# Can you beat the dealer at 21?

## Model the game

This exercise is to model a card game with the following rules.

### Game rules

The game is played with a single deck of playing cards.
There are only two players (in this case called Sam and the Dealer) who will play against each other. Initially each player is given 2 cards from the top of a deck of cards. Cards are given in the following order: `sam, dealer, sam, dealer`
The score of a player's hands is the sum of the individual cards.
The score of numbered cards are their point value. Jack, Queen and King count as 10 and Ace counts as 11.
If either player has Blackjack with their initial hand they win the game. (Blackjack is an initial score of 21 with two cards: `A + \[10, J, Q, K\])`
Sam wins when both players start with Blackjack
Dealer wins when both players start going bust with a value above 21: `(A + A)`
If neither player has Blackjack then Sam will start drawing cards from the top of the deck Sam
    will draw a card until the score of their hand is 17 or higher
    Sam will not draw more than their first two cards, if their initial hand is 17 or higher Sam
    has lost the game if their total reaches 22 or higher

When sam has stopped drawing cards the dealer will start drawing cards from the top of the deck The
    dealer will stop drawing cards when their total is higher than Sam's.
    The dealer has lost the game their total reaches 22 or higher

## Your task

Your program should model this behaviour and determine which player wins the game. The program should be deterministic. If the same deck is provided, the result should always be the same.

### Input

The game should be able to read a file containing a deck of cards, taking the path to the file as a command line argument, as a starting point. If no file is provided, a new shuffled deck of 52 unique cards should be initialized.

Example program execution: `java BeatTheDealer deck-of-cards.txt`

The list is in the following format:

```
CA, D4, H7, SJ,..., S5, S9, D10
```

Suits:

```
C: Clubs
D: Diamonds
H: Hearts
S: Spades
```

Values:

```
 2: 2
 3: 3
 ....
 10: 10
 J: Jack
 Q: Queen
 K: King
 A: Ace
```

## Output

At the end, the solution should print the name of the winner to standard out, together with the hands of both sam and the dealer. Using the following format:

```
 [sam|dealer]
sam: card1, card2,..., cardN
dealer: card1, card2,..., cardN
```

### Example

When supplied with the following cardlist: `CA, D5, H9, HQ, S8` , the output should look like:

```
sam
sam: CA, H9
dealer: D5, HQ, S8
```

## Testing

Your solution should include tests.

## Other

You may use any build tool you like, but please include how to build (including required versions if any) and run your program in your delivery.