1. Task Specification

You will develop, in Java, a ***multi-threaded*** card playing simulation. Within your design you will need to implement (at least) a thread-safe Card class and a thread-safe Player class (depending upon your design, you may also implement additional classes, for instance a CardDeck class). You will also develop an executable CardGame class.

The game has ***n*** players, each numbered 1 to ***n*** (which for clarity in the illustration below are named player1, player2, …, player***n***), with ***n*** being a positive integer, and ***n*** decks of cards, again, each numbered 1 to ***n*** (which for clarity in the illustration below are named deck1, deck2, … , deck***n***). Each player will hold a hand of 4 cards. Both these hands and the decks will be drawn from a **pack** which contains 8***n*** cards. Each card has a face value (denomination) of a non-negative integer1.

The decks and players will form a ring topology (see illustration in the figure below for the case where ***n*** = 4). At the start of the game, each player will be distributed four cards in a round-robin fashion, from the top of the pack, starting by giving one card to player1, then one card to player2, etc. After the hands have been distributed, the decks will then be filled from the remaining cards in the pack, again in a round-robin fashion.

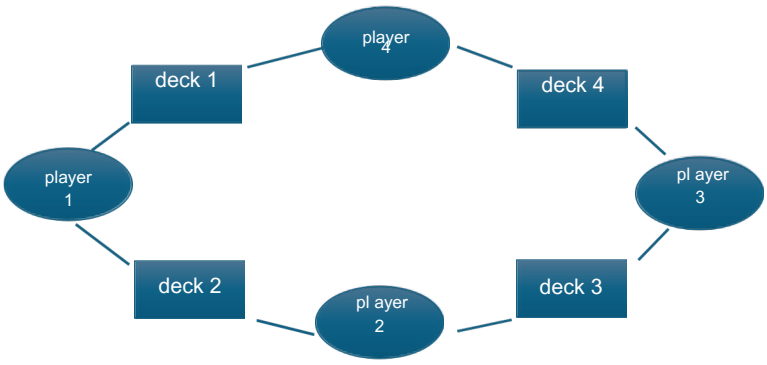

Figure 1: Topological relationship of the game players and card decks, in the situation where ***n*** = 4.

To win the game, a player needs four cards of the same value in their hand. If a player is given four cards which are all the same value at the start of the game, they should immediately declare this (by their thread printing “Player ***i*** wins”, where ***i*** should be replaced with the player index), that player thread should then notify the other threads, and exit2.

If the game is not won immediately, then the game progresses as follows: each player picks a card from the top of the deck to their left and discards one to the bottom of the

---

1 It is legal for the face value of a card exceeding *n*.

2 There is a chance that, two or more players are given four cards with the same value at the start of the game. You don’t need to handle this situation in your development.

deck to their right3. This process continues until the first player declares that they have four cards of the same value, at which point the game ends4.

1.1 Game playing strategy

If a player does not start with a winning hand, they will implement a simple game strategy, as specified below (note, the strategy is ***not*** optimal).

Each player will prefer certain card denominations, which reflect their index value, e.g., player1 will prefer 1s, player2 will prefer 2s, etc. After drawing a card from their left, a player will discard one of their cards to the deck on their right, (e.g. player1 will draw from deck1 and discard to deck2). The card they discard must display a value which is **not** of their preferred denomination. Additionally, a player **must not** hold onto a non-preferred denomination card indefinitely, so you must implement your Player class to reflect this restriction (otherwise the game may stagnate).

1.2 Solution development You will need to implement an executable class called CardGame, whose main method
requests via the command line (terminal window) or JOptionPane dialog box, the number of players in the game (i.e. '***n***'), and on receiving this, the location of a valid input pack. The screenshot below shows how this works (note, I put the pack file in the same directory of CardGame, so there is no path needed in the example shown below.)

```
Yuleis-Air:example_answer_CA1 yw433$ java CardGame
Please enter the number of players:
4
Please enter location of pack to load:
four.txt
```

A valid input **pack** is a plain text file, where each row contains a single non-negative integer value, and has 8***n*** rows. The screenshot below shows a partial pack

---

[3] By multi-threading, players should NOT play the game sequentially, i.e., NOT in a way that, when one player finishes actions another player starts.

[4] By multi-threading, there is a chance that, two or more players have their four cards of the same value at the same time. You don't need to handle this situation in your development.

After reading the input pack, the CardGame class should distribute the hands to the players, fill the decks and start the required threads for the players. If the pack is **invalid**, the program should inform the user of this, and request a valid pack file5.

If a player does not start with a winning hand, as a player processes their hand, each of its actions should be printed to an output file which is named after that particular player (i.e. the output file for the first player should be named player1_output.txt)6. In game actions should be printed to the file in a similar form to the following example:

player 1 draws a 4 from deck 1

player 1 discards a 3 to deck 2

player 1 current hand is 1 1 2 4

Additionally, at the start of the game the first line of the file should give the hand dealt, e.g.

player 1 initial hand 1 1 2 3

and at the end of the game, the last lines of the file should read either:

player 1 wins player 1

exits player 1 final hand:

1 1 1 1

if for instance player 1 wins, or

player 3 has informed player 1 that player 3 has won

player 1 exits

player 1 hand: 2 2 3 5

in the case where player 3 has won (and the values displayed match the hands and decks concerned).
There should also be a message printed to the terminal window (as is the case when a player wins immediately), i.e. if the 4th player wins, then

player 4 wins

should be printed on the screen.

There should only be one player declaring it has won for any single game7. If the game is won immediately (a winning hand is initially dealt), the output files should still be written for

---

[5] The game should not start until there are valid inputs for the 'number of players' and the 'pack' file.

[6] Each player should have its own output file, NOT a combined output file for all players.

[7] By multi-threading, there is a chance that, two or more players declare they have won the game simultaneously. You don't need to handle this situation in your development. You can simply give it another run.

all players (although each file will only contain four lines). In addition to the player output files, there should also be ***n*** deck output file written at the end of the game (named, e.g. deck1_output.txt), which should contain a single line of text detailing the contents of the deck at the end of the game, e.g.

deck2 contents: 1 3 3 7

The combination of a card draw, and a discard should be treated as a single atomic action. Therefore, at the end of the game every player should hold four cards. The program developed should follow the object-oriented paradigm.