Reflection

Wan-Ling Chang
COMPSCI-718
University of Auckland
Auckland, New Zealand
wcha305@aucklanduni.ac.nz

***Abstract*—The goal of assignment 2 is to create a simple game from scratch named Bulls and Cows. This report is to summarize the critical reflection on how I learned to accomplish this game as well as the design and implementation for my system.**

Keywords—Bulls and Cows, Wordle

I. Reflection on Assignment 2

The first task of this assignment is to analyse and design a class diagram based on what I learned from Module 6 and an useful lab called Farm manager 1984. For this game, I have to offer two different game options. One is the tradition Bulls and Cows along with three difficulty AI levels, and the other is its variation called Wordle. And there are two roles in each game including one player and one computer or an AI. Therefore, in my initial class diagram, there is a GameManager class working as the game console and flow controller. Also, I created a Game class that represent what different services can be provided in this game. The game player and compute which can be an AI sometimes can be recognised as different kinds of roles that have common features. Here I used inheritance concept from Module 5 to present the relationship between Role, Player and Computer classes. Role class is the parent that have common fields and methods. Player and Computer classes are the child classes that inherit all fields and methods from Role class. In this design, I planned to create all services that computer or AIs can do in the same class. In addition, I built three enumerations which can be sourced from Module 9 to predefine the commonly used constants, such as GameMode, RoleType, and AILevel.

However, during feedback session, the teaching staff Melissa, provided two productive feedbacks on my initial design. The first one is that if it is necessary to create my own packages: game and player. After a short discussion, we reached an agreement if this design wouldn't complicate my coding job too much, I could keep it. And then I decided to stick with my original design to group all related classes in the same package for better maintainable code. The second feedback is important and helpful while developing the hardest part of the game. The tutor recommended that I should create three different AI classes to fulfil all tasks instead of using only Computer class. Also, she suggested that I can take EasyAI as a general computer to perform the services that the single-player Bulls and Cows and Wordle can provide. At first, I haven't made up my mind to take this advice or not. I tried to build up the first prototype with my initial design. But later I realized what the problem was. I had to write lots of if-else statements to check the kind of computer before invoking the corresponding methods. It made my code less readable and badly structured. Therefore, I made a big change of my design based on the tutor's second feedback. I used a two-tier inheritance hierarchy instead. The first tier of inheritance is same as the old design. But Computer class becomes another parent class of the second tier that present common fields and methods that computer and AIs can do. It is also an abstract class which defines the default signature of common methods. EasyAI, MediumAI, and HardAI classes are the child classes that provide the implementation of guessPlayerCode method as it extends the Computer class. By doing so, the rest of my jobs became much easier and time efficient.

Throughout the whole journey of assignment 2, I did experience lots of ups and downs. The best part is that I put what I have learned into practice as much as I could. Except for those basic skills from Module 1 to 4, inheritance from Module 5 and how to make a class diagram from module 6, I applied the following concepts to this practice as well. First, about exceptions from Module 7, I caught checked and unchecked exceptions with try-catch clause in Game, GameManager, and Computer classes to stop any immediate crashes. Besides, I also developed my own exception class named WordleFileNotFoundException to define a checked exception. Thus, my system can throw and then catch the exception to recover from it. Next is regarding the use of java.io. * Package., I used PrintWriter to save the game result to a text file in Game class and Scanner to read a text file in Computer class. Meanwhile, I used try-with resources with its AutoCloseable feature to catch I/O exception. After that is generics. It can be used in different ways in Java. In this assignment, I used List collection to store the result of each guess and iterated through it with for-loops to print out messages on a text file. I also applied stream API and Lambda Expression to filer candidates in Line 37 of MediumAI class. Finally, towards how to reuse the existing method. I extracted the common statements to a newly created method. Thus, other methods can invoke this method if needed. The common methods I created can be found in Game, Computer and Game classes. However, there are still two issues that I haven't find a better way to solve it. One is about the game flow control mechanism. In my first sequence diagram, I totally left the quit option of the game behind. There was no way to quit a game earlier in my first prototype. I tried to write a better structured statements in the start method of GameManager class at the final stage of this assignment but in vain. Next time I will give it a try with parsing command mechanism which presented in Farm Manager 1984. The other issue is about how to ensure that my game can work as my expectation. I was struggled with this a lot due to an incomplete testing plan. I only did limit number of system tests as well as few unit tests by testing statements and hardcode. This is far away from a regular test. So, it is hard to guarantee about the quality and the outcome of my system. Hope that I can learn more about it later from this course and have another try in my final project.

No matter the good or bad part I have had in Assignment 2, they both brought new insights of programming to me. The best lesson for me from this assignment is that never be afraid of unknown bugs or issues. What you can do is to take your pen and do some practice. To find someone who can play a real game with you or write a pseudocode of your ideas is a better way to find clues that can help sort out your problems. The more you practice, the better you get.