Programming Challenge: Card Game

Reviewer Information

Please make sure to check the following:

- Program runs and gives output similar to the example at the bottom of the instructions
- Submission includes unit tests which run green and cover the cases describe below (bonus points if more cases are covered)
- Take a look at the shuffle implementation:
    - Implementing a Fisher-Yates or using built-in (e.g. Collection.shuffle()) are both acceptable solutions
    - Implementing an alternative shuffle should be properly documented otherwise deduct points
- Shuffle should be tested:
    - Overwrite Math.random and to guarantee a certain order after shuffle is preferred
    - Using shuffle and comparing to the order of the deck before shuffling is acceptable
- Check how a draw situation is handled. Make sure it can handle multiple draws in a row.
- Check how length of the deck is maintained. It should include draw pile, discard pile and the played card (so both players length of deck should always add to 40). Often the played card is not considered (but this should only be a minor deduction).

Also evaluate naming and structuring. It should not be over-engineered but also consider how it could be adjusted to changing rules e.g.:

- Cards now have suits (Clubs, Spades, Hearts, Diamonds)
- The game has more than 2 players
- Deck size is changed

Introduction

You are building a card game for 2 players. Use any programming language you feel comfortable with. Please include the source code, as well as instructions on how to compile and run the game and tests in your submission. Compilation should be possible with the command line and common command line tool. Dependencies should be managed by a tool like maven or npm.

The rules of the game are listed below and split into tasks. For each tasks, please implement the indicated unit tests. If you are struggling to complete the challenge, it is okay to submit a solution that does not solve all tasks. The order of tasks is just a suggestion, feel free to implement in whatever order you see fit.

Please do not spend more than 4 hours on this challenge.

Task 1: Create a shuffled deck of cards

Each card shows a number from 1 to 10. Each number is in the deck four times for a total of 40 cards. At the beginning of the game, the deck is shuffled (*Hint: Look up* `Fisher-Yates Shuffle Algorithm`) to make sure it is in a random order.

Each player receives a stack of 20 cards from the shuffled deck as their draw pile. The draw pile is kept face-down in front of the player. Each player also keeps a discard pile (see "Task 3" for more).

Tests:

1. A new deck should contain 40 cards
2. A shuffle function should shuffle a deck

Task 2: Draw cards

Each turn, both players draw the top card. If there are no more cards in the draw pile, shuffle the discard pile and use those cards as the new draw pile. Once a player has no cards in either their draw or discard pile, that player loses.

Test: If a player with an empty draw pile tries to draw a card, the discard pile is shuffled into the draw pile.

Task 3: Playing a turn

The players present their drawn card and compare the values. The player with the higher value card, takes both cards and adds them to their discard pile, next to the draw pile. If the cards show the same value, the winner of the next turn wins these cards as well .

Hint: The game will likely result in a stalemate, if this rule is not implemented.

Tests:

1. When comparing two cards, the higher card should win
2. When comparing two cards of the same value, the winner of the next round should win 4 cards

Task 4: Console Output

Your program should output the cards that are played each turn and who wins. At the end the program should output the player that won.

Example output:

```
Player 1 (20 cards): 8
Player 2 (20 cards): 1
Player 1 wins this round

Player 1 (21 cards): 1
Player 2 (19 cards): 10
Player 2 wins this round

[...]

Player 1 (38 cards): 4
Player 2 (2 cards): 4
No winner in this round

Player 1 (37 cards): 7
```

```
Player 2 (1 cards): 3
Player 1 wins this round

Player 1 wins the game!
```