COS20007

OBJECT ORIENTED PROGRAMMING

JORDAN BOEKEL

101625077

Learning Summary Report

# Self-Assessment Details

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

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

Self-Assessment Statement

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

Minimum Pass Checklist

|  |  |
| --- | --- |
|  | Included |
| All Credit Tasks are Complete on Doubtfire |  |

Minimum Credit Checklist (in addition to Pass Checklist)

|  |  |
| --- | --- |
|  | Included |
| Distinction tasks (other than Custom Program) are Complete |  |
| Custom program meets Distinction criteria & Interview booked |  |
| Design report has UML diagrams and screenshots of program |  |

Minimum Distinction Checklist (in addition to Credit Checklist)

|  |  |
| --- | --- |
|  | Included |
| HD Project included |  |
| Custom project meets HD requirements |  |

Minimum High Distinction Checklist (in addition to Distinction Checklist)

# 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: **JORDAN BOEKEL**

# Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS20007 Object Oriented Programming to a **Credit** level.

Change the grade and unit title. Then delete this box.

[Provide a justification for why you should receive this grade… Write this for the assessment panel – tell them why you should get this grade.

For Pass: you need to indicate how you have demonstrated all Unit Learning Outcomes to a minimal level.  
For Credit: you need to indicate how you have demonstrated all Unit Learning Outcomes to a good level.  
For Distinction: you need to indicate how you have been able to apply all of the Unit Learning Outcomes in achieving the distinction tasks.  
For High Distinction: you need to indicate how you have been able to extend beyond the material presented in the unit.

In this section, refer to the tasks you have completed. These will be attached by Doubtfire after this summary. Do not try to demonstrate the outcomes here, this is just a summary.

Think of this like a cover letter to a job application – here it is a cover letter to your grade application.]

Provide justification and delete everything in [ … ]. Then delete this box.

# Reflection

## The most important things I learnt:

How Object Oriented abstraction works. I have previously had a vague notion of programs abstracting away as much complexity as possible, but did not have a good idea of how to put that into practice. This class has done a great job of introducing me to these concepts and showing me how to use them. Probably the biggest thing for me was being able to properly comprehend and understand an object oriented program when I’m reading it; no more getting confused by the keyword “new” or interfaces.

I also greatly enjoyed seeing TDD (if at a basic level) in action. I’ve also heard about unit tests before, but never used them myself and had no idea how they worked. Being introduced to them was a good experience and I’m glad to have seen them.

I also enjoyed getting some exposure to different design patterns, and in particular the program design challenge. While I did not complete it, it definitely got some gears turning in my head about how to approach a problem and come up with solutions.

[ Think about topics covered, but also other general things you may have learnt. Think about what you have learnt in this subject, and reflect on what you think were key learning points, or incidents. Did you learn what you wanted/expected to learn? ]

## The things that helped me most were:

I greatly enjoyed the first few tasks, and in particular the shapedrawer program. It felt much more practical than the other problems, and I enjoyed completing the tasks related to it.

[ List and explain ]

## I found the following topics particularly challenging:

Interfaces. They seem like quite a roundabout concept at first. Upon consideration, I can see how they make sense, but it’s still a bit difficult to grasp.

I found program design very difficult, though maybe that’s because I was overthinking the problem. I always found myself pulled in multiple directions by trying to be DRY (Don’t repeat yourself) while implementing multiple classes. If I tried inheriting things, I very quickly ended up overriding most everything anyway and ending up with multiple similar yet different implementations. I found interfaces sort of useful with the problems I encountered, but it always felt limiting to not provide default implementations in them.

In addition, it was very easy to get stalled on an appropriate program design – I knew what I wanted to do, but had no idea how to do it well and in an extensible way, and I allowed this to block me for way too long. I think, in retrospect, I should have picked a simpler program to work on, instead of getting stuck the way I did.

## I found the following topics particularly interesting:

I enjoyed building the ShapeDrawer program, and I would have loved it if it was extended further than it was. I expect my custom program will use it as a basis for it’s own work. I also liked the filesaving component of that, since it really demonstrated how I can use programs outside of the limited context of a single run with no memory.

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

The concept of objects as a type. I would hope so, after 12 weeks of learning, but still, I think it’s a valuable concept to grasp. I’ve previously tried out some functional programming (with scheme) which had a similar concept (ie. Little difference between inbuilt language constructs like ints, strings etc and the constructs you can build yourself), but I hadn’t made the connection before in OO programs. Understanding what all those keywords mean in other languages is really enlightening.

[ List and explain – if none explain why, refer to your pieces for evidence to support your claims ]

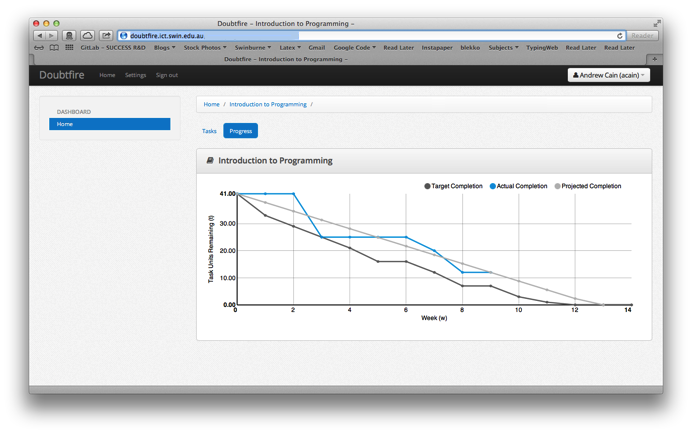
## I still need to work on the following areas:

Actual program design. The tasks provided were quite well detailed, but much of the real design work is left in the latter C/D/HD tasks. I think I just need more practice in that area building actual programs to get the hang of things. I feel that I’m fairly competent with low-level logic, but getting a good high-level design is *hard.*

[ List and explain – if none explain why, refer to your pieces ]

## My progress in this unit was …:

I waited until far too late in the semester to get started on problems – I procrastinated, and then ended up with my last couple weeks being dominated by work on my FYP project. I certainly should have been more diligent. I found the early tasks quite easy/trivial, and allowed this to justify my procrastination.



## This unit will help me in the future:

Knowing what and how OO works is fantastic, and I think these concepts will be very helpful to getting the ball rolling in the future. I do enjoy programming when I have a problem to work at (and don’t have 4 other subjects to worry about at the same time..).

[ How will the things you learnt relate to the rest of your studies, and career. What have you learnt that will be valuable for you in the future? ]

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

I would treat the deadlines for each unit submission as hard deadlines. I just don’t do well with soft deadlines, and delay too late.

## Other…:

[ Add any other reflections you think help you demonstrate your learning ]