A Basic Deck of Cards Game

Your assignment is to code, in an object-oriented programming language (Java), a set of classes and a REST API that represent a deck of poker-style playing cards along with the services for a very basic game between multiple players holding cards. A deck is defined as follows: Fifty-two playing cards in four suits: hearts, spades, clubs, and diamonds, with face values of Ace, 2-10, Jack, Queen, and King.

The game API is a very basic game in which one or more decks are added to create a 'game deck', commonly referred to as a shoe, along with a group of players getting cards from the game deck.

You must provide the following operations:

- Create and delete a game
- Create a deck
- Add a deck to a game deck
    - Please note that once a deck has been added to a game deck it cannot be removed.
- Add and remove players from a game
- Deal cards to a player in a game from the game deck
    - Specifically, for a game deck containing only one deck of cards, a call to shuffle followed by 52 calls to dealCards(1) for the same player should result in the caller being provided all 52 cards of the deck in a random order. If the caller then makes a 53rd call to dealCard(1), no card is dealt. This approach is to be followed if the game deck contains more than one deck.
- Get the list of cards for a player
- Get the list of players in a game along with the total added value of all the cards each player holds; use face values of cards only. Then sort the list in descending order, from the player with the highest value hand to the player with the lowest value hand:
    - For instance if player 'A' holds a 10 + King then her total value is 23 and player 'B' holds a 7 + Queen then his total value is 19, so player 'A' will be listed first followed by player 'B'.
- Get the count of how many cards per suit are left undealt in the game deck (example: 5 hearts, 3 spades, etc.)
- Get the count of each card (suit and value) remaining in the game deck sorted by suit (hearts, spades, clubs, and diamonds) and face value from high value to low value (King, Queen, Jack, 10….2, Ace with value of 1)
- Shuffle the game deck (shoe)
    - Shuffle returns no value, but results in the cards in the game deck being randomly permuted. Please do not use library-provided "shuffle" operations to implement this function. You may use library- provided random number generators in your solution.

    - Shuffle can be called at any time

The structure of the REST API is up to you but please consider the combination of resources and actions represented by this problem and make appropriate tradeoffs in compliance to strict REST doctrine.

We have intentionally left many details of this assignment vague. You should follow the principle of least surprise in making reasonable decisions regarding the implementation.

While this is a trivial assignment, pretend that this code will become a foundational part of a new product. Take whatever measures you feel are required for your code to meet this bar within the scope of the allotted time and be prepared to discuss the tradeoffs you made.