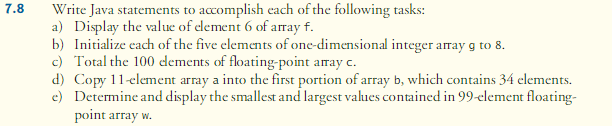
Lab6 CIS43 Due: 7/5/2016

Name: Nikhil Vytla

***Exercise: Ex 7.8, 7.17, 7.29, 7.30 and 7.31***



ArrayTasks

**package** P78;

/\* This program displays a variety of tasks using arrays.

\*

\* The output is as follows:

\* Part

\* Code

\* Result

\*

\* Each is separated by a blank line.

\* The code is limited to the task. The result is outputted using print lines.

\*/

**public** **class** ArrayTasks {

**public** **static** **void** main(String[] args) {

// part A

System.*out*.printf("PART A%n");

System.*out*.printf("Code: %nint[] f = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };%n");

System.*out*.println("System.out.println('Element 6 of array f is: %d%n', f[5]);");

**int**[] f = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };

System.*out*.printf("%nElement 6 of array f is: %d%n", f[5]);

// part B

System.*out*.printf("%nPART B%n");

System.*out*.printf("Code: %nint[] g = { 8, 8, 8, 8, 8 };%n");

**int**[] g = { 8, 8, 8, 8, 8 };

System.*out*.printf("%n%s%8s%n", "Index", "Value");

**for** (**int** counter = 0; counter < g.length; counter++){

System.*out*.printf("%5d%8d%n", counter, g[counter]);

}

// part C

System.*out*.printf("%nPART C%n");

String line =

"Code: \nfinal int ARRAY\_LENGTH = 100;\n" +

"int[] c = new int[ARRAY\_LENGTH];\n" +

"int total = 0;\n" +

"for (int counter = 0; counter < c.length; counter++){\n" +

" c[counter] = 1 \* counter;\n" +

" total += c[counter];\n" +

"}\n";

System.*out*.println(line);

**final** **int** ARRAY\_LENGTH = 100;

**int**[] c = **new** **int**[ARRAY\_LENGTH];

**int** total = 0;

**for** (**int** counter = 0; counter < c.length; counter++){

c[counter] = 1 \* counter;

total += c[counter];

}

System.*out*.println("Total for array c: " + total);

// part D

System.*out*.printf("%nPART D%n");

String line2 =

"Code: \nint [] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };\n" +

"int [] b = new int[34];\n" +

"System.arraycopy(a, 0, b, 0, a.length);\n";

System.*out*.println(line2);

**int** [] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };

**int** [] b = **new** **int**[34];

System.*arraycopy*(a, 0, b, 0, a.length);

System.*out*.printf("%s%8s%n", "Index", "Value");

**for** (**int** counter = 0; counter < b.length; counter++){

System.*out*.printf("%5d%8d%n", counter, b[counter]);

}

// part E

System.*out*.printf("%nPART E%n");

String line3 =

"Code: \nfinal int LENGTH = 99;\n" +

"int [] w = new int[LENGTH];\n" +

"int largest, smallest;\n" +

"int counter = 0;\n" +

"largest = w[1];\n" +

"smallest = w[1];\n" +

"for (counter = 0; counter < w.length; counter++){\n" +

" w[counter] = 1 \* counter;\n" +

"}" +

"for (counter = 0; counter < w.length; counter++){\n" +

" if (w[counter] > largest){\n" +

" largest = w[counter];\n" +

" }\n" +

" if (w[counter] < smallest){\n" +

" smallest = w[counter];\n" +

" }\n" +

"}\n" +

"System.out.printf('The largest number is %d.%n', largest);\n" +

"System.out.printf('The smallest number is %d.%n', smallest);\n";

System.*out*.println(line3);

**final** **int** LENGTH = 99;

**int** [] w = **new** **int**[LENGTH];

**int** largest, smallest;

**int** counter = 0;

largest = w[1];

smallest = w[1];

**for** (counter = 0; counter < w.length; counter++){

w[counter] = 1 \* counter;

}

**for** (counter = 0; counter < w.length; counter++){

**if** (w[counter] > largest){

largest = w[counter];

}

**if** (w[counter] < smallest){

smallest = w[counter];

}

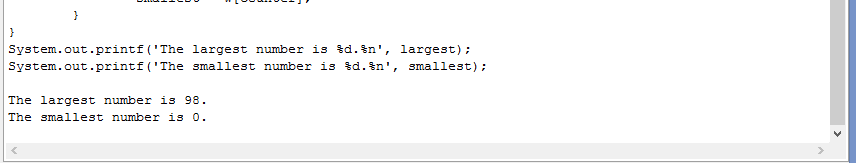
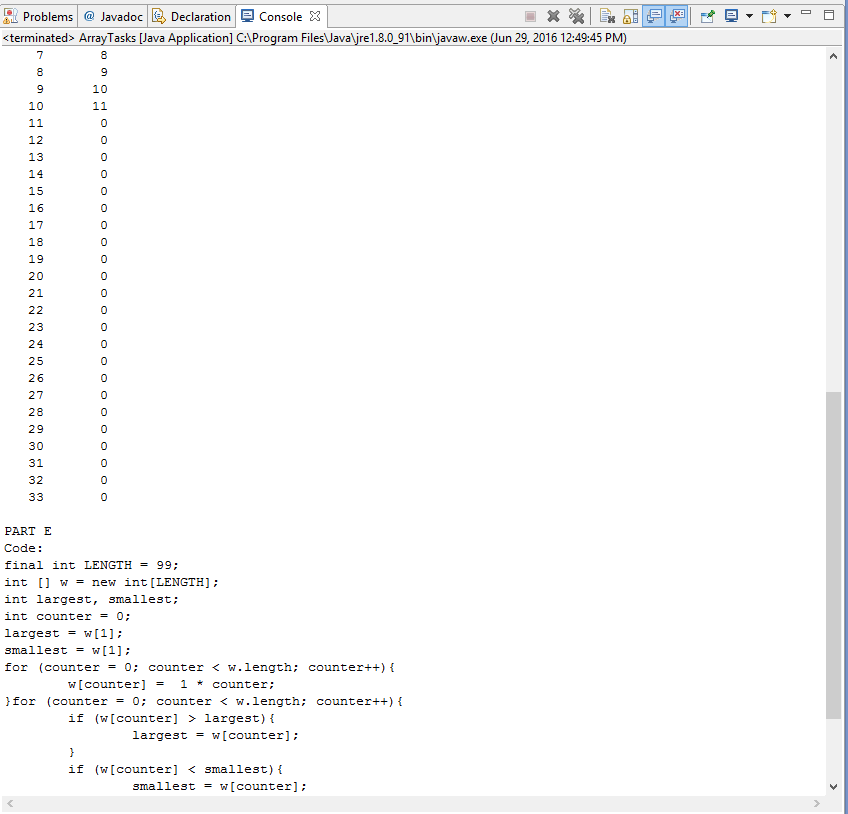
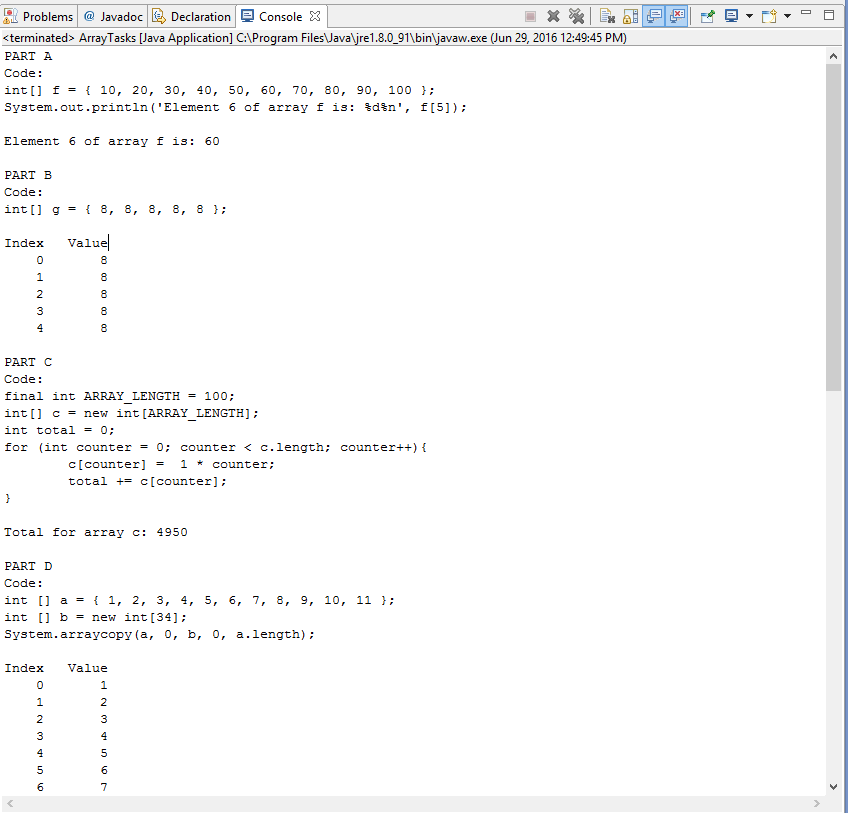
}

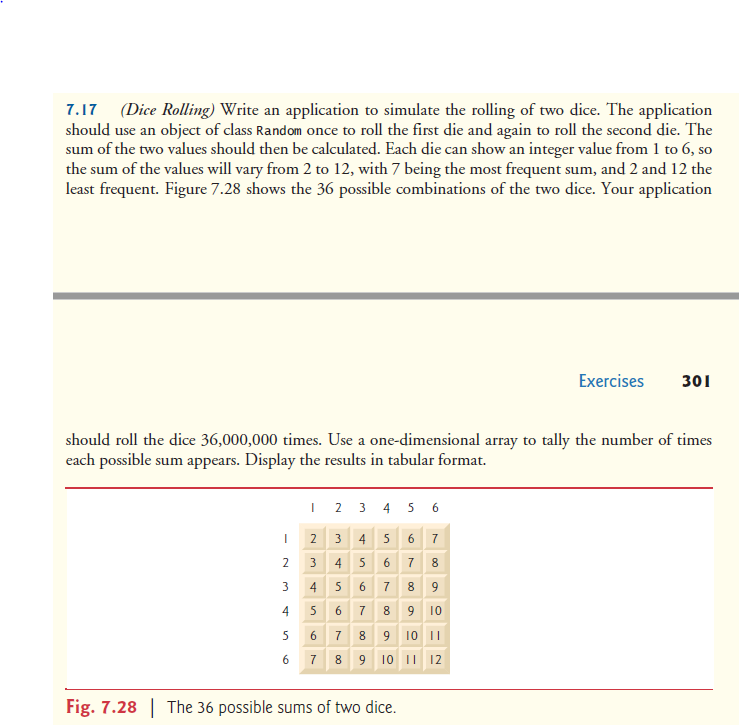
System.*out*.printf("The largest number is %d.%n", largest);

System.*out*.printf("The smallest number is %d.%n", smallest);

}

}





RollDice

**package** P717;

**import** java.security.SecureRandom;

**public** **class** RollDice {

**public** **static** **void** main(String[] args)

{

SecureRandom randomNumbers = **new** SecureRandom();

**int**[] frequency = **new** **int**[13];

**for** (**int** roll = 1; roll <= 36000000; roll++){

++frequency[1 + randomNumbers.nextInt(6) + 1 + randomNumbers.nextInt(6)];

}

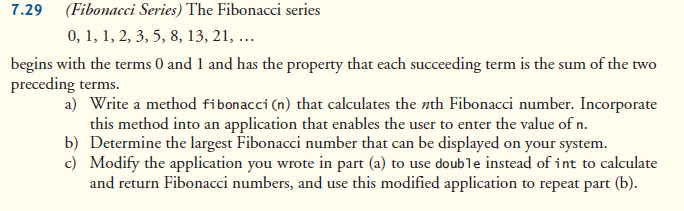
System.*out*.printf("%s%10s%n", "Face", "Frequency");

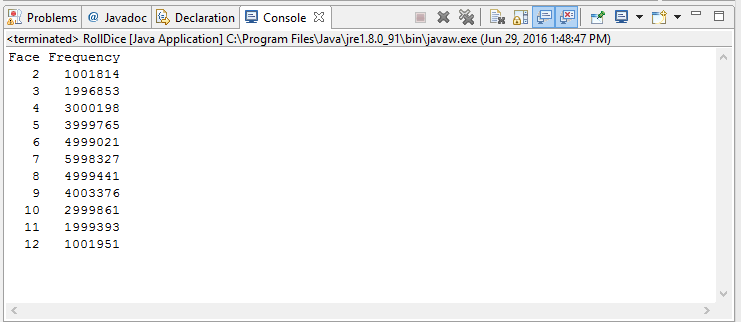
**for** (**int** face = 2; face < frequency.length; face++)

System.*out*.printf("%4d%10d%n", face, frequency[face]);

}

}





Fibonacci (uses int to calculate and return the number)

**package** P729;

**import** java.util.Scanner;

**public** **class** Fibonacci {

**public** **static** **void** main(String[] args) {

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter an integer n to find the nth fibonacci number: ");

**int** n = input.nextInt();

**int** fn = *fibonacci*(n);

System.*out*.printf("The nth fibonacci number is: %d%n%n", fn);

System.*out*.println("The largest, fully accurate number is the 47th Fibonacci number: 1836311903");

}

**public** **static** **int** fibonacci (**int** n) {

**if** (n <= 0)

System.*out*.println("You have entered a wrong number. Run the program again.");

**if** (n == 1)

**return** 0;

**if** (n == 2 || n == 3)

**return** 1;

**int** fib2 = 1, fib1 = 1, fibnum = 1;

**for** ( **int** i = 4; i <= n; i++){

fibnum = fib2 + fib1;

fib2 = fib1;

fib1 = fibnum;

}

**return** fibnum;

}

}

-------------------------------------------------

Fibonacci (uses double to calculate and return the number)

**package** P729;

**import** java.util.Scanner;

**public** **class** Fibonacci2 {

**public** **static** **void** main(String[] args) {

Scanner input = **new** Scanner(System.*in*);

System.*out*.print("Enter an integer n to find the nth fibonacci number: ");

**double** n = input.nextDouble();

**double** fn = *fibonacci*(n);

System.*out*.printf("The nth fibonacci number is: %.0f%n%n", fn);

System.*out*.println("The largest, fully accurate number is the 79th Fibonacci number: 8944394323791464");

}

**public** **static** **double** fibonacci (**double** n) {

**if** (n <= 0)

System.*out*.println("You have entered a wrong number. Run the program again.");

**if** (n == 1)

**return** 0;

**if** (n == 2 || n == 3)

**return** 1;

**double** fib2 = 1, fib1 = 1, fibnum = 1;

**for** ( **int** i = 4; i <= n; i++){

fibnum = fib2 + fib1;

fib2 = fib1;

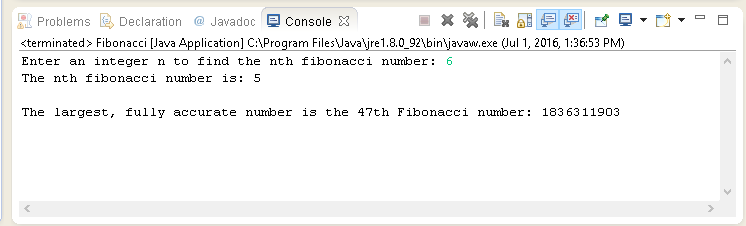
fib1 = fibnum;

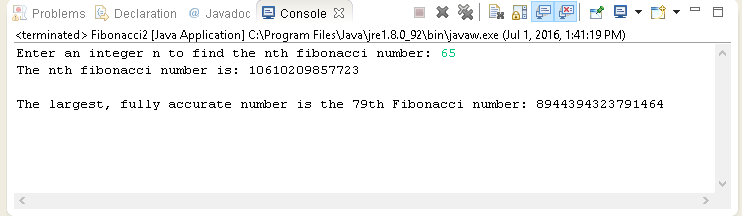
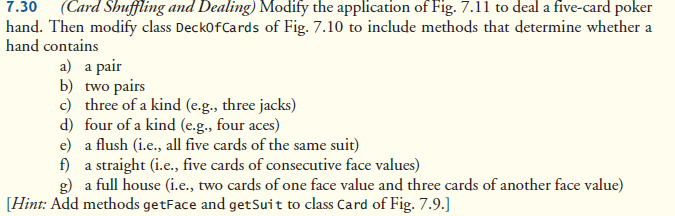
}

**return** fibnum;

}

}



Card

**package** P730;

// Fig. 7.9: Card.java

// Card class represents a playing card.

**public** **class** Card

{

**private** **final** String face; // face of card ("Ace", "Deuce", ...)

**private** **final** String suit; // suit of card ("Hearts", "Diamonds", ...)

// two-argument constructor initializes card's face and suit

**public** Card(String face, String suit)

{

**this**.face = face;

**this**.suit = suit;

}

// return String representation of Card

**public** String toString()

{

**return** face + " of " + suit;

}

**public** String getFace()

{

**return** face;

}

**public** String getSuit()

{

**return** suit;

}

} // end class Card

-------------------------------------------------

DeckOfCards

package P730;

// Fig. 7.10: DeckOfCards.java

// DeckOfCards class represents a deck of playing cards.

import java.security.SecureRandom;

public class DeckOfCards

{

private Card[] temp\_deck;

private Card[] deck; // array of Card objects

private int currentCard; // index of next Card to be dealt (0-51)

private static final int NUMBER\_OF\_CARDS = 52; // constant # of Cards

// random number generator

private static final SecureRandom randomNumbers = new SecureRandom();

// constructor fills deck of Cards

public DeckOfCards()

{

String[] faces = {"Ace", "Deuce", "Three", "Four", "Five", "Six",

"Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King"};

String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};

deck = new Card[NUMBER\_OF\_CARDS]; // create array of Card objects

currentCard = 0; // first Card dealt will be deck[0]

// populate deck with Card objects

for (int count = 0; count < deck.length; count++)

deck[count] =

new Card(faces[count % 13], suits[count / 13]);

}

// shuffle deck of Cards with one-pass algorithm

public void shuffle()

{

// next call to method dealCard should start at deck[0] again

currentCard = 0;

// for each Card, pick another random Card (0-51) and swap them

for (int first = 0; first < deck.length; first++)

{

// select a random number between 0 and 51

int second = randomNumbers.nextInt(NUMBER\_OF\_CARDS);

// swap current Card with randomly selected Card

Card temp = deck[first];

deck[first] = deck[second];

deck[second] = temp;

}

}

// deal one Card

public Card dealCard()

{

// determine whether Cards remain to be dealt

if (currentCard < deck.length)

return deck[currentCard++]; // return current Card in array

else

return null; // return null to indicate that all Cards were dealt

}

public void create\_deck\_of\_5\_cards()

{

temp\_deck = new Card[5];

for (int i = 0; i < 5; i++)

{

temp\_deck = this.deck;

}

}

public void checkPair()

{

int numOfPairs = 0;

for (int m = 0; m < 5; m++)

{

for (int i = m + 1; i < 5; i++)

{

if (temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfPairs++;

}

}

}

if (numOfPairs >= 1)

{

System.out.printf("%nThis hand has a pair.");

}

if (numOfPairs == 2)

{

System.out.printf("%nThis hand has two pairs.");

}

}

public void checkThreeAndFour()

{

int numOfMatches;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfMatches++;

}

}

if (numOfMatches == 2)

{

System.out.printf("%nThis hand has three of a kind.");

}

if (numOfMatches == 3)

{

System.out.printf("%n%nThis hand has four of a kind!!!!!!!!");

}

}

}

public void checkFlush()

{

int numOfMatches;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getSuit() == temp\_deck[i].getSuit())

{

numOfMatches++;

}

}

if (numOfMatches == 4)

{

System.out.printf("%nThis hand has a flush.");

}

}

}

public void checkStraight()

{

int cardValue[] = new int[5];

for (int i = 0; i < 5; i++)

{

if (temp\_deck[i].getFace().equals("Ace"))

cardValue[i] = 1;

else if (temp\_deck[i].getFace().equals("Deuce"))

cardValue[i] = 2;

else if (temp\_deck[i].getFace().equals("Three"))

cardValue[i] = 3;

else if (temp\_deck[i].getFace().equals("Four"))

cardValue[i] = 4;

else if (temp\_deck[i].getFace().equals("Five"))

cardValue[i] = 5;

else if (temp\_deck[i].getFace().equals("Six"))

cardValue[i] = 6;

else if (temp\_deck[i].getFace().equals("Seven"))

cardValue[i] = 7;

else if (temp\_deck[i].getFace().equals("Eight"))

cardValue[i] = 8;

else if (temp\_deck[i].getFace().equals("Nine"))

cardValue[i] = 9;

else if (temp\_deck[i].getFace().equals("Ten"))

cardValue[i] = 10;

else if (temp\_deck[i].getFace().equals("Jack"))

cardValue[i] = 11;

else if (temp\_deck[i].getFace().equals("Queen"))

cardValue[i] = 12;

else if (temp\_deck[i].getFace().equals("King"))

cardValue[i] = 13;

}

int numOfCard;

for (int m = 0; m < 5; m++)

{

numOfCard = 0;

for (int j = m + 1; j < 5; j++)

{

if (cardValue[m] + 1 == cardValue[j])

numOfCard++;

}

if (numOfCard == 4)

System.out.printf("%nThis hand has a straight.");

}

}

public void checkFullHouse()

{

int numOfPairs = 0;

for (int m = 0; m < 5; m++)

{

for (int i = m + 1; i < 5; i++)

{

if (temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfPairs++;

}

}

}

int numOfMatches;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfMatches++;

}

}

if (numOfPairs >= 1 && numOfMatches == 3)

System.out.printf("%n%nThis hand has a full house!!!!!!!!!!");

}

}

} // end class DeckOfCards

-------------------------------------------------

DeckOfCardsTest

**package** P730;

// Fig. 7.11: DeckOfCardsTest.java

// Card shuffling and dealing.

**public** **class** DeckOfCardsTest

{

// execute application

**public** **static** **void** main(String[] args)

{

DeckOfCards myDeckOfCards = **new** DeckOfCards();

//myDeckOfCards.shuffle(); // place Cards in random order

// print five Cards in the order in which they are dealt

**for** (**int** i = 0; i < 5; i++)

{

// deal and display five Cards

System.*out*.printf("%-19s", myDeckOfCards.dealCard());

**if** (i / 5 == 1) // output a newline after every fifth card

System.*out*.println();

}

myDeckOfCards.create\_deck\_of\_5\_cards();

myDeckOfCards.checkPair();

myDeckOfCards.checkThreeAndFour();

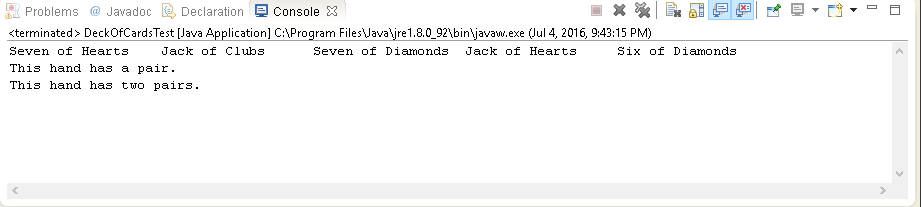
myDeckOfCards.checkFlush();

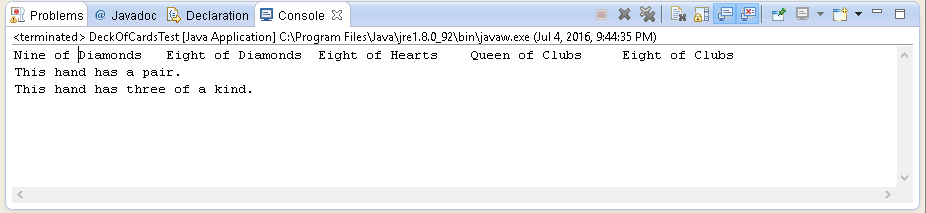
myDeckOfCards.checkStraight();

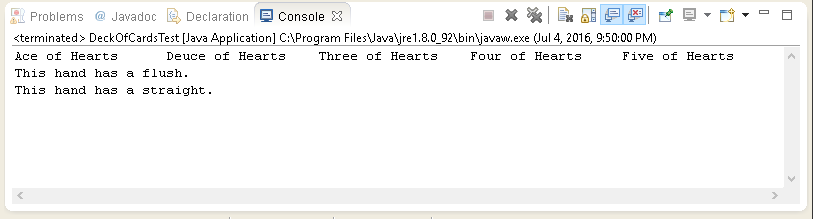
myDeckOfCards.checkFullHouse();

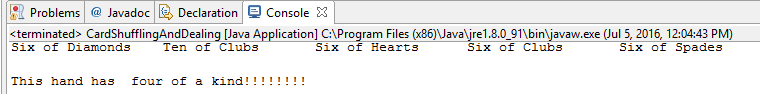
}

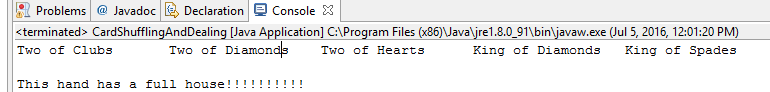
} // end class DeckOfCardsTest











\*Note\* The four of a kind and the full house have a different format than the rest because I was frustrated at the thought of not finding them in the program.



Card

**package** P731;

// Fig. 7.9: Card.java

// Card class represents a playing card.

**public** **class** Card

{

**private** **final** String face; // face of card ("Ace", "Deuce", ...)

**private** **final** String suit; // suit of card ("Hearts", "Diamonds", ...)

// two-argument constructor initializes card's face and suit

**public** Card(String face, String suit)

{

**this**.face = face;

**this**.suit = suit;

}

// return String representation of Card

**public** String toString()

{

**return** face + " of " + suit;

}

**public** String getFace()

{

**return** face;

}

**public** String getSuit()

{

**return** suit;

}

} // end class Card

-------------------------------------------------

DeckOfCards

package P731;

// Fig. 7.10: DeckOfCards.java

// DeckOfCards class represents a deck of playing cards.

import java.security.SecureRandom;

public class DeckOfCards

{

private Card[] temp\_deck;

private Card[] deck; // array of Card objects

private int currentCard; // index of next Card to be dealt (0-51)

private static final int NUMBER\_OF\_CARDS = 52; // constant # of Cards

// random number generator

private static final SecureRandom randomNumbers = new SecureRandom();

// constructor fills deck of Cards

public DeckOfCards()

{

String[] faces = {"Ace", "Deuce", "Three", "Four", "Five", "Six",

"Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King"};

String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};

deck = new Card[NUMBER\_OF\_CARDS]; // create array of Card objects

currentCard = 0; // first Card dealt will be deck[0]

// populate deck with Card objects

for (int count = 0; count < deck.length; count++)

deck[count] =

new Card(faces[count % 13], suits[count / 13]);

}

// shuffle deck of Cards with one-pass algorithm

public void shuffle()

{

// next call to method dealCard should start at deck[0] again

currentCard = 0;

// for each Card, pick another random Card (0-51) and swap them

for (int first = 0; first < deck.length; first++)

{

// select a random number between 0 and 51

int second = randomNumbers.nextInt(NUMBER\_OF\_CARDS);

// swap current Card with randomly selected Card

Card temp = deck[first];

deck[first] = deck[second];

deck[second] = temp;

}

}

// deal one Card

public Card dealCard()

{

// determine whether Cards remain to be dealt

if (currentCard < deck.length)

return deck[currentCard++]; // return current Card in array

else

return null; // return null to indicate that all Cards were dealt

}

public void create\_deck\_of\_5\_cards()

{

temp\_deck = new Card[5];

for (int i = 0; i < 5; i++)

{

temp\_deck = this.deck;

}

}

public int checkPair()

{

int numOfPairs = 0;

for (int m = 0; m < 5; m++)

{

for (int i = m + 1; i < 5; i++)

{

if (temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfPairs++;

}

}

}

if (numOfPairs >= 1)

{

return 1;

}

if (numOfPairs == 2)

{

return 2;

}

return numOfPairs;

}

public int checkThreeAndFour()

{

int numOfMatches = 0;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfMatches++;

}

}

if (numOfMatches == 2)

{

return 3;

}

if (numOfMatches == 3)

{

return 7;

}

}

return numOfMatches;

}

public int checkFlush()

{

int numOfMatches = 0;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getSuit() == temp\_deck[i].getSuit())

{

numOfMatches++;

}

}

if (numOfMatches == 4)

{

return 5;

}

}

return numOfMatches;

}

public int checkStraight()

{

int cardValue[] = new int[5];

for (int i = 0; i < 5; i++)

{

if (temp\_deck[i].getFace().equals("Ace"))

cardValue[i] = 1;

else if (temp\_deck[i].getFace().equals("Deuce"))

cardValue[i] = 2;

else if (temp\_deck[i].getFace().equals("Three"))

cardValue[i] = 3;

else if (temp\_deck[i].getFace().equals("Four"))

cardValue[i] = 4;

else if (temp\_deck[i].getFace().equals("Five"))

cardValue[i] = 5;

else if (temp\_deck[i].getFace().equals("Six"))

cardValue[i] = 6;

else if (temp\_deck[i].getFace().equals("Seven"))

cardValue[i] = 7;

else if (temp\_deck[i].getFace().equals("Eight"))

cardValue[i] = 8;

else if (temp\_deck[i].getFace().equals("Nine"))

cardValue[i] = 9;

else if (temp\_deck[i].getFace().equals("Ten"))

cardValue[i] = 10;

else if (temp\_deck[i].getFace().equals("Jack"))

cardValue[i] = 11;

else if (temp\_deck[i].getFace().equals("Queen"))

cardValue[i] = 12;

else if (temp\_deck[i].getFace().equals("King"))

cardValue[i] = 13;

}

int numOfCard = 0;

for (int m = 0; m < 5; m++)

{

for (int j = m + 1; j < 5; j++)

{

if (cardValue[m] + 1 == cardValue[j])

numOfCard++;

}

}

if (numOfCard == 4)

return 4;

return numOfCard;

}

public int checkFullHouse()

{

int numOfPairs = 0;

for (int m = 0; m < 5; m++)

{

for (int i = m + 1; i < 5; i++)

{

if (temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfPairs++;

}

}

}

int numOfMatches = 0;

for (int m = 0; m < 5; m++)

{

numOfMatches = 0;

for (int i = m + 1; i < 5; i++)

{

if(temp\_deck[m].getFace() == temp\_deck[i].getFace())

{

numOfMatches++;

}

}

if (numOfPairs >= 1 && numOfMatches == 3)

return 6;

}

return numOfPairs + numOfMatches;

}

} // end class DeckOfCards

-------------------------------------------------

DeckOfCardsTest

**package** P731;

// Fig. 7.11: DeckOfCardsTest.java

// Card shuffling and dealing.

**public** **class** DeckOfCardsTest

{

// execute application

**public** **static** **void** main(String[] args)

{

DeckOfCards myDeckOfCards = **new** DeckOfCards();

DeckOfCards anotherDeck = **new** DeckOfCards();

myDeckOfCards.shuffle(); // place Cards in random order

anotherDeck.shuffle();

// print five Cards in the order in which they are dealt

**for** (**int** i = 0; i < 5; i++)

{

// deal and display five Cards

System.*out*.printf("%-19s", myDeckOfCards.dealCard());

**if** (i / 5 == 1) // output a newline after every fifth card

System.*out*.println();

}

System.*out*.println();

**for** (**int** j = 0; j < 5; j++)

{

// deal and display five Cards

System.*out*.printf("%-19s", anotherDeck.dealCard());

**if** (j / 5 == 1) // output a newline after every fifth card

System.*out*.println();

}

myDeckOfCards.create\_deck\_of\_5\_cards();

**int** myTotal = myDeckOfCards.checkPair() +

myDeckOfCards.checkThreeAndFour() +

myDeckOfCards.checkFlush() +

myDeckOfCards.checkStraight() +

myDeckOfCards.checkFullHouse();

anotherDeck.create\_deck\_of\_5\_cards();

**int** anotherTotal = anotherDeck.checkPair() +

anotherDeck.checkThreeAndFour() +

anotherDeck.checkFlush() +

anotherDeck.checkStraight() +

anotherDeck.checkFullHouse();

**if** (myTotal > anotherTotal)

System.*out*.printf("%n%nThe hand from deck myDeckofCards is better.");

**else** **if** (anotherTotal > myTotal)

System.*out*.printf("%n%nThe hand from deck anotherDeck is better.");

**else** **if** (myTotal == anotherTotal)

System.*out*.printf("%n%nBoth hands are equal.");

}

} // end class DeckOfCardsTest

