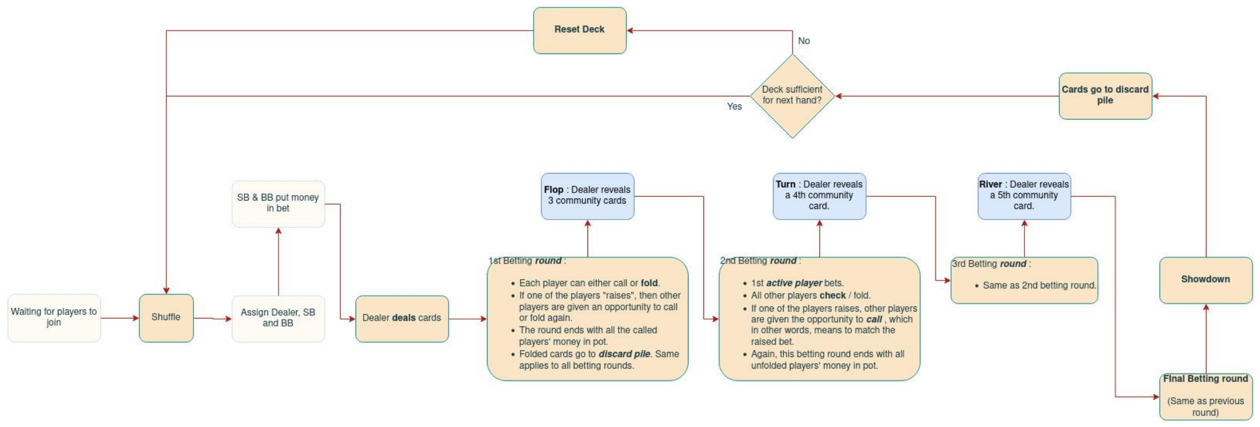
Deck of Cards

How is Poker played?


Reset Deck
No
Deck sufficient for next hand?
Yes
Cards go to discard pile
SB & BB put money in bet
Flop : Dealer reveals 3 community cards
Turn : Dealer reveals a 4th community card.
River : Dealer reveals a 5th community card.
1st Betting round :
Each player can either call or fold.
If one of the players "raises", then other players are given an opportunity to call or fold again.
The round ends with all the called players' money in pot.
Folded cards go to discard pile. Same applies to all betting rounds.
2nd Betting round :
1st active player bets.
All other players check / fold.
If one of the players raises, other players are given the opportunity to call , which in other words, means to match the raised bet.
Again, this betting round ends with all unfolded players' money in pot.
3rd Betting round :
Same as 2nd betting round.
Showdown
Waiting for players to join
Shuffle
Assign Dealer, SB and BB
Dealer deals cards
Final Betting round
(Same as previous round)

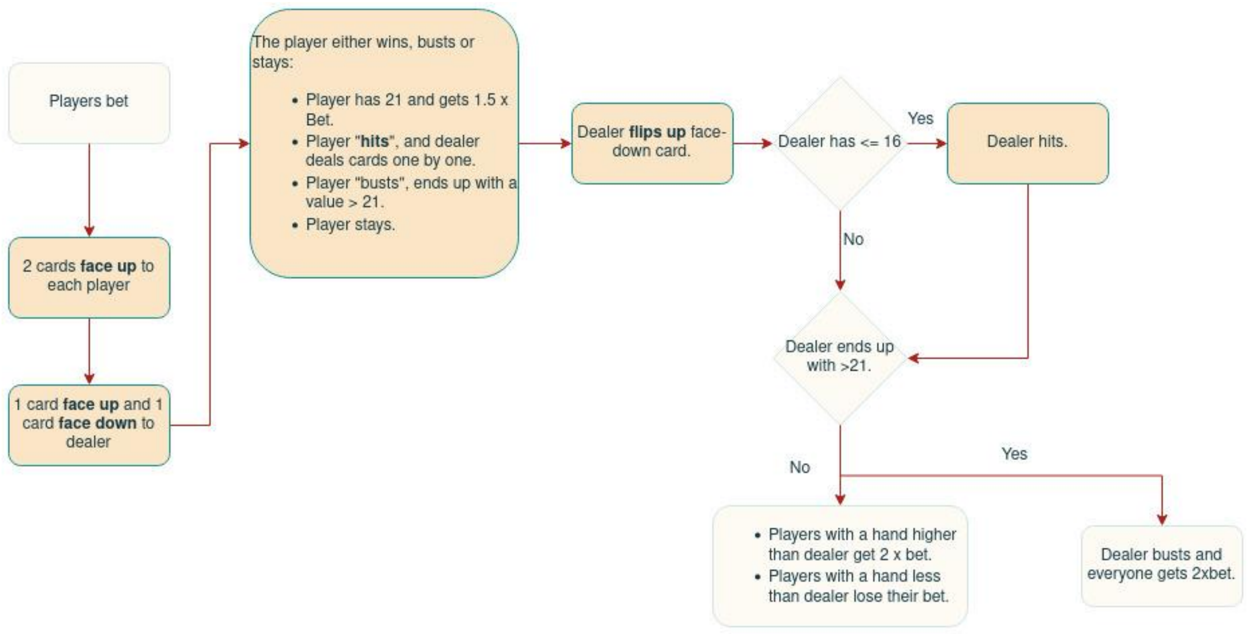
Shown above is a state-machine for how a Texas 500 Poker game is played that I deduced from this video. Let's analyze the interactions with the deck of cards.

- Deck is shuffled.
- 2 cards are dealt to each player.
- 5 community cards are dealt over the course of 4 betting rounds.
- New hand begins after showdown and deck is reshuffled.
- If there aren't enough cards in the deck, some of the cards can be taken back from the discard pile. But to simplify the game, the deck is simply *reset* i.e, a full deck is made.

Maximum number of players in this variant of poker is 10. This means (10*2) + 5 = 25 cards. This will be depleted in 2 hands and the deck would be reset for the 3rd hand.

How is Blackjack played?

Let's look at a different game where a full deck (in the variant I looked at) is made for each round. I used this video to study this game.


Players bet
2 cards face up to each player
1 card face up and 1 card face down to dealer
The player either wins, busts or stays:
Player has 21 and gets 1.5 x Bet.
Player "hits", and dealer deals cards one by one.
Player "busts", ends up with a value > 21.
Player stays.
Dealer flips up face-down card.
Dealer has <= 16
Yes
Dealer hits.
No
Dealer ends up with >21.
No
Yes
Players with a hand higher than dealer get 2 x bet.
Players with a hand less than dealer lose their bet.
Dealer busts and everyone gets 2xbet.

Notice the state in which the player keeps “hitting” i.e, demanding for another card. There is no upper bound placed on the number of cards that can be dealt to a player in each round, but no sane player could mathematically go above **13 cards** (4 As (if they choose to read an A as 1 instead of 11), 4 1S, 4 2s and 1 3. (unless they want to “bust” themselves intentionally. A standard blackjack game has a **maximum limit of 7 players**.

This means, once the player who ends up getting those 13 cards (next to impossible) and there are (6 players + 1 dealer) * 2 cards = 14 cards already dealt, there are (56 - (13 + 14)) = 29 cards in the deck. Assuming all players are sane (principle of least surprise), the next player could hit a maximum of 6 cards (3 3s and 3 4s). This means there are 23 cards left. The next player could then hit a maximum of 4 cards, we are left with 19 cards and 4 players. At this point, its trivial to see that the deck cannot be exceeded.

However, if it does turn out that the deck can be exceeded in the event that more than 7 players are allowed to participate. In this worst case scenario, no matter how unlikely it happens and assuming no one busts and there is nothing in the discard pile; a valid solution is to have multiple decks.

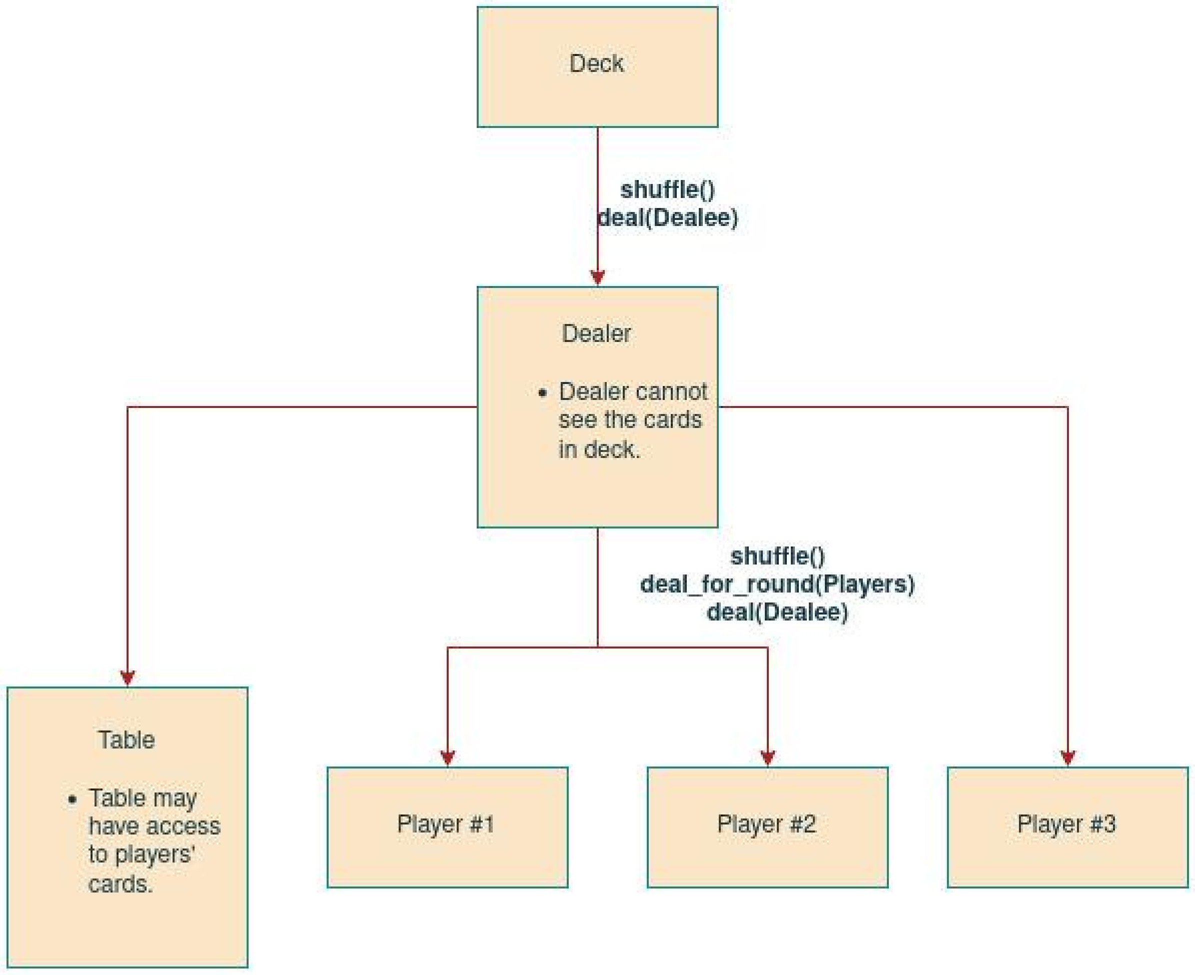
In Blackjack, the deck is reset and reshuffled at the beginning of each round. Therefore, we do not have to handle insufficiency in this game.

How does it differ from Poker?

- Deck is reset at the beginning of each hand. In poker, this is not done to make the game interesting. Smart players can count cards and guess the probability of how their hand is

compared to their players by eliminating what they could possibly have (previously used cards).
- Amount of cards that can be dealt in a poker hand is predetermined by the number of players, but that is not the case for blackjack.
- Cards are **face-up** and visible to everyone except for one.

Architecture & Interface Definition



- Due to the differing nature of the 100s of possible card games, sticking to the Open-closed principle for the Deck is prudent. This means the deck functionality is kept minimal and it is assumed it has no context of the game being played.
- In this case, the Proxy design pattern aligns most closely with our use-case. The players can only access the deck through the dealer. However, the dealer shouldn't have access to dealt cards (in different games, face-down cards are dealt). One way to tackle this is to pass the Player or the Table to the Deck. This way, the deck directly passes the cards to these "dealees" and the dealer doesn't see these cards.

- However, this might introduce the problem of creating a dependency with the players' library. To avoid this, we make one assumption about the "Dealee" objects. The objects provide a known interface called ***get_card().*** What happens inside this function is of no consequence to the Deck object as long as it is able to pass the card to this function.
- Thread-safety for the deck is promoted by having a single owner (the dealer) through which all calls are made.
- In conclusion, there are 2 entities which have separate concerns:
    - Deck, which is a data structure and manages only the cards it holds.
    - Dealer, which is an object and has / is provided the context of the game and can evaluate the conditions and determine the right way of interacting with the deck.

Best Data structure for Deck

- It must allow an efficient shuffling operation.
- Dealing is efficient (this is not really a significant problem for any data structure we can use).

- The shuffling process at its core involves swapping positions of 2 or more contiguous cards. We could do this with linked lists.
- The shuffling algorithms provided by the STL library do this exactly. The **std::random_shuffle** (C++14) does this by generating a random number and swapping the value pointed to by 2 iterators generated from the random numbers (indices) within the bounds of the data structure. The **std::shuffle** (C++17) provides more control over the random number generation process allowing us to also maintain a seed.
- We don't need a seed, but std::random_shuffle is deprecated. However, for simplicity (And assuming we are using C++14) It should suffice.
- The problem with using a linked list is that the swapping function used by the algorithm (std::iter_swap) takes O(n) time to get the iterator to the index by jumping pointers from the head. This is not the case with std::vector as the objects are contiguously stored in memory and the iterator can be "computed" instead of using traversal. Therefore, **std::vector** is clearly the better choice.
- For further efficiency (this is an overkill maybe), a std::array that maintains a copy of a full deck can be maintained for "resetting" the deck. An array only uses memory that is allocated (as opposed to the reserves maintained by a std::vector).
- We have not kept the option of putting in cards from a discard pile to keep things simple. However, if it were so, preventing duplication would require maintaining a separate hash map for faster checking of duplication (or avoiding it entirely). We couldn't however maintain a hash map as the default data structure for maintaining cards as both ordered and unordered hash maps use an implicit and explicit ordering mechanism which gets in the way of rearranging the objects.