Project Showdown: Design Document

Lucas Brossi Barbosa

June 21st, 2020

Project Description

Showdown project consists in a suit of 2-player thrilling card games, where you play against the computer. Users will be able to choose one of three popular card games to play: Blackjack, War, or 'Truco Paulista' (a popular card game among Brazilians).

Game Rules

The project will follow the standard game rules as described in the links below. In some cases, few adaptations will be necessary in order to make the game more suitable for a 2-player version where the opponent is a computer and not a human being.

Blackjack:
https://www.pagat.com/banking/blackjack.html

War:
https://www.pagat.com/war/war.html

Truco Paulista:
https://www.pagat.com/put/truco_br.html#paulista

Classes Specifications

The project code will be entirely composed of objects that relate frequently with each other. I intend to use 6 classes, as outlined below.

class PlayingCard

This class will be able to represent the 52 playing cards found on a standard deck, with suits in in ["♠", "♥", "♦", "♣"] and ranks in ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]. The value each card assumes for scoring purposes is game-dependent.

Attributes:
rank
suit
value

Methods:
__init__
__repr__
getrank
getsuit
getvalue

class CardDeck

This class will represent one or more decks of Playing Cards. It will be able to generate custom decks for each game depending on the game's rules (Truco, for example, requires a customized deck of 40 cards; War can be played with a custom selection of suits).

Attributes:
deck

Methods:
__init__
shuffle_deck

class Blackjack

This class will store all attributes and actions required to execute one round of Blackjack.

Attributes:
player_bet
player_cards
dealer_cards
player_score
dealer_score

Methods:
__init__
bet
deal
hit
split
insure
stand
score
assess
close

class War

This class will store all attributes and actions required to execute a round of War.

Attributes:
player_cards
dealer_cards
player_score
dealer_score

Methods:
__init__
deal
next
war
score
assess
close

class Truco

This class will store all attributes and actions required to execute a round of Truco Paulista.

Attributes:	Methods:
round_points	__init__
player_cards	deal
dealer_cards	play
player_score	truco
dealer_score	score
	assess
	close

class user_interface

This class will execute the interface with the user end-to-end. It will welcome the user, prompt him to choose the game he wants to play, and then control execution of all the game rounds, relying on the attributes and methods stored in the specific game classes. Users can abandon or switch game at any point in the application. User records will be stored during session.

Attributes	Methods
user_name	__init__
black_record	start_game
war_record	exit_game
truco_record	exit_app
game_list	
game	
round_num	
player_balance	
dealer_balance	

Classes Relationships: Inputs and Outputs

The cards used in all games will come from class PlayingCards. This class is accessed by the class Deck, which then creates custom decks for the games. The user_interface class will handle the user interface end-to-end, executing all the game rounds. Depending on the game chosen, it will access a specific set of methods stored in the classes Blackjack, War, and Truco.

Meeting Project Requirements

- This project will be composed entirely of objects, with scripting outside of objects limited to the strictly necessary;

- This project is expected to demand something around 300-500 lines of code;

- Code will require 6 separate classes as outlined above;

- The project will be run from the command line as a .py file;

- The project will do user error checking as well as have a help 'screen' or printed instructions for the user;

- All code will be well commented;

- Project will not use anything outside of the standard Anaconda-installed libraries.