**CMP1902M Object Oriented Programming 2023/24**

**Assignment 2: Report**

Name: Adam Peverill

Student ID: 27646423

Code repository URL:

Video URL:

**Application:**

**Reflection on the OO features within your code.**

My code is built of several classes all doing specialized tasks helping it run effectively. In this section, I will be identifying these areas and discussing their utility.

First, let’s talk about the two different game classes “Three or more” and “Sevens out. Using a base interface under the name “IGame”, both games contain; a public “Game” function which stores changes to a players points all under a while loop along with allowing separate classes from running the game, an internal “Gameplay” function that holds the game code as ruled by the supplied text file and the “Game data” function that updates the current wins of each player for the statistics class to display. Alongside this, both games inherit from the “Colourswitch” class which was created to clarify turn order whilst freeing up space within the code. Lastly, the dice class is introduced as a new object so that each game can have a pseudo-random number from 1 through 6 be picked each time the roll function is called.

Next, I will be discussing the “Game selection” class. Made up of two functions, “player selection” and “Game select”, this class will ask the user to input a number between 1 and 5 to gain access to either game, analysis class, testing class or to exit the program. This is contained within a while loop to allow someone to select different areas repeatedly until they were done with the program and use “exit”. The user input works through an integer saving the number results from a separate static function within the test class called “usable value” which uses try parse and a separate integer inputted beforehand to check if the value is allowed under the set circumstances. The same method is used to determine whether the user wants to play single player or multiplayer, returning a Boolean which effects aspect of the game code.

Lastly, we have the “Test” class. Built up of several static functions, this class tests out code actively whilst someone is playing a game by checking dice values and user responses for anything that could cause issues during the run time. Along with this is a game looper which tests a game and returns how long it takes for a user to gain victory to gauge how long an average match last. This can be used to tweak the goal requirements of each game or discover issues from suspicious results

**Reflection on your handling of error conditions in your code.**

Throughout my code, there are many sections which attempt to handle or prevent any errors from effecting the main code. Some of these methods are seen by simply breaking part of a loop to steer the code back into the correct direction, making use of try methods to catch any suspect code or by using debug assert to determine if there were any issues with logic.

I do believe there are some portions where the error handling could be more specific however. For example, the use of try catch often searched for the generic term “Exception” instead of searching for more specific error terms. This can affect the speed at which the code runs by forcing it to check multiple error instances, even ones that would likely never affect the section of code in the first place, instead of a simple specified district. This also prevents different actions to take place depending on the type of error is present as some aren’t as harmful as others. This could simply be fixed by determining the types of errors the code is most likely to encounter and code those instances.

**Reflection on your testing activities: What did you test, and how did you do it?**

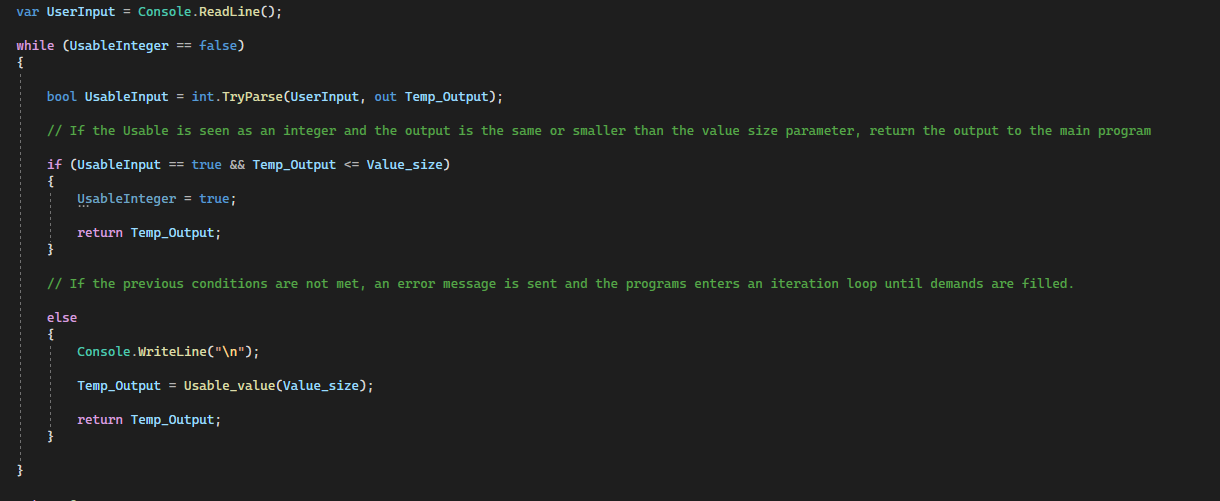
Within the code, I had made measures to test and prevent three explicit types of errors that had a likelihood of appearing within the code.

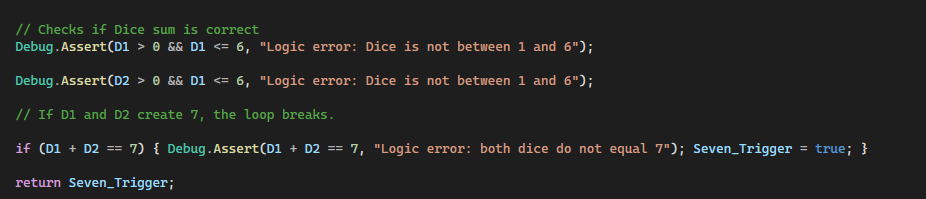
The first measure of testing was the use of try parse to convert string safely into integers to be used through the program. The reason for using integers in the beginning was due to their simplicity when asking for user input, not only is it quick to type in a number but it also makes it easier to code a response to that input too. The issue occurs in having the user input be verified as an integer due to its identity being string. This is where the Try parse comes in. By taking the variable from the user and inputting it into the method, we can get a bool stating whether the value passes the integer check. On top of this, we can output the user variable under the type which the method was searching for. All it took once that was coded in was to input a loop that determines if the user’s input was valid and only returning the value if it was. This prevents any unwanted code types from bleeding into the program and causing errors.

Secondly, there were multiple lines of code put in place to test for logic errors. Making use of the “Debug Assert” method, tests were made towards dice objects to check if the selected number and addition of separate variables leaded to correct results. These safe guards put in place were vital to the code running as any of these values being incorrect could lead to the program not working as fully intended.

Lastly, A loop function was introduced into the testing class to check the time it took for a player to win and give that data to the user. Using a similar bool method to how gameplay works in single player, test allows the bypass of most user required inputs. the data received from this function can help determine if there are any irregularities within the win time of players which could be due to errors within the program or help with deciding the point requirement for a player to win.

**evidence of the tests:**





**Reflection and Feedback**

**What was the most important thing you learned from this assessment?**

Through working through this assignment, I have learned the structural and separation of different elements, functions and methods within a codes structure. By having a project that required the application of several aspects of the course was an interesting challenge to overcome but I believe that what I have created is the best I can do at this current point in time. There are always more things that I will need to learn in the future such as how to apply methods such as “Cancel output read” but it is best not to overwhelm yourself with new features

**What was the most challenging aspect of this assessment and how did you approach it?**

I believe that the most challenging aspect of this assignment was the implementation of the two games themselves, especially three or more. Something that seems so understandable in terms of human comprehension is difficult to break down and relay to a program. For example, you may choose to reroll your dice if you have rolled two of the same for three or more. The issue arises trying to figure out how to let the computer understand that aspect of the game.

The way I overcame these challenges was to take a step back from the program and try to break it down piece by piece. You can’t get a solid answer to your question if you are trying to force the answer, all that will cause is frustration and burn out. There were many instances where I decided to write down my ideas on a piece of paper too so that I may visualise my main problem and the goal state that I wanted to achieve.

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

For this assignment, I had issues trying to understand how to construct a meaningful test class, I would like to gain more knowledge on how to properly apply these types of classes so that it could be more useful to a developer

I have also struggled with the over-use of if else statements to sort between my results. It feels as if there if a better way to represent these within the code.

Lastly, my interface seems lacklustre and feels more like an abstract class. Believe it would be useful to gain some insight on how to better utilize interfaces so that I may use them properly in future projects.

**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. |  |
| Die, Game and Testing classes are created. |  |
| Object instantiation, method calls evident. |  |
| Sevens Out game is created. |  |
| The Testing class is used. |  |
|  |  |
|  |  |

2:2 standard:

|  |  |
| --- | --- |
| The rules of the Sevens Out game, as specified, 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 erroneous input being made. |  |
| Class definitions show **encapsulation.** |  |
| The Testing class checks the dice sum is correct and that a total of 7 is detected. |  |
| A Statistics class is used |  |
|  |  |

2:1 standard:

|  |  |
| --- | --- |
| Sevens Out and Three Or More games are implemented. |  |
| **Inheritance** is implemented, showing a class hierarchy |  |
| public/private access control in classes |  |
| Generic collections (such as List<>) are used. |  |
| Exception handling is used |  |
| Testing class uses verification methods in code (such as debug.assert()) to check code. |  |
|  |  |

First standard:

|  |  |
| --- | --- |
| Interfaces and LINQ are used |  |
| Static and/or Dynamic **polymorphism** are evident |  |
| Use of virtual/abstract methods |  |
| protected access control is used in class hierarchy |  |
| The Testing class implements a way to record testing data (through a log file for example) |  |
|  |  |
|  |  |
|  |  |