Purpose

The purpose of this document is to capture the flow of events of the "Top Trump" game in a command line (cl) mode, with an expert cl user.
With this document, the goal is to enlighten the UML diagram and the necessary classes.
The events are going to be presented as stories from the "cl" user point of view and what the user expects to see in the command line.

1 Open the game

A/A	*comments*	**command line window**
1	$	*(*java TopTrumps.jar -c)
	welcome msg:	–Welcome–
		–Are you ready to play Top Trumps?–
2	showMenu()	type the number and press enter
	getAction()	1.- Play new game
		2.- View past statistics
		3.- Quit the game
3	$ newGame()	1
4	loadCards()	*"the controller initiates the model class for a newGame()"*
5	showGameStatus()	Round 1. Your opponents are:
		Computer 1, ... Computer 4
6	activePlayer()	**Your are not/are the active player**
7	nextCard()	Pick next Card (press Enter)
8	if active==TRUE	Select a category and press Enter:
		a.1 b.3 c.1
		d.2 e.0 f.6
9	showResult()	**You WON/lost this round/**
		There is a DRAW. Continue with the next card.

Continue the loop from step 5. until the user is the WINNER or is ELIMINATED by the computer.

A/A	*comments*	**command line window**
10	$	CONGRATULATIONS YOU WON THE GAME
11	getAction()	What do you want to do: Type the number and press Enter 1. Play new game 2. View past statistics 3. Quit

2 New Game -pre phase-

If the user select to *Play a new game* then the **Controller** should initiate the pre phase of the game, or the **Model**. The **Model** class consists of:

- loadCards() : a method that has the FileReader class in order to load the cards. Now, regarding the cards one possible way to store the information is 40 x 1 String Array.
- shuffleCards() : a method to allocate the card with a random index position between 0 and 39 after is loaded by the FileReader.
- createDecks() : a method to create 5 object arrays (1 for the user and 4 for the AI), called **Deck** [], with normal dealing, the top card goes to the user and continue clockwise. This method is not clear yet. Even the decision to store the card which is a string [] to an object array needs to be decided.
- assignActivePlayer() : again, a simple method that will decide randomly who is the active player for the 1st round.
- getGameStatus() : finally, the most critical method is the one that carries the information about the status of the game (round, active player, decks etc) to the **Controller**, which will inherit to the **View** class and to the **GameState** class.