**Design Choices**

* PlayerThread class that implements Runnable, with each player running their own instructions for playing the game in parallel. This was done so that the program runs more efficiently and to simulate how the game would be played in real life, with each player thinking and acting independently. Each player stores a list of Card objects to represent the player's hand, and it has methods to draw cards from the CardDeck object that represents the deck to the players left, and similarly to discard cards to the deck on the right. The references to these decks are given to the player when it is instantiated.
* Card class with each card containing an attribute to store its numerical value. This was done instead of just using integers so that multiple cards with the same value are still unique objects.
* CardDeck class representing a deck with four cards which are changed throughout the course of the game. Each deck stores its cards in a queue of Card objects because cards are always added to a deck at the bottom and always taken from the top, giving it a first-in-first-out rule of operations.
* CardGame class where the game is run, containing the main() method. The game creates a list of PlayerThread objects and CardDeck objects that each have a specified length, taken from a user input. These PlayerThread threads are run so that each player plays the game in parallel, at which point this main() method stops performing any computation until a player has won. When the game has finished this method will output which player has won the game and the program will stop.

**Testing Choices**

Test.