**CMP1902M Object Oriented Programming 2022/23**

**Assignment 1: Report**

Name: Connor Young

Student ID: 26500492

Git/Version Control Repo: https://github.com/readUTF/CMP1903M-1/

**Code Review**

1. **Who did you provide reviews for?** *(Name, Student ID)*

Alex Philips, 27200573

Lorna Foster 26093967

1. **Who provided reviews for you?** *(Name, Student ID)*

Alex Philips, 27200573

Lorna Foster 26093967

1. **Reflection on code review: What did you consider changing / changed after receiving the reviews?** *(~400 words)*

After receiving my review from Alex I wanted to address the missing documentation last as I knew this would change as I remove, modify and update the code base to address the rest of the issues. My first step was removing the redundant methods within Card. Methods such as ‘Card.getId()` were initially added to test that the program functioned correctly during development and should not have been left in to begin with.

Alex also pointed out that a majority of the important methods had too little to no validation. This type of validation is important and was an oversight when writing the code. I went through each of the key methods and addressed the following:

* For the public constructor within Card I added verification of the 2 inputted values, ‘suit’ and ‘value’, to ensure that fit within the ranges of 1-4 and 1-13 respectively. This can be found within Card.cs from lines 18 to 25
* For the Pack.ShuffleCardPack method I added validation so that if the ‘typeOfShuffle’ argument did not match any of the existing shuffle methods, and exception is thrown to let the accessor know that they have used an invalid shuffle. (Pack:69)
* Within for the Card.Deal method I throw an error if the pack was empty as there are no cards left to deal. (Pack:80)
* For Pack.DealCards the same type of validation for Card.Deal was needed including checking the deal amount was valid (Greater than 0). (Pack:94-97)

To address the documentation issues, I implemented the c# standard for generating documentation for methods. Majority of IDE’s can generate this documentation for any specific method, and covers all parameters, thrown exceptions, and leave space to create a summary that explains what the function does, its intended functionality as well as the expected side effects.

For example, with the ShuffleCardPack method, for the summary I specified the range of inputs (1-3), the type of shuffle to be expected from each input and that the list of cards is permutated globally as the variable is static. This gives any developer reviewing the code, or looking to implement it, an understanding of its side effects and expected use.

Finally, I addressed the validation/testing of methods. Alex pointed out that the tests were lacking and would become more readable, and easy to work with if broken down into categories. To address this is created tests for more specification functionality with the following categories: Dealing, Shuffling, Exceptions.

My next review from Lorna brought up valid points about Static declarations, method naming standards and leaving unused imports in majority of the classes. Although having unused imports does not directly affect the performance of the program it makes it clearer only having the necessary imports.

The Pack model also better suits a more specific object oriented approach that assigns each instance of Pack its own list of cards instead of each instance of pack sharing the same set of cards. This makes the current state of the pack clearer, allows for multiple active packs, and suits the object oriented nature of c# much better. This was addressed by simply removing the static keyword from all variables and methods and modifying the parts of code that accessed methods statically.

Lastly, I just reformatted each method to use Pascal case instead of Camel case.

**Include evidence of the reviews *(screenshots are OK)***

Text

Description automatically generatedText

Description automatically generated

**Reflection and Feedback**

1. **What was the most important thing you learned from this assessment?** *(< 200 words)*

The most important take away for me was to be a lot clearer and consistent with the comments / documentation I provide as it greatly increases the readable for people who are viewing your code for the first time without the need of a 1 to 1 explanation of the code base. Validation also goes hand in hand with better documentation as it can prevent issues that may occur from a lack of understanding cause issues down the line and forces the programmer to consider these exceptions and errors either before they occur or when they are thrown, instead of causing issues that need to be traced back.

1. **What was the most challenging aspect of this assessment and how did you approach it?** *(<200 words)*

*Eg: I started painting as a hobby when I was little. I didn't know I had any talent. I believe talent is just a pursued interest.*

1. **What would you particularly like to receive feedback on in this assessment?**

I would like feedback on the implicit readability of my code, rather than when guided with comments. When working in a team it is good to both have comments be written that explain the code but it is also important that the code itself is written in a standard way that is easy to modify and fix bugs within.

**Assignment 1: Checklist**

All of the elements in a section must be checked for it to be considered for that grade (this isn’t guaranteed though). All previous elements must also be complete for a grade to be considered.

Pass standard:

|  |  |
| --- | --- |
| Received two code reviews. | **✓** |
| Made two code reviews. | **✓** |
| Addressed **at least two** of the questions in each of your reviews. (*1. How is the code documentation, 2. How does the code handle errors, 3. How can the code be improved*) | **✓** |
| The code compiles and runs | **✓** |
| The classes provided by the base code are implemented | **✓** |
| The methods provided by the base code are implemented | **✓** |
| Object instantiation and method calls used in the code. | **✓** |
|  |  |

2:2 standard:

|  |  |
| --- | --- |
| Addressed the **three** questions in each of your reviews. | **✓** |
| There is a short description of the code review process. | **✓** |
| The code compiles and runs | **✓** |
| Some evidence of error handling | **✓** |
|  |  |
|  |  |
|  |  |
|  |  |

2:1 standard:

|  |  |
| --- | --- |
| The code compiles and runs | **✓** |
| No errors evident | **✓** |
| A Test class is used | **✓** |
| There is a description of the code review process | **✓** |
| **Additional** methods to those provided by the base code are implemented – point out with comments | **✓** |
|  |  |
|  |  |
|  |  |

First standard:

|  |  |
| --- | --- |
| Error handling, either by exception handling or other methods is complete and no errors are produced. | **✓** |
| Encapsulation an/or Abstraction are used in the code and identified. | **✓** |
| **Additional** classes to those provided by the base code are implemented – point out with comments |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |