**ECM2414 Software Development – Continuous Assessment**

**Cover Page**

|  |  |  |
| --- | --- | --- |
| Student ID | 710056122 | 710030439 |
| Weight | 50 | 50 |

**Development Lo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Time | Duration (H) | 710056122 | 710030439 |
| 29/10/22 | 14:00 | 1 | GitHub setup | GitHub setup |
| 01/11/22 | 13:00 | 2 | Observer | Driver |
| 03/11/22 | 10:00 | 2 | Driver | Observer |
| 04/11/22 | 13:00 | 2 | Observer | Driver |
| 05/11/22 | 13:00 | 4.5 | Driver/Observer | Observer/Driver |
| 06/11/22 | 12:00 | 2.5 | Driver/Observer | Observer/Driver |
| 07/11/22 | 14:00 | 2 | Driver/Observer | Observer/Driver |
| 08/11/22 | 11:00 | 2.5 | Driver/Observer | Observer/Driver |
| 09/11/22 | 13:00 | 2 | Driver/Observer | Observer/Driver |

**Design Choices with Respect to our Production Code**

One design choice we had to consider early on was how to go about implementing/organising the output files. Our original thought was just to write all the output files generated from a game to the current directory of the project, but what if we wanted to play another game after? The game would try to create the necessary output files but find some/all of them already exist which would likely cause problems. Our next thought was to simply overwrite the files so that consecutive games could be played without any unexpected errors occurring, but what if a ten-player game was played and then a five-player game was played, you’d have the output files from the five-player game but also the output files for players six through ten since these won’t have been overwritten so you would have all of the second game’s output files and half of the first game’s files. To fix this our next approach was to put all the output files into a folder dedicated to output files so that when a new game is played, that folder can be overwritten and no fragments from previous games remain, however, we decided being able to view output files from all games played, and not just the most recent one, would be a good idea. This brings us to our current and final choice for storing the output files where whenever a game is played, a new folder is created to store the output files allowing you to store (within reason) as many game output files as you want. Initially, we decided we would follow a naming convention of gameN where N is the Nth game played, but a count of how many games had already been played would need to be stored somewhere to avoid trying to create a folder that already exists. This problem led us to a rather unique solution where the folder name is the current Unix time, derived from java.util.Date(), and in the format of “yyyy.MM.dd.HH.mm.ss”. This solution allows us to not only store the output files from multiple games but allows us and any users to easily distinguish each game’s output folder as they can see the second each game took place.

Another key design choice we had to consider was how to go about making sure that no player holds onto a card of a non-preferred denomination indefinitely, which ruled out approaches like discarding the first card with a non-preferred denomination as that card could keep getting replaced with another non-preferred card denomination and any other cards in the hand could stay their indefinitely. Our first idea was to go through the player’s hand and from the cards that can be discarded (any card with a non-preferred denomination), randomly choose a card and discard that one. However, this approach had two main flaws: a card could theoretically (although highly unlikely) never get picked to be discarded and hence stay in the hand indefinitely, and many randomly generated numbers are merely pseudorandom, and we would therefore not be truly picking a random card. Both problems have a solution namely in the form of a queue. When a player gets their initial hand, we check each card and if it’s a non-preferred card, we add it to a queue called discardables. When it comes time to discard a card from the player’s hand, we simply pop the top card from the queue, which is the card we discard. Every time a player receives a card, we check if it can be discarded and add it to the queue. Due to the premise of a queue being First-In-First-Out, no card can stay in a player’s hand indefinitely since we are working our way through a queue instead of randomly choosing a card to discard.