**Poker Java Poker class**

package poker;

import java.util.HashMap;

import java.util.Map;

import java.util.List;

import java.util.TreeMap;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import java.util.Comparator;

/\* Poker in java,

input one hand parse the hand for score and a List

poker.scoreHand() => PokerResult object

PokerResult = { int score, List valuesRanks }

BestHand will compare two PokerResult objects

=> return a1 < a2 => -1, a1==a2 => 0; a1>a2 => 1

\*/

**public class Poker** {

public static final String RANKS\_INDEX = "--23456789TJQKA";

public static final Integer[] SPECIAL\_STRAIGHT = new Integer[]{ 2,3,4,5,14};

String hand;

private char suits[];

// it creates in createRanksHash()

private Integer[] ranksSorted = new Integer[5];

// store rank as key, value is count of rank

private HashMap<Integer, Integer> ranks\_hash = new HashMap<>();

// use to create PokerResult object, this should be reset to 0 in scoreHand()

int score = 0;

**Poker(String handStr**) {

// BestHand class will pass in each hand

hand = handStr.replace("10", "T"); // replace 10 to 'T'

suits = parseHand(hand, true); // get suits

createRanksHash(hand); // create ranks\_hash

// scoreHand(); //for testing Poker.java by itself

}

// parse hand to create suits and ranks char[],

// but only returns one of them

// if argument flag = true => suits char[], flag = false => ranks char[]

private **char[] parseHand**(String handStr, boolean flag) {

char[] suits = new char[5];

char[] ranks = new char[5];

// take out spaces

String str = handStr.replaceAll(" ", "");

// create two char[], one for suits and one for ranks

char[] handAry = str.toCharArray();

int idx1 = 0;

int idx2 = 0;

for( int i=0; i < handAry.length ; i++ ) {

if ( (i % 2) != 0) {

suits[idx1++] = handAry[i];

} else {

ranks[idx2++] = handAry[i];

}

}

// depend on flag return suits char[] or ranks char[]

if (flag) {

return suits;

} else {

return ranks;

}

}

// create ranks\_hash, key is rank, and value is count of the rank

private **void createRanksHash**(String handStr) {

char[] ranks = parseHand(handStr, false); // [ 6,5,K,5,T]

// convert ranks 'T', 'J', 'Q', 'A' into values

for( int i=0; i < ranks.length; i++) {

ranksSorted[i] = **RANKS\_INDEX.indexOf(ranks[i]);**

}

Arrays.sort(ranksSorted); //[ 5,5,6,10,13]

// create ranks hash with key= rank , value = count of rank

for(int i=0; i < ranksSorted.length; i++) {

**// if key already existed add 1**

if (ranks\_hash.containsKey(ranksSorted[i])) {

ranks\_hash.put(ranksSorted[i], ranks\_hash.get(ranksSorted[i]) + 1);

} else {

ranks\_hash.put(ranksSorted[i], 1);

}

}

}

/\*

poker.scoreHand() uses hand, suits, ranks\_hash

it calculate score by

- check for flush, special straight, straight, kinds

- get int score 0 - 8

create HashMap ranks\_hash

- with key= rank and value= rank's count

creates List valuesList

- ranks\_hash reverse sort by ranks

- then reverse sort by values

=> List valuesList

create PokerResult object

return PokerResult = { int score, List valuesList }

\*/

**public PokerResult scoreHand()** {

score = 0; // should be reset for each hand

// **check for flush**, work => false

boolean flush = true;

for(int i=1; i< suits.length; i++) {

if (suits[i] != suits[i-1]) {

flush = false;

break;

}

}

// **check for special straight** [ 2,3,4,5,14];

// so we can modify ranks\_hash

if ( Arrays.equals(ranksSorted, SPECIAL\_STRAIGHT)) {

// then change 14 to 1, change ranks\_hash => {1: 1, 2:2, 3:3, 4:4, 5:5}

ranks\_hash.clear(); //k I forgot to clear the hash

ranks\_hash.put(1,1);

ranks\_hash.put(2,2);

ranks\_hash.put(3,3);

ranks\_hash.put(4,4);

ranks\_hash.put(5,5);

// need to change ranksSorted from

//[ 2,3,4,5,14]; to [1,2,3,4,5]

ranksSorted = new Integer[]{1, 2, 3, 4, 5};

}

// **check for straight**

boolean straight = true;

for(int i=1; i < ranksSorted.length; i++) {

if (ranksSorted[i] != ranksSorted[i-1] + 1) {

straight = false;

break;

}

}

// create kinds\_collection ArrayList = values of ranks\_hash

Collection<Integer> kinds\_collection = new ArrayList<>();

kinds\_collection = ranks\_hash.values();

// convert Collection<Integer> into Object Array

Object[] kindsArray = kinds\_collection.toArray();

/\* **create kinds Integer array** to contain kinds, ex => [2,1,1,1]

\* I need kinds as Integer array to use sort() with Collections.reverseOrder()

\*/

Integer[] kinds = new Integer[kinds\_collection.size()];

// cast Object into int, then assign int to Integer

for(int i=0; i<kindsArray.length; i++) {

kinds[i] = (int) kindsArray[i];

}

// **reverse sort kinds Integer** array order, ex => [2,1,1,1]

Arrays.sort(kinds, Collections.reverseOrder());

Integer[] score\_7 = new Integer[]{4,1};

Integer[] score\_6 = new Integer[]{3,2};

Integer[] score\_3 = new Integer[]{3,1,1};

Integer[] score\_2 = new Integer[]{2,2,1,};

Integer[] score\_1 = new Integer[]{2,1,1,1};

**// calculate score: kinds, flush, straight**

if (flush && straight) {

score = 8;

} else if (Arrays.equals(kinds, score\_7)) {

score = 7;

} else if (Arrays.equals(kinds, score\_6)) {

score = 6;

} else if (flush) {

score = 5;

} else if (straight) {

score = 4;

} else if (Arrays.equals(kinds, score\_3)) {

score = 3;

} else if (Arrays.equals(kinds, score\_2)) {

score = 2;

} else if (Arrays.equals(kinds, score\_1)) {

score = 1;

} else {

score = 0;

}

// reverse sort on ranks of ranks\_hash(unordered HashMap)

// **to create ordered List ranksList on ranks**

List ranksList = entriesSortedByKeysOrValues(ranks\_hash, true);

/\*

\* I can't use entriesSortedByKeysOrValues(HashMap, flag),

\* because ranksList is ArrayList not HashMap.

\* So **to sort the ordered List by value => List valuesList,**

\* I just compare the value of ranksList and insert

\* in reverse order to create valuesList ArrayList

\*/

List valuesList = new ArrayList<Map.Entry<Integer, Integer>>();

Map.Entry<Integer, Integer> entry1;

Map.Entry<Integer, Integer> cur\_entry;

Integer cur\_value, value1;

/\* **create valuesList ArrayList by insert Map.Entry in reverse order**

\* the value of ranksList ArrayList

\*/

entry1 = (Map.Entry<Integer, Integer>) ranksList.get(0);

value1 = entry1.getValue();

valuesList.add(0, entry1);

for (int i=1; i< ranksList.size(); i++) {

cur\_entry = (Map.Entry<Integer, Integer>) ranksList.get(i);

cur\_value = cur\_entry.getValue();

**if ( cur\_value >= value1 ) {**

// insert this Map.Entry before entry1, set value1 = cur\_value

valuesList.add(i-1, cur\_entry);

value1 = cur\_value;

} else {

valuesList.add(i, cur\_entry);

}

}

PokerResult result = new PokerResult(score, valuesList);

return result;

}

public static void main(String[] args) {

String hand = "2S 8H 2S 8D JH";

Poker poker = new Poker(hand);

PokerResult result = poker.scoreHand();

}

// **compare two Map objects in a HashMap** with duplicated values, because values are not unique

// flag = true sort reverse key, flag = false sort reverse value

static <K, V extends Comparable<? super K>>

List<Map.Entry<Integer,Integer**>> entriesSortedByKeysOrValues(**Map<Integer,Integer> map, boolean flag) {

boolean flag\_K\_V = flag;

List<Map.Entry<Integer,Integer>> sortedEntries = new ArrayList<Map.Entry<Integer,Integer>>(map.entrySet());

Collections.sort(sortedEntries,

new Comparator<Map.Entry<Integer,Integer>>() {

@Override

public int compare(Map.Entry<Integer,Integer> e1, Map.Entry<Integer,Integer> e2) {

if (flag\_K\_V ) {

**// reverse sort by key**

return **e2.getKey().compareTo(e1.getKey());**

} else {

//k I didn't use this part, because ranksList is not HashMap

// reverse sort by value

return e2.getValue().compareTo(e1.getValue());

}

}

}

);

return sortedEntries;

}

}