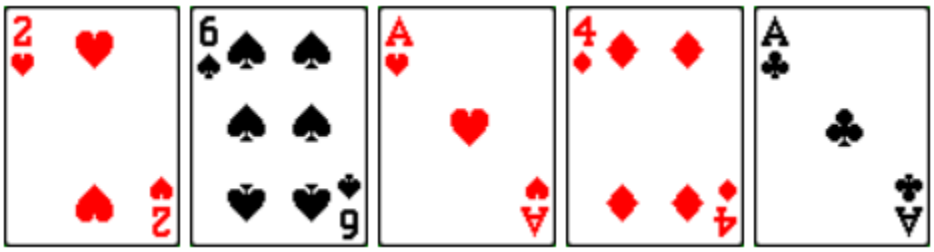
KnowTheFifth

https://threesixty360.wordpress.com/2008/08/04/math-card-tricks-i-know-the-5th-card/

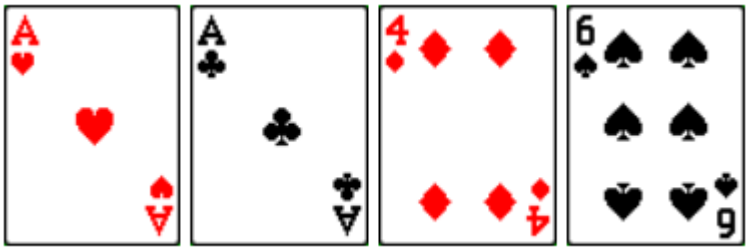
Math Card Tricks:

I was recently shown a card trick by a colleague and a former student. It was a two-person show, with the student playing the role of "Assistant" and our colleague playing the "Mind Reader". This is what we saw.

The Mind Reader left the room, and the Assistant explained that she could communicate with the Mind Reader telepathically. We chose five cards (total) from a standard deck of 52, then showed them to the Assistant. Here are our cards:



She picked one of the five (the ♥2), told us to remember it, hid it, then arranged the remaining four cards in a row, like so:



The Assistant then told the Mind Reader to enter (why didn't she just "think" him back in?). The Mind Reader studied the cards and proudly proclaimed that the last card was the ♥2. Ta-da!

After watching the trick two more times, it wasn't hard to figure out how the suit of the hidden card was signaled—the Pigeonhole Principle guarantees at least two cards in the same suit, so choose one of the cards from the repeated suit and place it first. But how to indicate the value? (Go ahead and think about it for a little while.)

The first step is to order the deck, typically from A-K within a given suit, and then order the suits (the contract bridge ordering of clubs, diamonds, hearts, spades being common). Now given any, oh, say **three** cards, we can label them HIGH, MIDDLE, and LOW. Why? Because there are now 3! = 6 permutations of these cards, and 6 is the *magic number*.

Next, suppose we're given five cards, two of which are of the same suit. Using the above example, we have the A and 2 of hearts. Hide the higher of the two with one caveat: if the cards are more than 6 apart, hide the lower card. (Here's why: If we add modulo 13, the lower card is actually no more than 6 "above" the higher card.)

Finally, arrange the other four cards so that the first card matches the suit of the hidden card, and the last three cards indicate how far the hidden card is "above" the first, using the following code:

LOW – MIDDLE - HIGH: +1
LOW – HIGH - MIDDLE: +2
MIDDLE – LOW - HIGH: +3
MIDDLE – HIGH - LOW: +4
HIGH – LOW - MIDDLE: +5
HIGH – MIDDLE - LOW: +6

Of course, we can always vary the location of the "suit indicator", especially if we're going to show the trick several times, and we could even vary the ordering of the cards, or the meanings of LMH, etc. A nice feature of this trick is that it is incredibly difficult to determine the "signal" that the Assistant uses to communicate the card to the Mind Reader, and if we only do the trick once, chances are nobody will figure it out.

According to Colm Mulcahy, this trick is known as "Fitch Cheney's Five Card Trick", and first appeared around 1950. (Read the *Math Horizons* article here. In the article, Mulcahy points out that this trick can be generalized to a 124-card deck. He also gives a variant that uses up/down (right-side up and upside-down) to encode the addition. After you've read the article, check out his other tricks (he does a column every month).

Now go forth and amaze your friends!

Now, turn the page and go forth and amaze yourself and your teammates by successfully completing this program.

You have been given the completed Card class shown below. Note the possible suits are: "club", "diamond", "heart", "spade" and the possible ranks are: 1 (or ace), 2, 3, …, 10, 11 (or jack), 12 (or queen), 13 (or king)

```
public class Card  {
/*
 *   suits: "club", "diamond", "heart", "spade"
 */
    private String suit;
/*
 *   rank: 1 = ace, 2, 3, …, 10, 11 (jack), 12 (queen), 13 (king)
 */
    private int rank;

    /**
     * Constructor for objects of class Card
     */
    public Card(String s, int r)  {
       suit = s;
       rank= r;
    }

    public int getRank()  {  return rank;  }

    public String getSuit()  {  return suit;  }
}
```

In this problem you will implement the following four methods: `isSmaller`, `arrangeCards`, `evaluateOrder` and `name5thCard`.

The `isSmaller` method returns the smaller of the two cards. First compare the rank and if the ranks are equal, use bridge ordering of suits: club (smallest), diamond, heart, spade (largest).

The following code shows the results of the `isSmaller` method.

The following code	Returns

`KnowTheFifth ktf = new KnowTheFifth();`
`Card c1 = new Card("heart", 2);`
`Card c2 = new Card("diamond", 5);`	

`ktf.isSmaller(c1, c2)`	C1

The following code shows the results of the `isSmaller` method.

The following code	Returns

`KnowTheFifth ktf = new KnowTheFifth();`
`Card c1 = new Card("spade", 2);`
`Card c2 = new Card("diamond", 2);`	

`ktf.isSmaller(c1, c2)`	C1

The `arrangeCards` method has a parameter `cards` with length 5. The method should return a `Card[]` containing four cards displayed to the Mind Reader in the proper order determining the fifth card. If multiple arrangements exist, return any of the arrangements

The following code shows the results of the `arrangeCards` method.

The following code	Returns

`KnowTheFifth ktf = new KnowTheFifth();`
 	
`Card[] ans = ktf.arrangeCards(new Card[] { new Card("heart", 2),`
`new Card("spade", 6), new Card("heart", 1),`
`new Card("diamond", 4), new Card("club", 1)});`	

`ans[0].getSuit()`	`"heart"`
`ans[0].getRank()`	`1`
`ans[1].getSuit()`	`"club"`
`ans[1].getRank()`	`1`
`ans[2].getSuit()`	`"diamond"`
`ans[2].getRank()`	`4`
`ans[3].getSuit()`	`"spade"`
`ans[3].getRank()`	`6`

The `evaluateOrder` method has a parameter `Card[]` with length 3. This method returns value according to the following criteria:

LOW – MIDDLE - HIGH: +1
LOW – HIGH - MIDDLE: +2
MIDDLE – LOW - HIGH: +3
MIDDLE – HIGH - LOW: +4
HIGH – LOW - MIDDLE: +5
HIGH – MIDDLE - LOW: +6

```
int evaluateOrder(Card[] cards)
```

The following code shows the results of the `evaluateOrder` method.

The following code	Returns

`KnowTheFifth ktf = new KnowTheFifth();`
 	
`Card[] order = new Card[] { new Card("heart", 2),`
`new Card("spade", 6), new Card("heart", 1)};`	

`ktf.evaluateOrder(order)`	`4`

The `name5thCard` has a parameter `Card[]` with length 4. This method returns the card indicate by the four cards in `cards`

The following code shows the results of the `evaluateOrder` method.

The following code	Returns

`KnowTheFifth ktf = new KnowTheFifth();`
 	
`Card[] theFour = new Card[] { new Card("heart", 2),`
`new Card("spade", 6), new Card("heart", 1),`
`new Card("club", 4)};`
 	
`Card fifthCard = ktf.name5thCard(theFour);`	

`fifthCard.getSuit()`	`"heart"`
`fifthCard.getRank()`	`7`