use of properties with getters (e.g., WidthInCM, HeightInCM, LengthInCM, Size) reflects my proficiency in encapsulating fields and providing controlled access to the class's data.
- String Building: The GenerateDetails method, which uses StringBuilder, demonstrates my ability to construct and manipulate strings effectively, a common requirement in C# programming.
- Practical Application: The overall implementation of the Box class and the Main method in the Program class, including array initialization and iteration, shows my practical skills in applying C# language syntax to develop a functional, object-oriented program.

ILO 3: Writing Programs

- Pass Task 7:
  - Overview: Adding unit tests to the Box class to validate its functionality.
  - Relevance to Writing Programs:
    - Illustrates the process of designing, developing, and debugging a program.
    - Emphasises the importance of testing in software development.

ILO 4: Object Oriented Design

- Pass Task 8:
  - Overview: Adding comprehensive XML documentation to the Box class.
  - Relevance: While focused on documentation, this task is crucial for object-oriented design as it requires understanding the class's structure and behaviour. Effective documentation is a key part of communicating the design and functionality of object-oriented solutions.
- Credit Task 1
  - Overview: Development of a UML Class Diagram for the Swinburne Room Booking Management System
  - Relevance: Focuses on the design aspect, requiring the creation of a detailed class diagram that accurately represents the relationships and structures within the system, demonstrating a clear understanding of object-oriented design principles.

ILO 5: Program Quality

- Pass Task 8:
    - Overview: Adding comprehensive XML documentation to the Box class.
    - Relevance: Quality in Documentation: Demonstrates the importance of clear documentation in conveying the design and functionality of object-oriented solutions.
    - Attention to Detail: Reflects a commitment to quality through thorough documentation, enhancing code readability and maintainability.

Reflection

The most important things I learned:

- Understanding Object-Oriented Principles: The key learning point for me was grasping the core concepts of object-oriented programming, such as abstraction, encapsulation, inheritance, and polymorphism. These principles have fundamentally changed how I approach software design.
- Practical Application of Theory: Implementing these concepts in various programming tasks, especially in Pass Task 12 and 3, was enlightening. It bridged the gap between theoretical knowledge and practical application.
- Importance of Good Documentation: Through tasks like Pass Task 8, I learned the significance of thorough documentation in making code understandable and maintainable.

The things that helped me most were:

- Hands-On Projects: Applying concepts in real-world scenarios helped solidify my understanding.
- Instructor Feedback: Constructive feedback on assignments was crucial in identifying areas for improvement.

I found the following topics particularly challenging:

- Advanced Polymorphism: Implementing complex polymorphic structures in C#, C++ was initially challenging, as it required a deep understanding of how objects of different classes can be used interchangeably.
- Efficient Code Design: Ensuring code efficiency while maintaining readability and following best practices was a balancing act that posed a significant challenge.

I found the following topics particularly interesting:

- Design Patterns: Learning about various design patterns and their applications in solving common software design problems was fascinating.
- Unit Testing: The concept of unit testing in Pass Task 7 was intriguing, highlighting the importance of testing in software development.

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

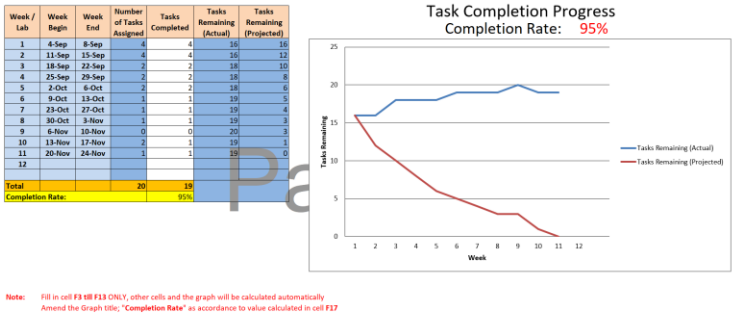
- Class Design and Inheritance: The creation of class hierarchies with proper inheritance and overriding methods, as demonstrated in Pass Task 12, is something confident about.
- Documentation Standards: Writing comprehensive XML documentation in C# is a skill I developed well, as evidenced in Pass Task 8.

I still need to work on the following areas:

- Understanding and Applying C++ Concepts: One of the more significant challenges I faced was fully grasping and applying the concepts of C++ programming. Despite my progress and achievements in other areas, such as C# and object-oriented design, I found the syntax and some advanced features of C++ to be particularly daunting.
- Reason for Difficulty: The primary reason for this struggle was the transition from C# to C++, which, despite both being object-oriented languages, have notable differences in syntax and memory management.

- Future Focus: To address this, I plan to dedicate more time to studying C++ in-depth, focusing on its unique features such as pointers, memory management. I believe that a stronger foundation in these areas will not only improve my proficiency in C++ but also enhance my overall programming skills and understanding of computer science principles.

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-Sep	8-Sep	4	4	16	16
2	11-Sep	15-Sep	4	4	16	12
3	18-Sep	22-Sep	2	2	18	10
4	25-Sep	29-Sep	2	2	18	8
5	2-Oct	6-Oct	2	2	18	6
6	9-Oct	13-Oct	1	1	19	5
7	23-Oct	27-Oct	1	1	19	4
8	30-Oct	3-Nov	1	1	19	3
9	6-Nov	10-Nov	0	0	20	3
10	13-Nov	17-Nov	2	1	19	1
11	20-Nov	24-Nov	1	1	19	0
12						
Total			20	19		
Completion Rate:				95%		

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

I was able to submit most of the Tasks, but there were still some issues with some of them. Therefore, I want to continue studying C# and C++ again during Semester Break.

This unit will help me in the future:

- Foundation for Advanced Programming: The principles and practices learned will be fundamental in my future studies and career in software development.
- Problem-Solving Skills: The ability to think abstractly and solve complex problems is valuable in any technical field.

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

- More Practice on Challenging Concepts: I would spend more time practising and experimenting with challenging concepts like polymorphism and design patterns.
- Seek Feedback Earlier: I would seek feedback more proactively to identify and correct misunderstandings earlier in the learning process.

Other…:

- Time Management: Balancing the workload with other commitments was challenging but taught me valuable time management skills.
- Continuous Learning: This unit reinforced the idea that learning is an ongoing process, especially in a field as dynamic as software development.

Concluding Statement

In summary, I believe my portfolio demonstrates that I deserve a distinction grade. Compiling tasks and projects within this portfolio showcases my understanding and application of object-oriented principles. It reflects my ability to engage with complex programming concepts and implement them effectively. From mastering the fundamentals in C# to tackling advanced topics such as design patterns and unit testing, my work evidences a comprehensive grasp of the subject matter. Moreover, the challenges I encountered and overcame, mainly in documentation and code optimisation, have significantly contributed to my growth.