ANJALI JAIN

CODE

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Random;

// Enum for card suits
enum Suit {
    SPADE, CLUB, HEART, DIAMOND
}

// Enum for card ranks
enum Rank {
    ACE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN,
KING
}

// Class representing a single card
class Card {
    private final Suit suit;
    private final Rank rank;

    public Card(Suit suit, Rank rank) {
        this.suit = suit;
        this.rank = rank;
    }

    public Suit getSuit() {
        return suit;
    }

    public Rank getRank() {
        return rank;
    }

    @Override
    public String toString() {
        return rank + " of " + suit;
    }
}

// Comparator for sorting cards
class CardComparator implements Comparator<Card> {
    @Override
    public int compare(Card card1, Card card2) {
        // First, compare based on color
```

```
        int colorCompare = getColorValue(card1.getSuit()) -
getColorValue(card2.getSuit());
        if (colorCompare != 0) {
            return colorCompare;
        }
        // Then, within each color, compare based on suit
        int suitCompare = card1.getSuit().ordinal() -
card2.getSuit().ordinal();
        if (suitCompare != 0) {
            return suitCompare;
        }
        // Finally, compare based on rank
        return card1.getRank().ordinal() - card2.getRank().ordinal();
    }

    // Helper method to determine color value
    private int getColorValue(Suit suit) {
        return (suit == Suit.HEART || suit == Suit.DIAMOND) ? 1 : 0;
    }
}

// Class representing a deck of cards
class Deck {
    private List<Card> cards;

    // Constructor to initialize deck with 52 cards
    public Deck() {
        cards = new ArrayList<>();
        for (Suit suit : Suit.values()) {
            for (Rank rank : Rank.values()) {
                cards.add(new Card(suit, rank));
            }
        }
    }

    // Method to shuffle the deck
    public void shuffle() {
        Collections.shuffle(cards);
    }

    // Method to draw a card from the deck
    public Card drawCard() {
        if (cards.isEmpty()) {
            throw new IllegalStateException("Deck is empty");
        }
        return cards.remove(cards.size() - 1);
    }
```

```
    // Method to draw multiple cards from the deck
    public List<Card> drawCards(int numCards) {
        List<Card> drawnCards = new ArrayList<>();
        for (int i = 0; i < numCards; i++) {
            drawnCards.add(drawCard());
        }
        return drawnCards;
    }

    // Method to check the size of the deck
    public int size() {
        return cards.size();
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a new deck
        Deck deck = new Deck();

        // Shuffle the deck (optional)
        deck.shuffle();

        // Draw 20 cards from the deck
        List<Card> drawnCards = deck.drawCards(20);

        // Sort the drawn cards using custom comparator
        Collections.sort(drawnCards, new CardComparator());

        // Display the sorted drawn cards
        for (Card card : drawnCards) {
            System.out.println(card);
        }
    }
}
```

ANJALI JAIN

OUTPUT

```
PS D:\java task> javac Main.java
PS D:\java task> java Main
TWO of SPADE
THREE of SPADE
FIVE of SPADE
SIX of SPADE
NINE of SPADE
TEN of SPADE
TWO of CLUB
THREE of CLUB
FIVE of CLUB
NINE of CLUB
TWO of HEART
NINE of HEART
TEN of HEART
ACE of DIAMOND
FOUR of DIAMOND
SIX of DIAMOND
EIGHT of DIAMOND
NINE of DIAMOND
TEN of DIAMOND
QUEEN of DIAMOND
PS D:\java task>
```

ANJALI JAIN

Explaination of code

Enums for Card Representation: Suit and Rank Enums represent the suit and rank of the playing cards, respectively. Enums are used here because they provide a convenient way to define custom constants.

Card Class - A card class represents a single playing card. There are two areas: Suits and Ranks, which store suits and ranks of cards. The constructor initializes these fields, and getter methods are provided to access them. The toString() method is overridden to provide the string representation of the card.

Configuring the Comparator: The CardComparator class implements the Comparator interface to define custom comparison logic for configuring cards. Papers are compared by color (red or black), then by uniform, and finally by rank. The getColorValue() method is a helper method for determining the color value of the given suite.

Deck Class: A deck class controls a deck of cards. It contains a list of Card objects that represent the deck. The creator starts the deck with 52 cards (one for each suit and rank combination). The shuffle() method uses Collections.shuffle() to shuffle the cards in the deck. The drawCard() method draws a card from the deck and removes it from the deck. The drawCards() method draws more cards from the deck. The size() method returns the number of cards remaining in the deck.

Main Class: The main class contains the main() method, which acts as the entry point of the program. In main(), the Deck object is created, shuffled (optional), and then 20 cards are drawn from the deck. These downloaded cards are stored in inventory and sorted by CardComparator. Finally, the list of documents is printed on the console.