**Classes**

**Card.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* Simple class to represent a playing card as an object

\* @author Jacob Huesman

\*/

public class Card {

private String suit, rank;

private int value;

/\*\*

\* Creates a new card object of a certain rank and suit

\* @param r rank

\* @param s suit

\* @throws Exception if rank or suit do not match that of a standard deck of playing cards

\*/

public Card(String r, String s) throws Exception{

switch(r.toUpperCase().trim().charAt(0)){

case '2':

rank = "2";

value = 2;

break;

case '3':

rank = "3";

value = 3;

break;

case '4':

rank = "4";

value = 4;

break;

case '5':

rank = "5";

value = 5;

break;

case '6':

rank = "6";

value = 6;

break;

case '7':

rank = "7";

value = 7;

break;

case '8':

rank = "8";

value = 8;

break;

case '9':

rank = "9";

value = 9;

break;

case '1':

rank = "10";

value = 10;

break;

case 'J':

rank = "J";

value = 11;

break;

case 'Q':

rank = "Q";

value = 12;

break;

case 'K':

rank = "K";

value = 13;

break;

case 'A':

rank = "A";

value = 14;

break;

default:

throw new Exception("Invalid Rank");

}

switch(s.toUpperCase().trim().charAt(0)){

case 'H':

suit = "H";

break;

case 'C':

suit = "C";

break;

case 'S':

suit = "S";

break;

case 'D':

suit = "D";

break;

default:

throw new Exception("Invalid Suit");

}

}

/\*\*

\* Returns the integer value used in comparisons of the rank of two cards

\* @return integer value of the card

\*/

public int getValue(){

return value;

}

/\*\*

\* Returns the rank of the card

\* @return rank

\*/

public String getRank(){

return rank;

}

/\*\*

\* Returns the suit of the card

\* @return suit

\*/

public String getSuit(){

return suit;

}

}

**CardHand.java**

import java.util.Iterator;

import java.util.NoSuchElementException;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* Class to represent a person arranging a group of cards in his or her hand

\* @author Jacob Huesman

\*/

public class CardHand implements Iterable {

private Position<Card> hearts, clubs, spades, diamonds;

private final LinkedPositionalList<Card> hand;

/\*\*

\* Creates a new hand object ready to have cards added

\*/

public CardHand(){

hand = new LinkedPositionalList<>();

}

/\*\*

\* Adds a card to the hand, given a specific rank and suit

\* @param r rank of card

\* @param s suit of the card

\* @throws java.lang.Exception if the specified rank and suit do not correspond to the rank and suit of a standard deck of cards

\*/

public void addCard(String r, String s) throws Exception{

Card card = new Card(r,s);

switch(card.getSuit()){

case "H":

if(hearts == null){

hearts = hand.addLast(card);

} else {

hearts = sort(hearts, card);

}

break;

case "C":

if(clubs == null){

clubs = hand.addLast(card);

} else {

clubs = sort(clubs, card);

}

break;

case "S":

if(spades == null){

spades = hand.addLast(card);

} else {

spades = sort(spades, card);

}

break;

case "D":

if(diamonds == null){

diamonds = hand.addLast(card);

} else {

diamonds = sort(diamonds, card);

}

break;

}

}

/\*\*

\* Adds an already existing card object to the hand

\* @param card card object

\*/

public void addCard(Card card){

switch(card.getSuit()){

case "H":

if(hearts == null){

hearts = hand.addLast(card);

} else {

hearts = sort(hearts, card);

}

break;

case "C":

if(clubs == null){

clubs = hand.addLast(card);

} else {

clubs = sort(clubs, card);

}

break;

case "S":

if(spades == null){

spades = hand.addLast(card);

} else {

spades = sort(spades, card);

}

break;

case "D":

if(diamonds == null){

diamonds = hand.addLast(card);

} else {

diamonds = sort(diamonds, card);

}

break;

}

}

/\*\*

\* Adds a card to the hand so the the suit of cards is arranged in descending order with the higher valued cards first and lower value cards to the right

\* @param start first card in the suit

\* @param card card to be added

\* @return

\*/

private Position sort(Position<Card> start, Card card){

Position<Card> current = start;

do {

if(current.getElement().getValue() < card.getValue()){

if(current == start){

return hand.addBefore(start, card);

} else {

hand.addBefore(current, card);

}

return start;

} else if(hand.after(current) == null){

hand.addAfter(current, card);

return start;

} else if(! hand.after(current).getElement().getSuit().equals(card.getSuit())){

hand.addAfter(current, card);

return start;

} else {

current = hand.after(current);

}

} while(true);

}

/\*\*

\* Removes the highest card of a specified suit from the player's hand

\* @param s String representation of the suit to be removed

\* @return highest card of the specified suit

\* @throws Exception if the String representation of the suit doesn't start with the letter of any particular suit

\*/

public Card play(String s) throws Exception{

Card card = new Card("2", s);

switch(card.getSuit()){

case "H":

hearts = hand.after(hearts);

return hand.remove(hand.before(hearts));

case "C":

clubs = hand.after(clubs);

return hand.remove(hand.before(clubs));

case "S":

spades = hand.after(spades);

return hand.remove(hand.before(spades));

case "D":

diamonds = hand.after(diamonds);

return hand.remove(hand.before(diamonds));

}

return card;

}

/\*\*

\* Returns an iterator of all the cards in the hand

\* @return iterator of all the cards in the hand

\*/

@Override

public Iterator<Card> iterator(){

return hand.iterator();

}

/\*\*

\* Returns an iterator for all the cards of a specific suit in the hand

\* @param s String representation of the suit to be removed

\* @return Iterator of all the cards of the specified suit in the hand

\* @throws Exception if the String representation of the suit doesn't start with the letter of any particular suit

\*/

public Iterator<Card> suitIterator(String s) throws Exception{

return new SuitIterator(s);

}

/\*\*

\* Defines an Iterator that can be used to iterate through all the cards of a specific suit in a hand

\*/

private class SuitIterator implements Iterator<Card> {

private Position<Card> next;

private boolean hasNext;

/\*\*

\* Creates an iterator for a specified suit

\* @param s suit to be iterated

\* @throws Exception if the String representation of the suit doesn't start with the letter of any particular suit

\*/

public SuitIterator(String s) throws Exception{

Card card = new Card("2", s);

switch(card.getSuit()){

case "H":

next = hearts;

break;

case "C":

next = clubs;

break;

case "S":

next = spades;

break;

case "D":

next = diamonds;

break;

}

hasNext = next != null;

}

/\*\*

\* Returns if there is another card of the specified suit

\* @return true - if there is a card; false - otherwise

\*/

@Override

public boolean hasNext() {

return hasNext;

}

/\*\*

\* Returns the next card of the specified suit

\* @return Card of the specified suit

\*/

@Override

public Card next() {

if(hasNext){

if(hand.after(next) == null){

hasNext = false;

return next.getElement();

} else if(!hand.after(next).getElement().getSuit().equals(next.getElement().getSuit())){

hasNext = false;

return next.getElement();

} else {

next = hand.after(next);

return hand.before(next).getElement();

}

} else {

throw new NoSuchElementException("No next element");

}

}

}

}

**Deck.java**

import java.util.Random;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Jacob Huesman

\*/

public class Deck {

private CircularlyLinkedList<Card> deck;

private final String[] rank = {"2","3","4","5","6","7","8","9","10","J","Q","K","A"};

private final String[] suit = {"H","C","S","D"};

private Random random = new Random();

private int cards;

public Deck(){

deck = new CircularlyLinkedList<>();

cards = 52;

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

for(int j=0; j<rank.length; j++){

try {

deck.addLast(new Card(rank[j],suit[i]));

} catch(Exception e){System.out.println("Not Working " + i + j);}

}

}

}

public Card card(){

for(int i=0; i<random.nextInt(cards); i++){

deck.rotate();

}

cards--;

return deck.last();

}

}

**Game.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* Class that represents a card game

\* @author Jacob Huesman

\*/

public class Game {

private final int players;

public final int maxCards;

private final Deck deck;

public final CardHand[] hands;

/\*\*

\* Creates a new game with a specified number of players and a max number of cards a player can hold

\* @param players number of players in this game

\* @param maxCards max number of cards a player can hold

\*/

public Game(int players, int maxCards){

this.players = players;

this.maxCards = maxCards;

deck = new Deck();

hands = new CardHand[players];

for(int i=0; i<players; i++){

hands[i] = new CardHand();

}

}

/\*\*

\* Deals a card to each player

\*/

public void getCard(){

for(int i=0; i<players; i++){

hands[i].addCard(deck.card());

}

}

}

**Lab07.java**

import java.util.Iterator;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* A client class that tests the classes by dealing a card game

\* @author Jacob Huesman

\*/

public class Lab07 {

public static void main(String args[]){

//Create a new game

Game game = new Game(4,13);

Card card;

Iterator iterate;

//Deal a hand and output the cards in each player's hand

for(int i=0; i<game.maxCards; i++){

game.getCard();

System.out.format("Card %02d:\n", i+1);

for(int j=0; j<game.hands.length; j++){

System.out.format("Player %d: ", j+1);

iterate = game.hands[j].iterator();

while(iterate.hasNext()){

card = (Card) iterate.next();

System.out.format("%2S%S ", card.getRank(), card.getSuit());

}

System.out.println("");

}

}

}

}

**Screen Captures**



