**CMP1902M Object Oriented Programming 2022/23**

**Assignment 2: Report**

[*Expand the sections as necessary*]

Name: Harsimranjot singh gill

Student ID: 26872861

**Application:**

1. **Reflection on the OO features within your code.** (~400 words)

* Encapsulation: In my code every field that holds data are protected and have controlled access through properties, some of the more sensitive fields such as suits of the cards / operators also set value limit so that it won’t store unexpected value. Access to variables and methods are also restricted, some methods can only be called upon by the inheriting class.
* Abstraction: My code contains general classes that are used to store and manipulate more general information but are extended by other classes to be more specific.

* Inheritance: Some Classes inherits from more general classes such as card inherits from data class, pack inherits from group class to extend those classes and make them more specific for their tasks while also using their methods and fields. Furthermore, some classes also implement interfaces to extend their functionality such as pack which implements shuffle and deal interfaces to shuffle the deck of cards and deal them and classes implement interfaces to provide a template about the methods they have such as display class implements operator interface which has get operator method to get the label of the operator based on the integer value. Calculation class implements calculation interface which has a method to calculate the value of the two cards in any binary operation.
* Polymorphism: Dynamic: card class inherits from data class, it overrides the methods of data class according to its requirements, such as ‘add data’ method to insert data into its fields and ‘to string’ method to know to represent it in strings. Group and data class both overrides ‘to string’ method to have their own representation into strings.

Static: Operator overloading: Card class provides operator overloaded methods so that doing calculation with them is easy, it has four overload methods which provides implementations similar to integers doing their operations of {‘+’,’ -,’ \*,’ /’}.

Function Overloading: Display class provides overloaded method which accepts either three or five card objects to display a sum of mathematical equation for the end user and similarly Calculation class provides method to calculate either two cards or three by using the same method but by providing different signature.

1. **Reflection on your handling of error conditions in your code.** (~200 words)

As error occurs usually occur through user input I decided to put all the error handlers as well as guard clauses in the ‘tutorial’ class which holds the ‘main’ method. There are total three four inputs required from the user, these are verified and only then are supplied to the methods as arguments so that no unexpected arguments go into methods as they do not expect those values.

* Name: first the program asks user his name in a try block if user does not anything than it triggers the catch statement as name cannot be empty, so it repeats until it gets a non-empty name.
* Difficulty: program asks for game difficulty in either 1 or 2 nothing else otherwise catch and repeat until gets right.
* Answer: When program displays a mathematical problem it expects a response from the user mainly the answer to the problem, so the answer getting variable is also in the try and catch block, which will only go forward if the provided response from the user not a text or null but only an integer or real number.
* Exit: Lastly the program asks user if they want to exit the program and if the user enters ‘q’ or ‘Q’ it exits otherwise for any other input it continues.

1. **Reflection on your testing activities: What did you test, and how did you do it?** (~400 words)

I have tested all the methods that are responsible for the working of the application, within an new test class.

In my application all the classes are responsible for one tasks, and all of them has their own methods and in some cases constructors that does the work, so these are the methods that I have to test in different scenarios. So I have created a test class and assign it the ‘Main’ function temporarily from ‘tutorial’ class, so that I can test all the methods easily without involving the ‘Tutorial’ class which essentially holds the program together by taking user data and giving it to other classes.

* Initiating objects: As some of the classes does not have static methods, first I have initiated those classes by creating instances of them: making calculation object, display object whose methods are tested later and pack objects which has all the methods in the constructor, with both cases easy (1) and hard (2), which in turn tests all the methods of the pack class as its constructor deals with its method itself, so by doing that we can test all its methods
* Random numbers: There is a random object which will be later used to supply random number according to the specified so that we can truly test different numbers without actually inserting them into the methods.
* While loop: There is a while loop which is infinite so that I can test as much as I want, without restarting the program.
* In the while loop I have created five card objects: three for values and two for suits, these are then used to test the remaining functions. Which are calculate and its overloaded version which calculates the value of the card supplied, get Operator which is used to get a string label of a given operator based on the integer number, problem and its overloaded version to print the value of the cards in a meaningful manner. All of these methods are called one by one using the randomly generated card values and suits using random object and next method and their values are tested.

1. **Include evidence of the tests *(screenshots are OK)***

**Reflection and Feedback**

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

One of the most important thing that I learned was how to implement a good hierarchy structure using inheritance, to make my code more compact and useful for future use. As we were using ideas/code from the previous assessment I thought if I make my code in such a way that some or all features can be used for other general tasks or can be further specified for more specific tasks, than it will help me reuse my code as well as make my code look easy to follow for other. By doing so another thing I encountered was dynamic polymorphism which to me is very much related to good hierarchy structure as it makes those general classes more flexible as rather than leaving some functions un used after inheriting the class we can override them for more specific problems. Which in essence we are extending the general classes than just adding or not using some methods which can confuse others who might want to use that code or further extend it.

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

For me the most difficult task was how to deal with card objects as sometime they act as integers and sometime as operators for both calculating the value and displaying them on the screen for the user. To deal with them as integers I implemented operator overloaded methods in the ‘card’ class which provides a method to apply basic mathematical operations on only their values and returns a card with calculated values and suit = 0. This helps me deal with two cards as if they are integers with values equal to card.value. For operators I have implemented a ‘switch’ statement, based on the suite values of the selected card objects that will act as operators, to identify which is the operator. For convention the selected card objects are those in between other normal card objects used as values, like in normal mathematical equation. This solution is implemented in both calculation and display according to the needs but it essentially performs the same and helps me deal with card as integers and operators where necessary.

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

**Assignment 2 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:

|  |  |
| --- | --- |
| The code compiles and runs. |  |
| Card, Pack, classes are created. |  |
| Class definitions and object instantiation evident. |  |
| Method calls to methods in the same class as ‘Main’ |  |
|  |  |
|  |  |
|  |  |

2:2 standard:

|  |  |
| --- | --- |
| The rules of the tutorial as specified in the brief are implemented. |  |
| Application repeats or quits the game gracefully according to user choice. |  |
| Method calls from ‘Main’ to methods in other classes |  |
| Error handling is evident, some errors are captured, such as (but not limited to), card sums are problematic (don’t add up, are not random etc) |  |
| Class definitions show **encapsulation.** |  |
|  |  |
|  |  |
|  |  |

2:1 standard:

|  |  |
| --- | --- |
| Exception handling is used |  |
| Inheritance showing a class hierarchy |  |
| public/private access control in classes |  |
|  |  |
|  |  |
|  |  |
|  |  |

First standard:

|  |  |
| --- | --- |
| Interfaces are used |  |
| Static and/or Dynamic polymorphism (eg. method overriding) |  |
| Use of virtual/abstract methods |  |
| protected access control is used in class hierarchy |  |
|  |  |
|  |  |
|  |  |
|  |  |