COS20015

Object Oriented Programming

Learning Summary Report

S M Ragib Rezwan
103172423

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: **S M Ragib Rezwan**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieved all Unit Learning Outcomes for COS20015 Unit Title to a **High Distinction** level.

For instance, in this course I have understood several new and interesting concepts like abstraction (mapping real-world objects into code), encapsulation (storing data and functionality in a class to restrict access to certain component), inheritance ("is a kind relation" between parent and child classes) and polymorphism (accessing different types of objects using same interface). Although their use can basically be seen in varying degrees throughout all the tasks done in this course (as they are the four fundamental principles of OOP), detailed explanation about them can be found in tasks 7.1, 8.1 (alongside a simple concept map in 7.2), which in turn shows my depth of understanding regarding them.

Moreover, in this course, I have not only learnt to code well in C#, (as seen in all of my tasks) but also in other OOP languages (as seen in task 10.1 where I coded in C++ and in task 10.2 where I compared Python with C# in terms of coding following the OOP paradigm in each). Also, I have learnt how to use different libraries like Splashkit effectively (as seen in shape drawer tasks2.2, 3.1, 4.1, 5.2, 5.3), make full use of any given library's documentations and also on ways to add some extended methods to our codes(as seen in task 5.2). This in turn helped me gain the courage to code my custom program using an unknown library called Xna framework (which you can see in my task 6.4). Furthermore, I have also learnt ways to convert a code into a dll (dynamic link library) class and use a different library (ie WinForms) to implement it as a GUI (as seen in my task 9.1).

Alongside these, though out the course, I have also learned how to set up different tests using "Nunit Test" framework (as seen in tasks2.3, 3.2, 4.2, 5.1, 6.1, 7.3) and have also gained knowledge regarding which aspects of codes we should test, debug and the different ways to do them. Also I have found the best general purpose functions to use to test them (which can be seen by my constant use of those functions throughout those tasks). All of these helped my understand reasons behind occurrence of common issues (like null reference error, inconsistent accessibility, read-only property can't be assigned values, etc) and learn tips on ways to detect, avoid and correct them using testing and debugging steps.

Furthermore, I have also learned how to read and write UML (Unified Modeling Language) and sequence diagrams which determine the static and dynamic structure of the code respectively (as seen my tasks 6.1, 6.2, 7.3, 8.1, 10.3). Although these took a lot of time, I can now confidently say that I have the needed skills to immediately complete any code written by another person and understand it (as also modify it if needed) as long as I have the UML and sequence diagram related to that piece of code (to let me understand what the programmer had wanted that code to do).

Last but not the least, I have also obtained knowledge regarding how to code a program well in OOP manner. Although this has been taught to us subconsciously throughout the course (as seen in earlier shape drawer and swinadventure tasks where we learnt how to code inheritance, composition, polymorphism, etc without actually realizing that their true worth in OOP style coding), it has been mainly addressed in week 6, 8, 9 and 10's lectures where Charlotte had spoken about responsibility driven design, good OO design, GRASP and design patterns. Furthermore, you can also notice them when you look at my custom codes (task 6.4) where you can see the use of various design patterns (like Strategy pattern, singleton, etc) alongside maintaining good OOP design by following its principles.

All of these have helped me not only complete all tasks of P,C and D level of difficulty, but also further utilize my knowledge to complete a custom code and research project to an HD level.

In the custom code, I have made a top down shooter type of game where I have not only utilized the concepts I have learnt from this unit but also tackled unknown and new ones too (as seen in tasks 6.2 and 6.4). And the main reason I could tackle the unknown was because I had a good foundation built up via this course.

In the research project, I ran an experiment to compare and contrast between Factory Pattern with Composition against Inheritance to see which is actually superior. Unfortunately the code I had used for my method had actually backfired on me, making me unable to come up with a clear-cut solution (which you can see when reading the task 10.3). But even so it has given me new experiences and has helped me further improve my skills, techniques and understandings, mainly regarding the concepts related to good OO Design and design patterns, both of which I have acquired in this unit.

Thus with this portfolio, I believe I have not only achieved all Unit Learning Outcomes, but also have extended beyond the material presented, making me eligible for High Distinction.

Reflection

The most important things I learnt:

- I have already spoken about the principles I have learnt in the previous part. Thus I am not going to repeat those. Instead I will speak of the other important thing I have learnt here: "Any developer who doesn't have a full tab bar of google and stack overflow tabs isn't working".
  Previously I used to be under the misconception that one must never look up stuff on the internet as its cheating and instead one must focus on gathering knowledge from books and using the knowledge retained to resolve issues. Thus realising that googling was not only ok but also expected, helped me realise how much I had unnecessarily handicapped myself and thus made me regret over my past decisions

- Furthermore, I have also learnt that one must be able to subdue their own embarrassed feeling and ask questions when they are confused or uncertain about anything. That's because not everyone learns at the same rate, and here we are all at the stage of learning, and not understanding something is part of the process. Thus, even if one feels embarrassed, they should still keep asking questions regarding the concepts until they fully understand it, even if it means the class/tute or helpdesks may go a bit over time.

The things that helped me most were:

- Attending and listening to classes, especially tutorials. That's because there I could communicate with my peers and goof off a bit, giving me some mental satisfaction and encouragement. Furthermore my tutor was also quite nice and brilliant, explaining even the most complicated theories (they were complicated to me at that point in time) in simple laymen terms helping me understand and correct my misconceptions smoothly.
- Demonstrations in the lecture recordings showing off pieces of codes to link what has been taught in theory to a sample codes giving me ideas on where to use, how to use and also what limitations and restrictions are present in each way of coding

I found the following topics particularly challenging:

- In the very beginning I had found everything challenging due to the switch from Procedural to Object Oriented. But even so, the thing that was hardest for me drawing UML and Sequence diagrams. Although these look quite simple at this point in time, I can still remember myself struggling with them for a really long time. This had even ended up delaying me while coding swinadventure iterations and also in understanding the design patterns, making me mistakenly believe I was bad in everything.

I found the following topics particularly interesting:

- The most interesting I found was the fact that Object Oriented programming focuses a huge amount on writing codes that can be swapped out and reused wherever needed. It reminded of playing Lego with where blocks used to make a plane can be reassembled to make a car or a house. Although it might feel extremely basic to most people, this concept had blown my mind completely. That's because it had changed my idea of coding from "working like a machine to write convoluting stuff that somehow works" to "drawing a piece of art that can be appreciated and understood by all"
- Moreover, I also liked the emphasis on being lazy and writing codes that can used even later on in different types of programs. These made me realize that what I am doing now is not only directly helping me now but will also directly help me in the future. This explicit, direct, visible form of making "foundations for the future" is extremely inspiring and encouraging to me.

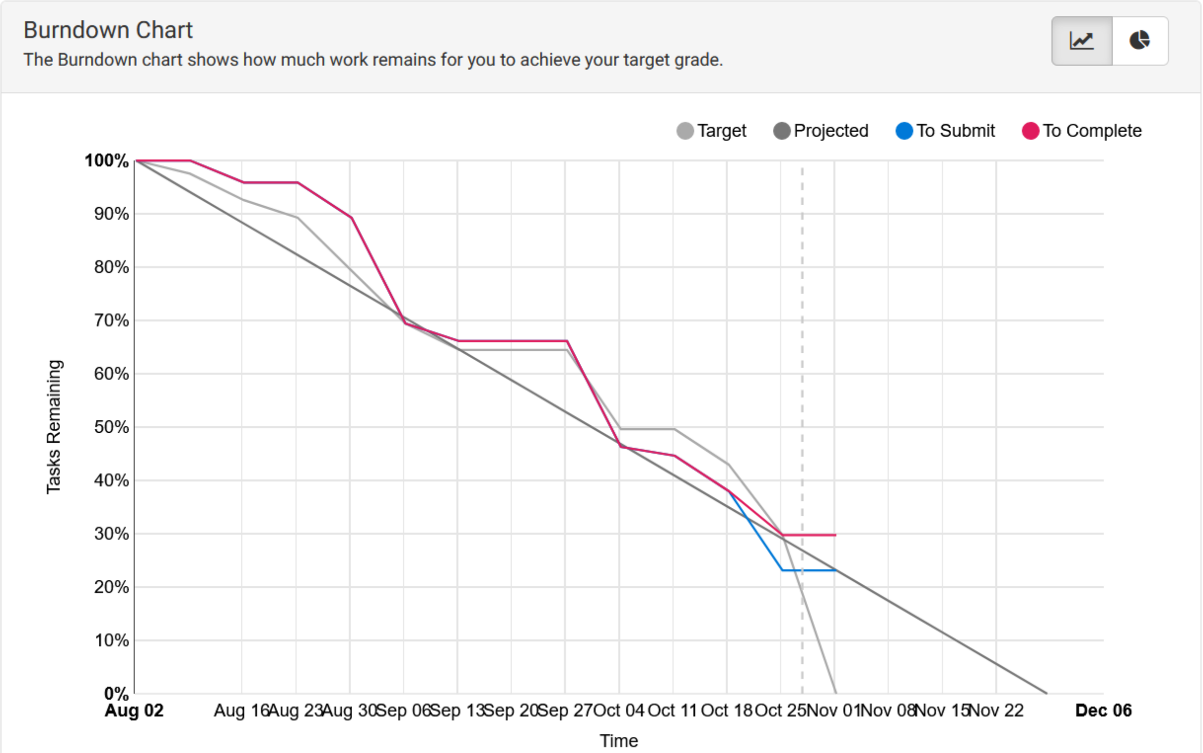
I feel I learnt these topics, concepts, and/or tools really well:

- Although I have learnt different concepts, including the principles, design patterns, delegating roles and responsibilities, etc, I believe I have learnt the concepts of inheritance and compositions the most. That's mainly because I have been using them constant throughout the year (as you can notice in my tasks) playing around with them, seeing what limitations they can have and what they can and can't do. This lead me to find some interesting and useful things about them (in terms of flexibility, hierarchy, grouping objects, etc) which you can notice in my Custom code and research project.

I still need to work on the following areas:

- There are a couple of things that come to mind when I think about things to improve on. But the thing that stands out the most is design patterns. Although I have understood and have been able to utilize some of them successfully (like singleton, factory, strategy, etc), I have not played around with them enough. Thus I am still not at that stage where I can create the perfect version of each design in terms of lines of code used and efficiency and type of output obtained. Instead I am at the stage of tinkering with them, trying to see what fits and not, drawing out each and every limitations they have (both mentioned and hidden) and ways they can be modified to better suit my current and future needs.

My progress in this unit was …:



Burndown Chart

The Burndown chart shows how much work remains for you to achieve your target grade.

- In the graph you can see that my completion rate had fluctuated, sometimes being steady, sometimes sharply moving down. Furthermore near the end you can see that it flatlined.

  The fluctuations in the beginning as:

  a) It took me time to get used to the new style of coding
  b) Tasks were being checked once a week
  c) And most importantly, due to countless resubmits (as I had a hard time adjusting to reading and understanding UML and sequence diagram and thus used to make silly mistakes or coding in a different way that give same output, but not following what had been asked)

  The flat line at the end is because now only LSR, Custom code and Research Project is yet to be submitted and marked green and thus most of those files are blank.

This unit will help me in the future:

- Well currently I am not sure which way I will go to be honest. I only have the desire to be in an industry related to computer science and mainly plan on focusing and developing my skills for becoming a cyber-security expert. So, I am not sure where I will use my coding knowledge in as deep extent as seen here. But I will probably keep on coding and making simple games as a hobby, following the coding style I learnt here, just as I do for drawing and playing instruments. So, honestly speaking, the most valuable thing I learn that will help me in future is probably the 3 philosophical ideals that this coding style follows: program classes should be lazy, antisocial, conformist.

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

- If I did it again, then I will probably try my best to quietly sneak into all the different tutorials and somehow get the permission to attend all of them and listen. That's mainly because I find it easier to learn when conversing with different people about related concepts and in a class/ tute environment, people are more tuned in to thinking about codes and design styles in depth.
- Furthermore I would try googling from the start instead of wasting time trying to "reinvent the wheel" for common problems. Doing that would have not only saved a lot of time for me, but would have also let me understand both my current problems and also find similar problems other people have faced (and the way they solved it). All of these would have helped me not only understand things faster but also instantly find out new and interesting errors and their solutions as well!
- Last but not the least, I will keep persuading my tutor to ensure my tasks are checked, especially in case of iteration type tasks (where we build up on previous codes) like for Shape Drawer and Swin adventure. Thus I wouldn't fall in the situation where I am coding for the one iteration part, only to get feedback that there is mistake in previous iteration and thus I have to scrap and rewrite the parts I wrote based on that iteration.