Integrating Required Extension

New Dragon Card 1 - The “Reverse” card

For my contribution to Sprint 4, I worked on adding in the first required extension - the new chit card (New Dragon Card 1). This card requires that when a player draws this card, they are placed into the first starting cave on the board *behind them* which is unoccupied. This does not end their turn, and if they are already in a cave, they continue their turn as normal, without moving.

Generally, the addition of this new card was not overly complex due to the way that our system was configured. However, I did identify an opportunity to refactor our source code to make the new functionality more OOP-oriented and extensible. To do this, I refactored our original ChitCard class into an abstract class with two subclasses, the StandardCard (with all the normal behaviour of the original card), and the ReverseCard, which represents the new extended functionality.

To take this an additional step further, I refactored the system to follow a Command pattern when utilising the cards, by creating an invoker to manage a card with a series of action classes that handle invocations by the game that rely on the cards. This was the lengthiest portion of the refactor but, once integrated, made extending the functionality of the cards very straightforward.

Because I adjusted the system to use an invoker which references an existing chit card, with identical method signatures, I could almost leave the existing code exactly as is, with the exception of instantiating the invoker classes and retargeting the code towards this, instead of directly onto a chit card. The original code can still call the same methods on the same variables and achieve the same functionality, once the original object was substituted from a ChitCard to a ChitCardInvoker, which ultimately highlights the benefit of using the Command pattern, as it let me extend the specific functionality of the subclasses very quickly and efficiently with minimal refactoring to the source code once setup.

If I were to start again in Sprint 3, I think the biggest change I would have made is making use of the Command pattern at an earlier time, as I can now see how its usage makes refactoring substantially more streamlined and less prone to errors, as well as making the extension of existing classes much more straightforward and easy. Part of this process would mean that I need to rethink my approach from a holistic standpoint, looking at my analysis and diagram development as well as my general programming approach, so that I can visualise the opportunity to use design patterns from the outset, rather than retroactively through refactoring.