ECM2414 Card Game Report

By 710048828 and 710033318

Final mark allocation: 50/50

Development log:

Date:	Duration (hours)	Driver:	Observer:	Worked on:
01.11.22	0.2	N/A	N/A	● Talked over whatsapp, deciding to create a UML design for the project
04.11.22	0.2	N/A	N/A	● Looked over both the UML designs and chose what we liked from each, and which one to go forward with
05.11.22	1	710048828	710033318	● Created all the classes from the UML design and the basic class methods
08.11.22	2	710033318	710048828	● Testing basic methods created last time ● Swapped driver half way through because of JUnit
09.11.22	1	710048828	710033318	● Creating more advanced methods eg. Opening pack file, creating players, creating card decks
13.11.22	1	710033318	710048828	● Creating the player run method, and all the log methods

Design choice:

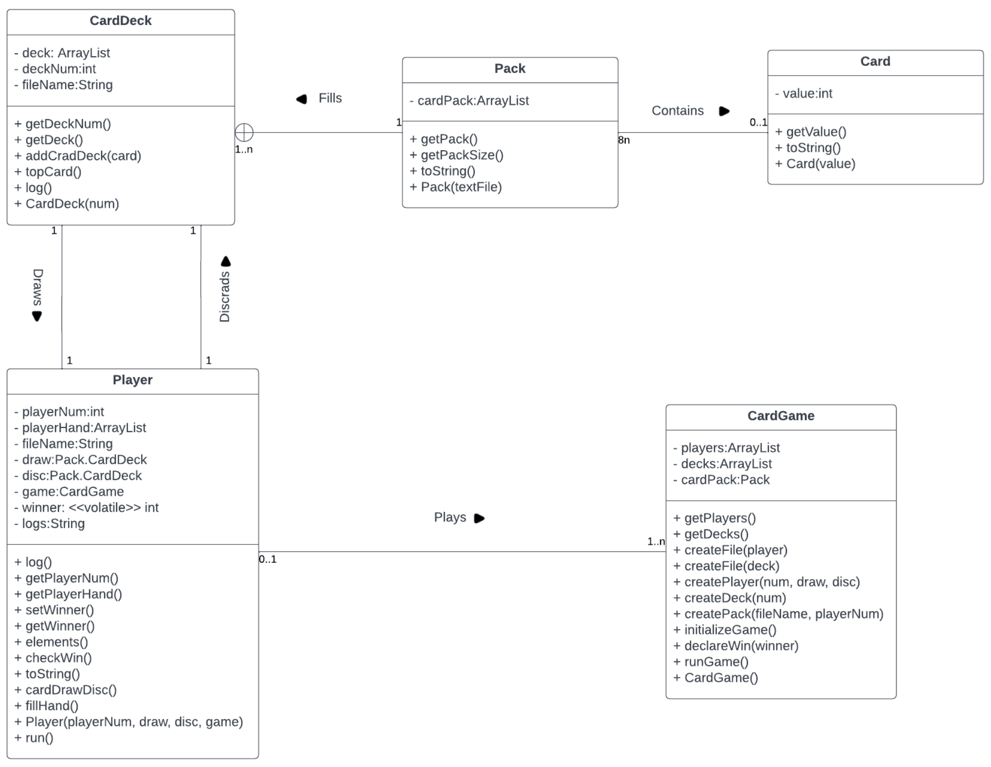

Figure 1: *UML design for the Card Game*

The specification required us to make a thread safe card game, containing at least a Card class, Player class, and an executable Card Game class. Firstly, to make sure that the CardGame class was executable, the public static method was written inside the CardGame class. Reading through the assessment specs to make sure the game runs properly we decided to add 2 more classes. First one being a Pack class, and a static nested CardDeck class within the Pack class. This can be seen in FIgure 1. At the start we thought of making the Pack and CardDeck 2 separate classes. However, since the cards from the Pack are used to fill the Card Decks, the 2 classes are related. To ensure the code is more readable and more maintainable we decided to make the Card Deck an inner nested class. Furthermore, the Card Deck is a static class. The Pack class is the one that fills the Card Deck. In other words, the Card Deck doesn't need to access and invoke methods on the instance of the Pack class.

To ensure that the cardinalities of the players, cards, and card decks conform to the game rules, from Figure 1 within the Card Game class we have a method called initializeGame(). When invoked, the method will ask the user to input the number of players participating in the game after which it will ask for the file containing the values of the cards for a pack. It will not create a pack until a file with the correct amount of values is loaded ($8 \times The\ number\ of\ Players$). After the method successfully creates the pack it will go through a for loop and create the players for the game and an equal amount of Card Decks.

One thing we also had to guarantee was that the players and the card decks formed a ring topology. For this to happen, as could be observed in Figure 1 under player class, we decided to include the Card Deck the player would discard to and draw from as 2 arguments in the Player object constructor. This made sure that the players would only draw and discard to specific card decks. However, this isn't the end of it. We couldn't just say in our code player[i], i being the player number, draw from deck[i] and discard to deck[i+1], because then player[n] would have to discard to deck[n+1] which does not exist. To avoid this in the initializeGame() method we create the card decks first, place them into an ArrayList and then when creating a player we assign them decks to draw and discard to. Within a code there is an if statement that checks for the last player to assign him to discard to deck 1.

We decided to use ArrayList as our main data structure because we felt it was the most effective and easiest to implement within the game. The players and card decks are stored in an ArrayLIst within the game, and the cards in the player hand are stored in an ArrayList within the player. There are 2 main reasons we decided to go with ArrayLists. First one being direct access to elements. There are no search methods within the game and most of the methods used in the game only require to use the first or last element within the list. Hence, there is no performance loss of using ArrayList since we will never need to search for anything. Second reason, the ArrayList will not contain null values. If instead of an ArrayList we used an Array there could have been the danger of a thread drawing a null value into its hand.

In the initializeGame() method, the players are given 4 cards in a round robin fashion, and then each card deck is given 4 cards from the pack also in a round robin fashion. To check if any player has been given a winning hand at the beginning of the game we use 3 methods. From Figure 1 in the player class the checkWin(), declareWin() and the runGame() in the card Game class. Once the runGame() is invoked the system checks every players hand with the checkWin() method, if any player has a winning hand checkWin() will set the volatile variable winner to the player number and evoke a method declareWin() which will notify every player who has won and then log all their actions. It was very important to make the winner a volatile variable. If not there is a risk of the threads not noticing that the variable has changed and would result in it continuing to run when it shouldn't.

If none of the players have a winning hand at the start the system will go through every player and invoke the run() method. Once the run() method has been evoked every player will draw and discard cards and then check if they have a winning hand with the checkWin() method. Since multiple threads could access the same Card Deck and cause race condition, the player cardDrawDisc() method is synchronised. In addition, at the beginning of the method it checks that the draw deck contains exactly 4 cards otherwise it returns from the method. Furthermore, the draw and discard method had to be 1 atomic action. That is why we made it 1 method. At first the player will check his current cards in hand to see if any card is of the same value as his player number. The other cards will be added to an ArrayList which is then shuffled, to ensure a random card being discarded, and the first card will be removed from the hand. The discarded card is then added to the discard Card Deck after which the player picks 1 card from his draw Card Deck.