# Bulls and Cows Assignment Reflection

Assignment 2: COMPSCI718 – S1 2022

## I. Reflection

During this assignment, I created a Bulls and Cow game application utilising the Java programming language. In the creation of this application, I utilised programming techniques including inheritance, polymorphism, UML Class diagrams, as well as a range of other techniques and learnt some key skills in the design and implementation of a Java program. One example of this, was how much my final design differed from my initial plan as demonstrated in the difference between my initial class diagram, and my final class diagram. I found the initial class diagram very useful to structure the start of my program, so I was able to create classes with empty methods, then begin implementing the functionality by then progressively filling in these methods, which made the size of the project significantly less intimidating. It allowed for a much more structured approach to the implementation of my program. While implementing these methods however, I came across a number of instances where I had found my initial design plan did not appear to be the most ideal way to implement my program. For example, I had planned to use custom made exceptions for the error handling the users input if there were too many or too few characters, or an otherwise invalid input, however in terms of programming efficiently found this easier to manage through conditional statements and control flow. Another example was my plan to implement a writer interface, however found it likely more appropriate to have these functions of the main class where the game is run, instead of having those as functions of the User class or Computer class. Another thing I learnt through the design and implementation of this application was the usefulness and practicality of utilising inheritance and polymorphism. For example, having each of my different difficulties inherit an abstract class (Computer), I was able to save a significant amount of code duplication in my BullsAndCows class, where without inheritance and polymorphism I would have likely had to do a lot of if/else statements checking the difficulty before making each guess, but instead I could call the parent class, and the function called would be worked out at runtime depending on the difficulty selected. In the implementation of my program, to ensure that it was working effectively, I continually ran the program after most changes so I could ensure it is still functional after each change, and reducing the need to go back through my code to find where the mistakes are. As part of this, I also focused on implementing a basic skeleton of the program and building on this, so at each stage the code would compile and I could run my own tests. I ensured I took in to account variations in input and how this might affect my code by entering invalid inputs, and trying to account for as many ways as possible that might cause the code to have an error. Most of this was done through error handling and control flow.

Particular modules I found useful for the implementation of this program included (although not limited to): module 5: Inheritance, for purposes as mentioned above, module 6: UML, for the design and planning of my project, module 8: I/O, for tasks involving file input and output including reading guesses from a file, and the save game option, and module 9: Collections for some of the more challenging logic required in the implementation of the game such as storing previous guesses, bulls and cows to either help with the medium and hard difficulties guessing algorithms, and saving the game to a file. Through the use of concepts learnt in these modules, I believe I was able to implement a good design for my program in the end. Particular areas I believe I did well in with my design include the creation of many different methods, to assist with code readability, and bug-fixing, commenting my code so it is obvious for myself, or a code reviewer to know what the intended function of each method is, naming my methods and variables appropriately to assist with code readability, and the use of an abstract class and inheritance for my Computer classes to reduce code duplication. Particular things I think I could have done to make a better design would have been to break my BullsAndCows class up a bit more, as it ended up being a much bigger class, with a lot of different functions, which can then effect code readability as the reader then has to scroll through a number of different functions to find one they are looking for. Another improvement I could have made in my design, was further shortening some of my methods, for example my saveGame() method in particular, and possibly the use of static methods for some methods where I am rewriting similar loops, like I have done in a method looking for repeating characters in multiple places, and checking for the amount of bulls and cows.

For my advanced configuration, I chose to add the ability to change the number of rounds a game, and the number of characters a code is allowed to be. This is done by altering the constants MAX_ROUNDS for the length of the game, and CODE_LENGTH for the length of the code. These are found as fields in the BullsAndCows class, so the user is easily able to adapt this, and in the comments it is specified that it can be altered for easy, but is not compatible with hard. These were both made possible by using these variables instead of the default value in methods dealing with the amount of rounds, and the length of the code (for example, error handling in the user class, instead of using i<4 as the condition in the for loop, using the CODE_LENGTH variable, so if this is changed, so are the error messages and allowed length of the users code). The same principles were applied for the MAX_ROUND constant, where instead of using 7 as a conditional to check if the game is over, I was able to use the MAX_ROUND constant.

Overall, I found the design and implementation of this program both challenging and highly beneficial for learning and practicing a significant amount of different concepts and techniques in Object-Oriented Programming.