Advanced Topics in Programming – "Stack Deck"

Complete the following methods of the Deck class. Use the Card class.
You will need to call some of these methods to properly implement other methods.

```
public class Deck {
	Stack<Card> deck;

	public Deck() {
		deck = new Stack<Card>();
		loadDeck();
	}

	public Stack<Card> getDeck() {
		return deck;
	}

	// Load the stack with 52 cards in order
	public void loadDeck(){}

	 // Print the stack - for uniformity,
	 // make 13 rows in four columns, filling
	 // in each row from left to right
	public String toString() {}

	// Return and remove the top card
	public Card deal() {}

	// Take the top half of the deck (26 cards) and alternate card by card with
	// the bottom half
	public void bridgeShuffle() {}

	// Split the deck at a random spot. Put the stack of cards above the random
	// spot below the other cards
	public void cut() {}

	// Complete a bridge shuffle and then cut the deck. Repeat 7 times
	public void completeShuffle() {}

	 // Reverse the order of the cards in the deck
	 private void reverse() {}

	 // Given a Stack of cards as an explicit parameter,
	 // reverse the order of the cards in the deck
	 private Stack<Card> reverse(Stack<Card> x) {}

	 // Combine two decks, one as the implicit
	 // parameter, the other as the explicit parameter
	 public void combineDecks(Stack<Card> other) {}



	public static void main(String[] args) {

	// example method calls - you should make your own
		Deck a = new Deck();
		System.out.println(a.getDeck());
		System.out.println(a);
		a.bridgeShuffle();
		System.out.println(a);
		a.cut();
```

```
            a.completeShuffle();
            System.out.println("After 1 bridge shuffle");
            System.out.println(a);
            a.reverse();

            for (int i = 1; i <= 5; i++)
                System.out.println(a.deal());
        }
}
```

```
public class Card implements Comparable<Card> {

        private int suit;
        private int value;

        /**
         * Constructor for objects of class Card
         */
        public Card() {
                suit = 1;
                value = 2;
        }

        public Card(int mySuit, int myValue) {
                suit = mySuit;
                value = myValue;
        }

        public int suit() {
                return suit;
        }

        public int value() {
                return value;
        }

        public String getValue() {
                String output = "";
                switch (value) {
                case 2:
                        output = "2";
                        break;
                case 3:
                        output = "3";
                        break;
                case 4:
                        output = "4";
                        break;
                case 5:
                        output = "5";
                        break;
                case 6:
                        output = "6";
                        break;
```

```
		case 7:
			output = "7";
			break;
		case 8:
			output = "8";
			break;
		case 9:
			output = "9";
			break;
		case 10:
			output = "10";
			break;
		case 11:
			output = "J";
			break;
		case 12:
			output = "Q";
			break;
		case 13:
			output = "K";
			break;
		case 14:
			output = "A";
			break;
		default:
			output = "Unknown type: " + value + "\n";
			break;
		}
		return output;
	}

	public String getSuit() {
		String output = "";
		switch (suit) {
		case 1:
			output = "Clubs";
			break;
		case 2:
			output = "Diamonds";
			break;
		case 3:
			output = "Hearts";
			break;
		case 4:
			output = "Spades";
			break;
		default:
			output = output + "Unknown type: " + suit;
			break;
		}
		return output;

	}

	public String toString() {
		return (getValue() + " of " + getSuit());
	}

	public int compareTo(Card other) {
		return value - other.value;
```

```
} }
```