Task 8: Reflection

I learned a lot by doing this assignment, a lot of which has been learning the context and application of module content in greater detail. While the labs and assignment 1 offered skeleton code, getting started on an empty page was really difficult. From this experience, I learned that you have to start SOMEWHERE, and sometimes you might not write code in the order you think you will. For example, when I first started this project I looked at each task separately, thinking I could complete one task and then move on to the next. This was not the best approach! I also learned just how important and helpful having a thorough UML Class Diagram is, and how important it is to plan your approach before starting any programming.

I found this assignment very challenging, however I think it went well that I was able to implement most of the tasks. Unfortunately I was not able to implement Task Seven, which means overall the assignment didn't go great. My approach to this project was to focus on Tasks 2 - 6 to develop a working game before implementing the Task 8 Wordle modifications. While I think this was a good approach, I struggled with modifying code that was already "complete" for separate functions, and many of my attempts to implement Task 8 resulted in breaking my working game from Tasks 2 - 6.

I found that most of the lectures and labs were relevant for this assignment! The Farm Manager lab was particularly helpful, as it had a lot of similar principles to what was needed for this assignment. For example, the task on creating new animals was really helpful for learning about creating new classes, the benefits of subclasses, and keeping code organised. Lab 6 was also very helpful for better understanding class diagrams and thinking about programming in terms of the sequencing of events. It also helped me understand the user input interactions from the code side as well as the user side.

Which part(s) of your design (final class diagram) and/or implementation is good?
If you do not think your design / implementation is good, explain what and how you could improve.
Please also specify the class(es), method(s) and/or line number(s)

I think my use of inheritance and polymorphism in the Computer class and its subclasses were a good design choice. A good piece of implementation was deviating from my original design to introduce a separate class for the GameFlow instead of including all the GameFlow code in the Game class. In my first draft UML Class Diagram and in my early implementation, this code was all in the Game class.

One improvement would be to include error handling during the user input section of the GameSession class where the user enters guesses. I do not have an "invalid guess" prompt, instead it just prompts the user to Enter your guess again (Line 57).

When it comes to my Hard difficulty class, I don't know if this code is good or not. I struggled with this a lot, and the "possible strategy" suggested was very helpful otherwise I was completely stuck. However, I onl ever achieved a "Draw" result, and so don't know if the Hard class is a success (I expected it to beat me more).

If your final implementation is different to the initial design, explain why you have made such changes.

Some of the change is based on my feedback session, but even without that I definitely would have needed to make changes. My first UML Class Diagram was quite limiting with so few classes, and not making use of the different natures of relationships we can have between classes.

As mentioned earlier, I originally had all the GameSession content in the Game class. This was not working very well and was very messy before separating the content into a different class to handle the actual gameplay.

Also, in my original UML I didn't have subclasses. Having the Easy/Medium/Hard subclasses proved to be a much better way of proceeding, so I benefit a lot from the feedback session.

How did you ensure your program worked as expected?
I tested it a lot, and paid a lot of attention to the (MANY) error codes I encountered. I also paid attention to the many prompts that IntelliJ offers while writing code.