// Name: Shijie Shi  
// VUnetID: shis2  
// Email: shijie.shi@vanderbilt.edu  
// Class: CS1101, Vanderbilt University  
// Honor statement: I have neither given nor received unauthorized help on this assignment  
// Date: 07/02/2017  
  
// Description: Represents a single playing card.  
  
  
public class Card {  
   
 // DO NOT CHANGE THESE TWO LINES OF CODE  
 public static final String[] *suits* = { "Spades", "Hearts", "Clubs", "Diamond" };  
   
 public static final String[] *ranks* = { "Ace", "2", "3", "4", "5", "6", "7",  
 "8", "9", "10", "Jack", "Queen", "King" };  
  
 private int suit;  
 private int rank;  
  
 //initiate a constructor for card class  
 //precondition: the value of suit should between 0-3 inclusive  
 // the value of rank should between 0-12 inclusive  
 public Card ( int newSuit, int newRank ) {  
 if ( newSuit<0 || newSuit >3 ) {  
 throw new IllegalArgumentException();  
 }  
 if ( newRank <0 || newRank>12 ) {  
 throw new IllegalArgumentException();  
 }  
 suit = newSuit;  
 rank = newRank;  
 }  
  
 */\*\*  
 \* getSuit--this method returns the suit of the Card as a String value  
 \** ***@return*** *String, value of the suit of the Card  
 \*/* public String getSuit(){ return *suits*[suit]; }  
  
 */\*\*  
 \* getRank-- this method returns the rank of the Card as a String value  
 \** ***@return*** *String, rank of the Card  
 \*/* public String getRank(){ return *ranks*[rank]; }  
  
  
 */\*\*  
 \* toString-- this method returns a string that represents both the rank and suit of the card  
 \** ***@return*** *String, rank and suit of the card  
 \*/* public String toString(){ return getRank() + " of " + getSuit(); }  
  
 */\*\*  
 \* equals-- this method determines if two cards have the same rank and suit  
 \** ***@param*** *other an object (could be any kind of object)  
 \** ***@return*** *boolean, return true when two cards have the same suit and rank  
 \*/* public boolean equals( Object other ) {  
 if ( other instanceof Card ) {  
 Card compareCard = (Card) other;  
 return suit==compareCard.suit && rank==compareCard.rank;  
 }else { //not a Card object  
 return false;  
 }  
 }  
  
 */\*\*  
 \* getDeeptiValue-- this method returns the Deepti Value of the card  
 \** ***@return*** *integer  
 \*/* public int getDeeptiValue() {  
 if ( suit==0 || suit==1 ) { //suit is Spades or Heart  
 if( rank==0 ) { //rank is ace  
 return 2\*14;  
 }else { //rank is not ace  
 return 2\*(rank+1);  
 }  
 }else {  
 if ( rank==0 ) { //rank is ace  
 return 14;  
 } else { //rank is not ace  
 return rank+1;  
 }  
 }  
 }  
  
  
}

// Name: Shijie Shi  
// VUnetID: shis2  
// Email: shijie.shi@vanderbilt.edu  
// Class: CS1101, Vanderbilt University  
// Honor statement: I have neither given nor received unauthorized help on this assignment  
// Date: 07/02/2017  
  
// Description: Deck of cards.  
  
import java.util.Random;  
  
public class Deck {  
  
 public static final int *SIZE\_OF\_DECK* = Card.*suits*.length \* Card.*ranks*.length; //52  
   
 private Card[] deck;  
 private int numOfCards;  
 private Card lastCard;  
  
 //This is the constructor for the Deck class  
 public Deck() {  
 deck = new Card[*SIZE\_OF\_DECK*];  
 initialize();  
 lastCard = null;  
  
 }  
  
 */\*\*  
 \* initialize-- creates a Card object for each card in the deck  
 \*/* public void initialize() {  
 int number =0;  
 for ( int i=0; i<=3; i++ ) {  
 for ( int j=0; j<=12; j++ ) {  
 deck[ number ] = new Card(i,j);  
 number++;  
 }  
 }  
 numOfCards = *SIZE\_OF\_DECK*;  
 }  
  
 */\*\*  
 \* isEmpty-- this method tells if the deck is empty  
 \** ***@return*** *boolean, returns true when the deck is empty  
 \*/* public boolean isEmpty() {  
 for ( int i=0; i<*SIZE\_OF\_DECK*; i++) {  
 if( deck[i]!=null || numOfCards!=0 ) {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 */\*\*  
 \* numberOfCards-- this method will report the number of cards still remaining in the deck  
 \** ***@return*** *integer, the number of cards remaining in the deck  
 \*/* public int numberOfCards() {  
 int numberOfLeftCards = 0;  
 for ( int i=0; i<*SIZE\_OF\_DECK*; i++) {  
 if ( deck[i] != null ) {  
 numberOfLeftCards++;  
 }  
 }  
 return numberOfLeftCards;  
 }  
  
 */\*\*  
 \* draw-- picks a random card from those remaining in the deck, and return it to the caller  
 \** ***@return*** *Object Card, the card drawn from the remaining deck  
 \*/* public Card draw() {  
 if ( isEmpty() ) {  
 initialize();  
 }  
 //pick a random number from the number of rest cards  
 int randomCardIndex = (int) ( Math.*random*()\*numOfCards );  
 //assign the random picked card to the lastCard  
 lastCard = deck[randomCardIndex];  
 //swap the random picked card with the last non-null card in the deck  
 deck[randomCardIndex] = deck[ numOfCards-1 ];  
 //set the random picked to null (remove it from the deck)  
 deck[ numOfCards-1 ] = null;  
 //decrement number of cards left in the deck  
 numOfCards--;  
 //return the random picked card to the caller  
 return lastCard;  
 }  
  
 */\*\*  
 \* getLastDraw-- this method returns the last card drawn from the deck  
 \** ***@return*** *Object Card, returns the last card drawn from the deck  
 \*/* public Card getLastDraw() {  
 return lastCard;  
 }  
  
}