WAR CARD GAME

# Overview

The War card game program is designed to simulate a game of war given a certain number of players and cards. The game has two main phases:

* Battles – when cards are drawn and compared.
* War – when multiple cards are drawn, face down. Wars conclude with battles to determine the victor.

At the end of each battle, whether from war or stand alone, the victor collects all cards involved. The game concludes when either a single player holds all the cards, or any opponents do not have sufficient cards to complete a war.

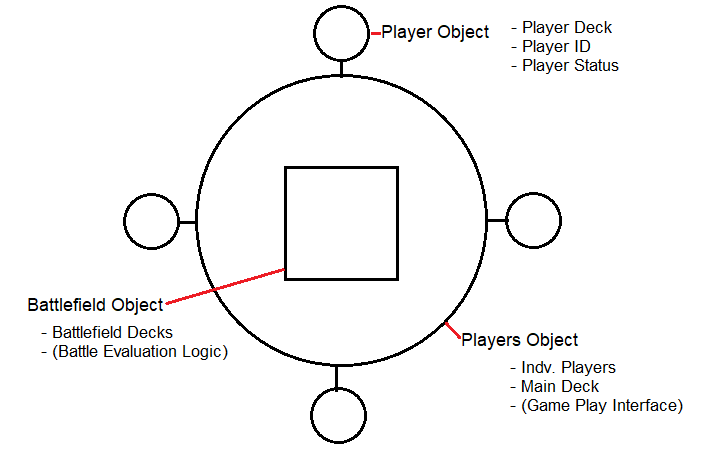
The game uses 5 files to simulate the card game.

* Cardobjects.py – simulates the physics of card decks. (see Card Objects.docx for details)
* War\_objects.py – simulates the players which includes card decks and additional details to facilitate game play
* War\_stat.py – the high level code using cardobjects and war\_objects to play and calculate statistics on game play.
* War\_objects\_print.py – Copy of war\_objects.py with added structure to enable printing results.
* War\_game\_print.py – Copy of war\_stat.py without statistical functions for running war\_objects\_print.py

# War\_objects.py overview

War\_objects contain three main objects:

* Player – This maintains the player card decks, player id, and status of each individual player.
* Players – This contains the initial main deck, all individual players, and the logic for game play.
* Battlefield – This contains the battle decks and the evaluation logic for each battle.



The players object is the main interface for game play, and internally uses the player and battlefield objects. Upon creation, the players object automatically generates a main deck of cards, shuffles it, and deals it out to all players. The players object only allows up to 5 decks of cards and 10 players. Decks are not allowed to be divided unevenly. Attempting to exceed these limits throws a value error.

# Player object

|  |  |
| --- | --- |
| **PROPERTIES** | **DESCRIPTION** |
| Player\_ID | ID of the player – used for game play tracking and printing. |
| Is\_Active | Player status. Defaults to True – Set to False when player out of cards. |
| **METHODS** | **DESCRIPTION** |
| Take\_Card | Receives a card object and adds it to the playerhand (deck) object. Sets active status to True. |
| Give\_Card | Evaluates active status. Generates a player\_card object, 3-tuple, containing a card object, the player Id, and the player status. Sets active status to False if last card is given. |

# Battlefield object

|  |  |
| --- | --- |
| **METHODS** | **DESCRIPTION** |
| Take\_Card | Receives a player\_card objects, unpacks it, and adds the inner card object to a “battle” (deck) contained within a battle list. Sets player status based on player\_card object status. |
| Evaluate | Reviews the values of the last card of each battles within the battle list. Returns a winner or war status. |
| Return\_Cards | Gathers all cards from all battles within the battle list. Shuffles the cards and returns all cards to caller. |

# Players object

|  |  |
| --- | --- |
| PROPERTIES | DESCRIPTION |
| Winner | Sets or returns the ID of the winner. |
| Active\_Players | Returns the number of active players in the game. |
| METHODS | DESCRIPTION |
| Battle | Takes one player\_card object from each player and places on the battlefield. Calls battlefield.evaluate. Returns the result to the caller. |
| Draw | Takes n player\_card objects from each active player and places them on the battlefield. |
| Cards\_To\_Winner | Retrieves all cards from battlefield.return\_cards. Adds all cards to the winner’s deck. |