Card Sharp

Design, implement and unit test a suitable OOP solution to represent a traditional playing card.

Cards should consist of one of four Suits (Clubs, Hearts, Spades, Diamonds) and one of 13 Face Values (Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King, Ace).

Card objects should have a toString method which will display a friendly name representing the card value eg. "Two of Clubs" etc.

It should be possible to take any two Card objects and determine which is the high card based on the Face Value. i.e. "Seven of Clubs" is higher than "Three of Spades" because Seven is higher than Three.
Aces should be considered higher than Kings, and two cards with the same Face Value, but different suits should be considered equal.

Now represent a Deck of 52 cards, either as another OOP object of your own design, or just a suitably populated Collection object from the Java Collection Framework.

Write a Driver class with 2 static methods which can be called from the main method. These methods should take a Deck as their input and then simulate and report the results (via print statements to the Console) of 2 simple card games between virtual players as per pseudocode/description below.

Note: these DO NOT need to be interactive games, they can simply run in a loop and use a random number generator to simulate any player decisions that would be necessary.

Game 1: High Card

Deck of 52 is divided evenly between two "players". (Should "shuffle" the deck first)

- On each turn, each player cuts their deck at a random point to choose one card at random to compare with the other player.
- Highest Card wins a point.
- Play 5 rounds to determine the winner (Best of 5, i.e. first to 3)

Game 2: Snap

Deck of 52 is divided evenly between two "players". (Should "shuffle" the deck first)

- Players take it in turn to turn over the top card of their "pile".
- This card is put face up onto another central "pile" visible to both players.
- If the cards both have the same value, one of the players may shout "snap". (use a random number gen to determine which player snapped first)
- The first player to shout "snap" when two cards of the same value are showing wins the round.
- The loser has to put all of the cards from the central face up pile onto the bottom of their face-down pile.

- The winner is the first person to get rid of all their cards.

Extra

Implement a method which will sort a shuffled deck back into sorted order. (Clubs 2-Ace, Hearts 2-Ace, Spades 2-Ace, Diamonds 2-Ace)

One Possible Sample Solution: **CardSharpChallenge_sampleSolution.zip (https://canvas.qub.ac.uk/courses/11041/files/1704177?wrap=1)** **(https://canvas.qub.ac.uk/courses/11041/files/1704177/download?download_frd=1)**

But be sure to try your own as there is no "right" way to do it. e.g. perhaps PlayingCard class should have a natural ordering based on face value? - it doesn't in my solution but the desired functionality is still achieved.

Note: I've updated this to include 2 Comparators for sorting by Suit. The first attempt was much more complex than necessary, but useful to leave it in for reference.