Introduction

Welcome! This is my terminal-based rendition of "Love Letter", a fairly simple game that I picked up with my friends and we still play to this day. It's an extremely short and sweet game where each round can easily be wrapped up in less than ten minutes, as such, I thought it would be perfect to do for a short coding challenge such as this well. Since it was suggested that we spend no more than 3 hours on this, I figured limiting the number of players to only two would simplify things a fair amount.

Rules

Sort ascending	Strength	Number in deck	Effects
Guard	1	5	Player designates another player and names a type of card. If that player's hand matches the type of card specified, that player is eliminated from the round. However, **Guard** cannot be named as the type of card.
Priest/Spy	2	2	Player is allowed to see another player's hand.
Baron	3	2	Player will choose another player and privately compare hands. The player with the lower-strength hand is eliminated from the round.
Handmaiden	4	2	Player cannot be affected by any other player's card until the next turn.
Prince	5	2	Player can choose any player (including himself) to discard his hand and draw a new one. If the discarded card is the **Princess**, the discarding player is eliminated.
King	6	1	Player trades hands with any other player.
Countess	7	1	If a player holds both this card and either the **King** or **Prince** card, this card must be played immediately.
Princess	8	1	If a player plays this card for any reason, he is eliminated from the round.

- Taken from https://en.wikipedia.org/wiki/Love_Letter_(card_game), here is a table describing each card, their strength, the number of cards in the deck, and their effect
- We will be starting with a deck of 15 cards, with 1 set off to the side every game so that there will be an factor of uncertainty with which cards may or may not be in play.
- Each player must draw and play one card during their respective turn.

Game Flow

- The 2 players will start out with one card each.
- During their respective turns, the player will start by drawing a card and then playing their card.
- If one of the players wins by a card effect, then that round is over, and we start from scratch.
- However, if a player does not win by a card effect before they go through all the cards, then the players at the end compare by card strength to determine a victor.

Design Choices

There isn't too much to explain here: most of this is fairly simple Java that doesn't really require any obscure libraries. The main data structure used here were just plain old arrays: there was no reason or need to use a different data structure since I only need a static, predetermined amount of slots to hold information about both the player's cards and about the deck of cards they draw from. A player will be holding onto at most two cards at any given time, so I allocated one array with 4 slots to hold all the information about the 2 players' hands. This information can be quickly accessed through array indices, and if I were to revisit this project and add more players, I could simply allocate two more spaces in the array for each addition player. Similarly, for our deck of cards, we only have 16 cards in the first place, and will not be adding cards back to the deck at any given point in the game. Our 16 cards are generated at the start of the program, and then shuffled to emulate a deck. When drawing a card, we simply move from one index to the next until we hit the last card, instead of deleting and shifting the cards over, which would generate unnecessary instructions.

And...that's it!

To run the game, you may use a Terminal and either use the commands
"javac Main.java" and "java Main"
OR
"make" and "make run"

Enjoy!