**Poker Design**

**Part I : best\_hands(hands)**

hands = array of hands ex: ["3S 4S 5D 6H JH", "4S 3S JD 3H 10H"]

1\_ **create poker object** for each hand // hand = "3S 4S 5D 6H JH"

2\_ call **poker.scorehand**( each hand)

* [ score, values\_ranks ]

// int, hash is sorted with reverse ranks keys first, then sorted with reverse ranks values

hand = “4S 3S JD 3H 10H" => [ 1, { 3=> 2, 11=> 1, 10=> 1, 4=> 1}]

3\_ compare score and values\_ranks **hash to find the winner**

\_compare score to find best hand

\_ **if same score**, then compare the ranks array = keys of the values\_ranks hash

hands = ['4S 2H 6S 2D JH', '2S 4H 6S 4D JH']

values\_ranks(1) => { 2=>2, 11=> 1, 6=> 1, 4=>1}; keys = [2, 11, 6, 4]

values\_ranks(2) => { 4=>2, 11=> 1, 6=> 1, 2=>1 } keys = [4, 11, 6, 2]

* [4, 11, 6, 2] wins

**Part II: Poker class**

1\_ constructor: =>

// parse hand to get suits, "4S 3S JD 3H 10H" => [ ‘S’, ‘S’, ‘D’, ‘H’, ‘H’]

**suits** array

**ranks hash** : keys are ranks sorted ASC, values are rank count

\_ change ‘10’ into ‘T’ in hand

\_ convert rank character into value: [4,3,J,3,T] => [4,3,11,3,10]

\_ create ranks\_hash: key Is rank, value is rank count: { 4=> 1, 3=>2, 11=>1, 6=>1 }

\_ sort ranks\_hash in ranks ASC

2\_ **poker.score\_hand** // calculate score

\_ check **for flush**: [ ‘S’, ‘S’, ‘D’, ‘H’, ‘H’] => false

\_ check for **special straight** [2,3,4,5,14] => modify ranks hash

\_ check **for straight**

\_ **calculate score: 0-8**

straight && flush => score = 8

kinds [ 4,1] => score = 7 // square

kinds [ 3,2] => score = 6 // full house

flush => score = 5

straight => score = 4

kinds [ 3,1,1] => score = 3 // three

kinds [ 2,2,1] => score = 2 // double pair

kinds [ 2,1,1,1] => score = 1 // one pair

else => score = 0

\_ **create values\_ranks** with ranks\_hash

\_ make sure ranks\_hash is sorted ranks reverse

\_ then sorted ranks\_hash with reverse values

**return [ score, values\_ranks** ]

score\_hand('4S 2H 6S 2D JH') =>

score = 1

values\_ranks(1) => { 2=>2, 11=> 1, 6=> 1, 4=>1};

return [1, { 2=>2, 11=> 1, 6=> 1, 4=>1} ]