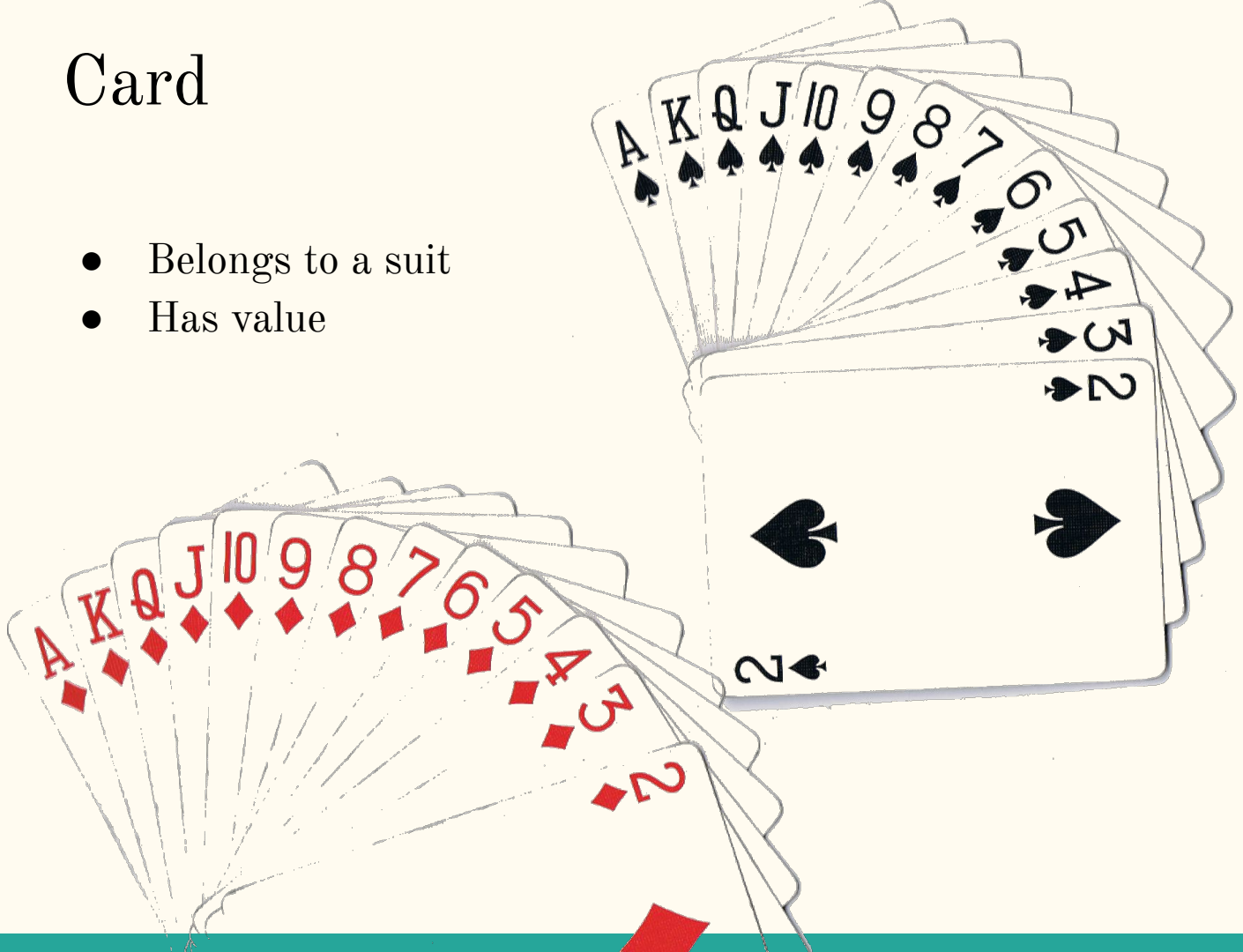
Cards, Deck & Hand

Card

- Belongs to a suit
- Has value



Card	
SUITS	String[]
RANKS	String[]
VALUES	int[]
suit	String
name	String
setSuit(String)	void
setName(String)	void
getName()	String
getSuit()	String
getValue()	int
toString()	String

Mutator methods

Card should belong to a suit &

have a valid name

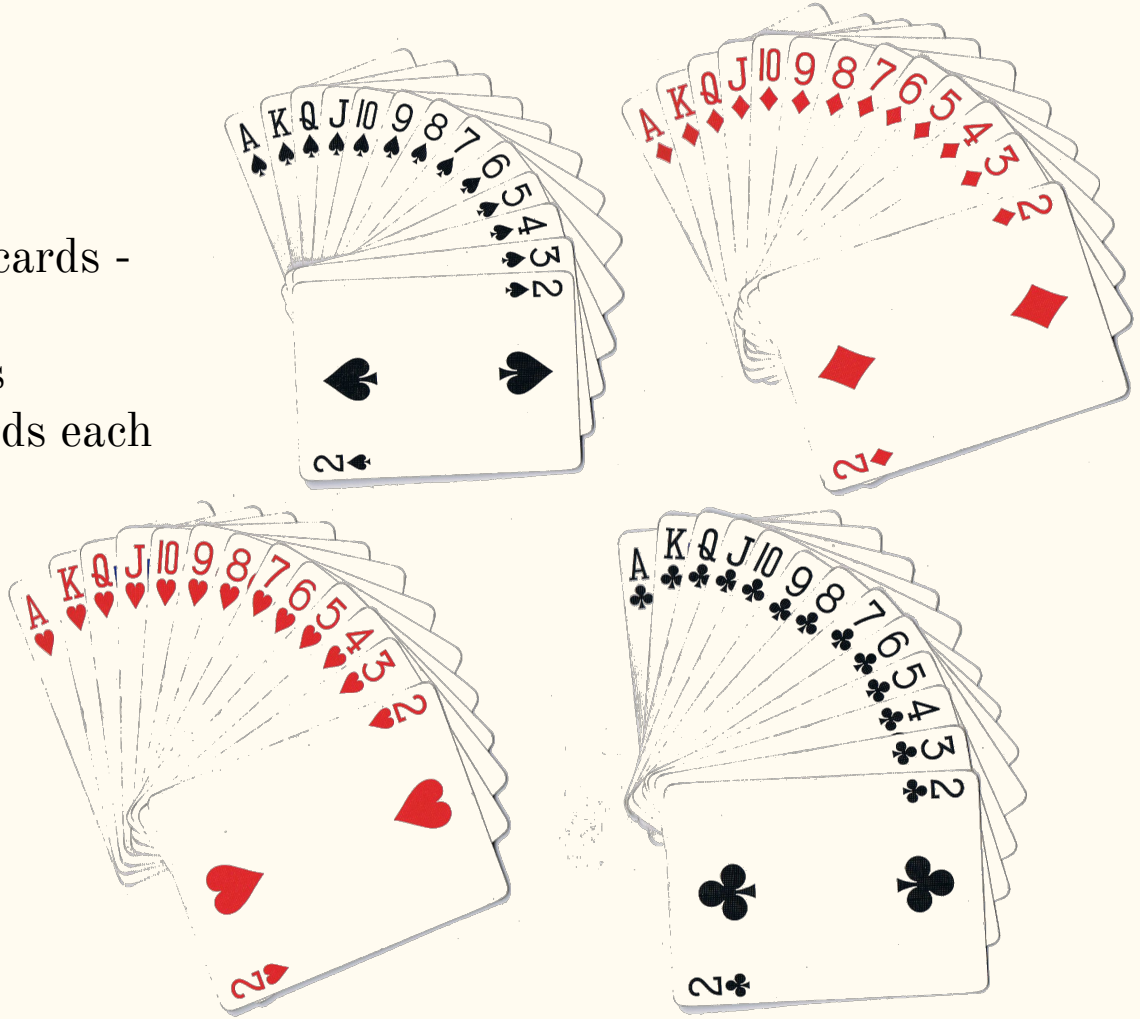
```java
public void setSuit(String suit) throws Exception {
    if (Arrays.asList(SUITS).contains(suit)) {
        this.suit = suit;
    } else {
        throw new Exception("\nInvalid Suit entered.");

    }
}

public void setName(String name) throws Exception {
    if (Arrays.asList(RANKS).contains(name)) {
        this.name = name;
    } else {
        throw new Exception("\nInvalid Name entered.");
```

```java
private final String[] SUITS = {"Clubs", "Diamonds","Hearts","Spades"};
private final String[] RANKS = {"Ace", "2", "3", "4", "5", "6", "7",
        "8","9","10","Jack","Queen","King"};
private final int[] VALUES = {1,2,3,4,5,6,7,8,9, 10,11,12,13};
private String suit;
private String name;
```

Deck

52 unique cards -

- 4 suits
- 13 cards each



```
Deck
SUITS          String[]
RANKS          String[]
VALUES         int[]
deck           ArrayList<Card>
shuffle()      void
addition(Card) void
remove()       Card
sort()         void
toString()     String
```

```java
public Deck() throws Exception {
    deck = new ArrayList<Card>( initialCapacity: 52);

    for (String suit : SUITS) {
        for (String name : RANKS) {
            deck.add(new Card(suit,name));
        }
    }
}

/**Shuffles the cards in the deck by randomly swapping every card in the deck.*/
public void shuffle(){
    for ( int i = deck.size()-1; i > 0; i-- ) {
        int rand = (int)(Math.random()*(i+1));
        Card temp = deck.get(i);
        deck.set(i, deck.get(rand));
        deck.set(rand, temp);
    }
}
```

Adding and Removing card

```java
/** Add a new card to the deck
 * @param add: Card that needs to be added to the deck
 */
public void addition(Card add) { deck.add(add); }

/** Removes the first card in the deck
 * @return the removed Card from the deck
 */
public Card remove() {
    Card removedCard = deck.get(0);
    deck.remove( index: 0);
    return removedCard;
}
```

Sorting a deck

```java
public void sort() {
    ArrayList<Card> sortedDeck = new ArrayList<>();
    while (deck.size() > 0) {
        int position = 0;
        Card prevCard = deck.get(0);
        for (int i = 0; i < deck.size(); i++) {
            Card nextCard = deck.get(i);
            if (nextCard.getValue() < prevCard.getValue()) {
                position = i;
                prevCard = nextCard;
            }
        }
        deck.remove(position);
        sortedDeck.add(prevCard);
    }
    deck = sortedDeck;
}
```

Hand

Set of 5 cards from a deck (sample)

```java
public class Hand extends Deck {
    public static int CARDSINHAND = 5;
    private static ArrayList<Card> hand;

    /**Default Constructor: sets a hand of Cards to an empty set of Cards
     */
    public Hand() throws Exception {
        super();
        hand = new ArrayList<Card>(CARDSINHAND);
    }

    @Override
    public void addition(Card add) {
        hand.add(add);
        int position = 0;
        for (int i = 0; i < deck.size(); i++) {
            if (deck.get(i).toString().equals(add.toString())) {
                position = i;
            }
        }
        deck.remove(position);
    }
```