6.3

Declare a one dimensional array name score of type int that can hold 9 values

int[] score = new int[9];

Declare a 2-dimensional array named price of type float that has 10 rows and 3 columns.

float[][] price = new float[10][3];

Declare and initialize a 2-dimensional array named matrix of type long that has 4 rows and 3 columns to have all it’s values set to 5.

long[][] matrix = {

{5, 5, 5},

{5, 5, 5},

{5, 5, 5},

{5, 5, 5}

};

Declare and initialize a one dimensional byte array named values of size 10 so that all entries contain 1

byte[] values = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};

byte[] values = new byte[10];

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

values[i] = 1;

}

Without typing in the code determine the output of the following program. int num[] = {7,7,6,6,5,5,4,4}; for(int i = 0; i < 8; i = i + 2) System.out.print(num[i]);

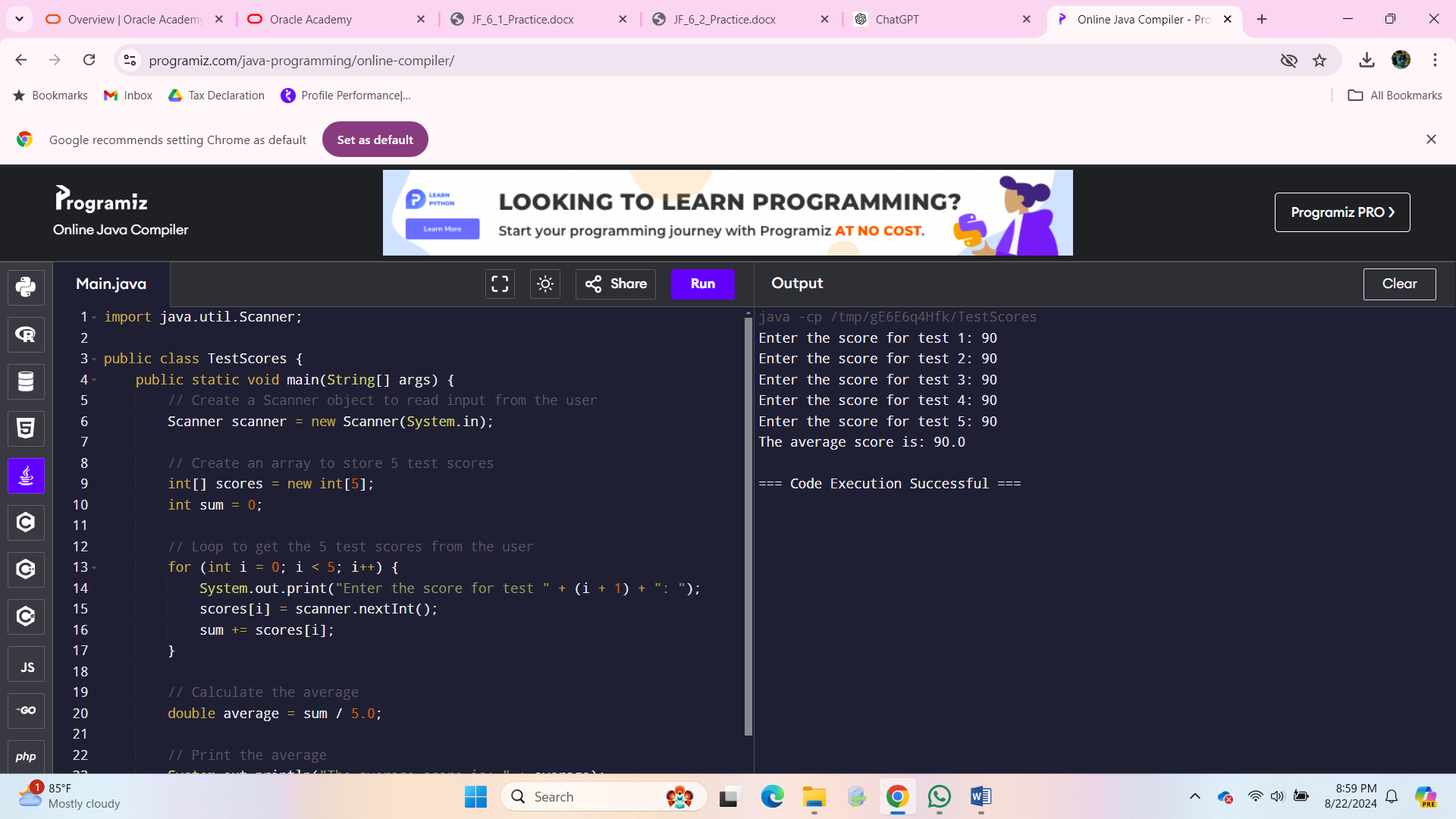
The program will output: 7654.

Without typing in the code determine the output of the following program. int[][] num = {{3,3,3},{2,2,2}}; int[] array = {4,3,2}; for(int i = 0; i < 3; i++){ num[1][i] = num[0][i]+array[i]; } for(int i = 0; i < 2; i++){ for(int j = 0; j < 3; j++){ System.out.print(num[i][j]); } System.out.println(); }

333

765

In a certain class, there are 5 tests worth 100 points each. Write a program that will take in the 5 tests scores for the user, store the tests scores in an array, and then calculate the stu



import java.util.Scanner;

public class Hollowtangle{

public static void main(String[] args){

int my\_input, k, i, j;

Scanner my\_scanner = new Scanner(System.in);;

System.out.print("Enter the size : ");

my\_input = my\_scanner.nextInt();

for( i=1;i<=my\_input;i++){

if(i==1 || i==my\_input)

for( j=1;j<=i;j++){

System.out.print("#");

} else {

for( j=1;j<=i;j++){

if(j==1 || j==i)

System.out.print("#");

else

System.out.print(" ");

}

}

System.out.println();

my\_input = my\_scanner.nextInt();

for( i=1;i<=my\_input;i++){

if(i==1 || i==my\_input)

for( j=1;j<=i;j++){

System.out.print("#");

} else {

for( j=1;j<=i;j++){

if(j==1 || j==i)

System.out.print("#");

else

System.out.print(" ");

}

}

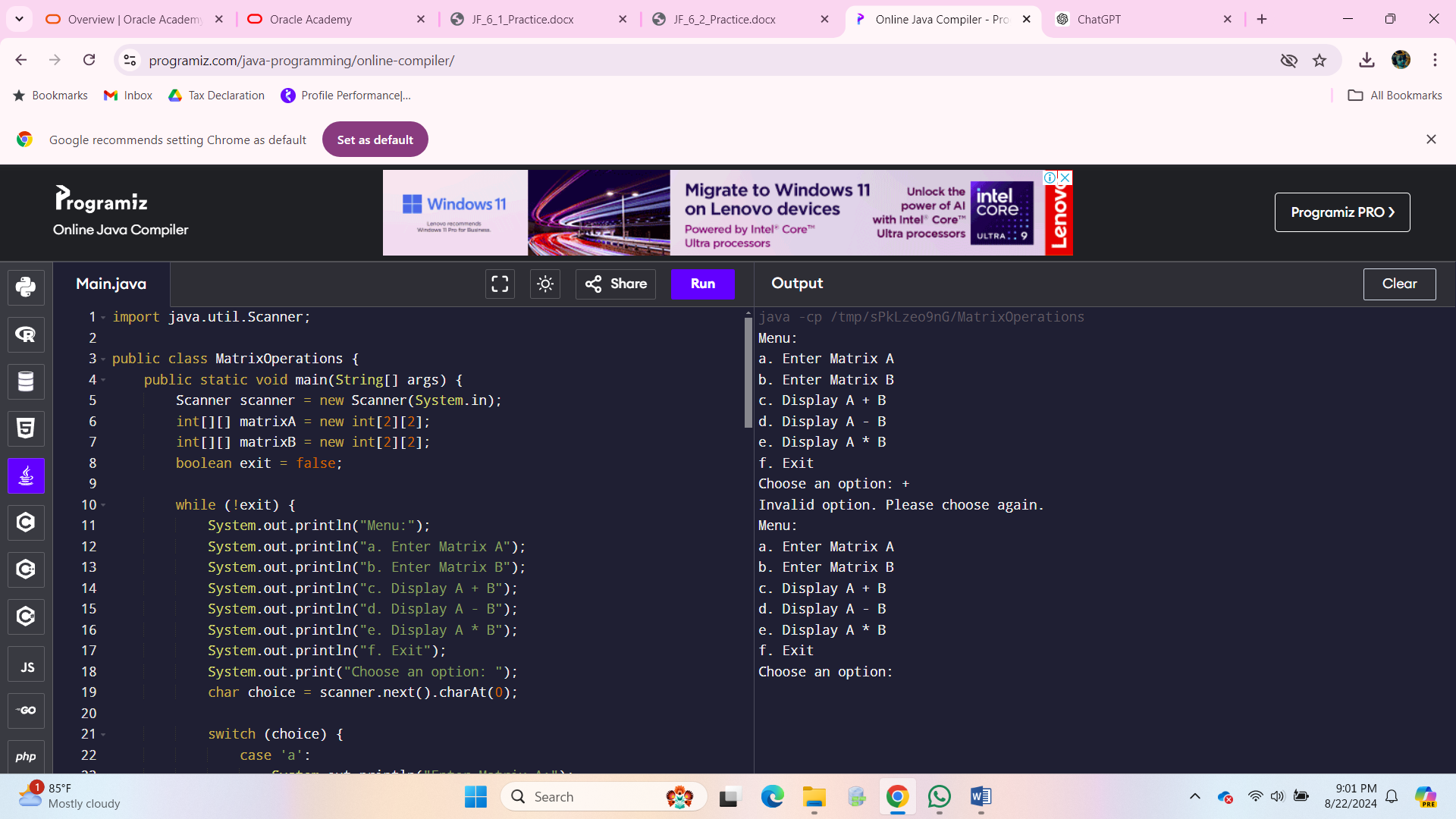
System.out.println();

}

}

}

In Algebra class we learn about matrices. We learn to add, subtract, and multiply 2x2 matrices and 3x3 matrices. Below are some examples from Algebra class with the answers: � 3 4 5 6 �+� 1 0 −2 3 �=� 4 4 3 9 � � 3 4 5 6 �-� 1 0 −2 3 �=� 2 4 7 3 � � 3 4 5 6 �\*� 1 0 −2 3 �=� −5 12 −7 18� Copyright © 2022, Oracle and/or its affiliates. Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. 3 It is almost apparent how to add. We add the first position in the first matrix with the first position in the second matrix. We continue with the corresponding positions to get the answer. Subtraction follows this same positional methods. Multiplication of matrices appears to be confusing since it does not follow the positional method used in addition and subtraction. The answer is achieved by taking the row from the first matrix and the column from the second matrix and multiplying the respective values and then taking the sum of the products. The answer above was achieve as follows: 3(1)+4(-2)=-53(0)+4(3)=12 5(1)+6(-2)=-75(0)+6(3)=18 Write a program that take in two matrices and then allow the user to choose to add, subtract, or multiply them and display the answer. The program will display the following menu: a. Enter Matrix A b. Enter Matrix B c. Display A + B d. Display A - B e. Display A \* B f. Exit The program should loop and allow the user to continue to choose different op



**Deck Class with Shuffle Method:**

The Deck class is enhanced with a shuffle method.

**2. Hand Class:**

A Hand class is created to manage the cards in a hand, including dealing, displaying, and calculating the hand's total points.

**3. Card Class Adjustment:**

The Card class handles the logic for the Ace's dual value, where it starts as 11 but can be reduced to 1 if the total hand value exceeds 21.

**4. Main Class:**

The Main class ties everything together, allowing the user to play a simplified version of Blackjack against the dealer.

Here’s the full code:

java

Copy code

import java.util.\*;

public class Main {

public static void main(String args[]) {

Deck d = new Deck();

d.shuffle();

d.print();

// Create player and dealer hands

Hand playerHand = new Hand();

Hand dealerHand = new Hand();

// Deal two cards to player and dealer

playerHand.addCard(d.deal());

playerHand.addCard(d.deal());

dealerHand.addCard(d.deal());

dealerHand.addCard(d.deal());

// Display player's hand and total points

playerHand.displayHand("Player");

System.out.println("Player's total points: " + playerHand.getPoints());

// Player's turn to hit or stand

Scanner scanner = new Scanner(System.in);

while (playerHand.getCardCount() < 5 && playerHand.getPoints() <= 21) {

System.out.print("Do you want another card? (yes/no): ");

String choice = scanner.next();

if (choice.equalsIgnoreCase("yes")) {

playerHand.addCard(d.deal());

playerHand.displayHand("Player");

System.out.println("Player's total points: " + playerHand.getPoints());

} else {

break;

}

}

// Dealer's turn

dealerHand.displayHand("Dealer");

while (dealerHand.getPoints() < 16) {

dealerHand.addCard(d.deal());

dealerHand.displayHand("Dealer");

System.out.println("Dealer's total points: " + dealerHand.getPoints());

}

// Determine the winner

determineWinner(playerHand, dealerHand);

}

public static void determineWinner(Hand playerHand, Hand dealerHand) {

int playerPoints = playerHand.getPoints();

int dealerPoints = dealerHand.getPoints();

if (playerPoints > 21) {

System.out.println("Player busts! Dealer wins.");

} else if (dealerPoints > 21) {

System.out.println("Dealer busts! Player wins.");

} else if (playerPoints > dealerPoints) {

System.out.println("Player wins!");

} else if (dealerPoints > playerPoints) {

System.out.println("Dealer wins!");

} else {

System.out.println("It's a tie!");

}

}

}

class Deck {

Card[] cardArray = new Card[52];

int currentCard = 0;

Deck() {

int suits = 4;

int cardType = 13;

int cardCount = 0;

for (int i = 1; i <= suits; i++)

for (int j = 1; j <= cardType; j++) {

cardArray[cardCount] = new Card(i, j);

cardCount++;

}

}

public void print() {

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

System.out.println(cardArray[i]);

}

public void shuffle() {

Random rand = new Random();

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

int randomIndex = rand.nextInt(cardArray.length);

Card temp = cardArray[i];

cardArray[i] = cardArray[randomIndex];

cardArray[randomIndex] = temp;

}

currentCard = 0; // Reset to start dealing from the top of the shuffled deck

}

public Card deal() {

if (currentCard < cardArray.length) {

return cardArray[currentCard++];

} else {

System.out.println("No cards left in the deck!");

return null;

}

}

}

class Card {

String suit, name;

int points;

Card(int n1, int n2) {

suit = getSuit(n1);

name = getName(n2);

points = getPoints(name);

}

public String toString() {

return "The " + name + " of " + suit;

}

public String getName(int i) {

if (i == 1) return "Ace";

if (i == 2) return "Two";

if (i == 3) return "Three";

if (i == 4) return "Four";

if (i == 5) return "Five";

if (i == 6) return "Six";

if (i == 7) return "Seven";

if (i == 8) return "Eight";

if (i == 9) return "Nine";

if (i == 10) return "Ten";

if (i == 11) return "Jack";

if (i == 12) return "Queen";

if (i == 13) return "King";

return "error";

}

public int getPoints(String n) {

if (n == "Jack" || n == "Queen" || n == "King" || n == "Ten")

return 10;

if (n == "Two")

return 2;

if (n == "Three")

return 3;

if (n == "Four")

return 4;

if (n == "Five")

return 5;

if (n == "Six")

return 6;

if (n == "Seven")

return 7;

if (n == "Eight")

return 8;

if (n == "Nine")

return 9;

if (n == "Ace")

return 11; // Initially count Ace as 11

return -1;

}

public String getSuit(int i) {

if (i == 1) return "Diamonds";

if (i == 2) return "Clubs";

if (i == 3) return "Spades";

if (i == 4) return "Hearts";

return "error";

}

}

class Hand {

private Card[] hand;

private int cardCount;

private int points;

Hand() {

hand = new Card[5];

cardCount = 0;

points = 0;

}

public void addCard(Card card) {

if (cardCount < hand.length) {

hand[cardCount] = card;

points += card.points;

cardCount++;

adjustForAces(); // Check if points need to be adjusted for Aces

} else {

System.out.println("Cannot add more cards. Hand is full.");

}

}

public int getCardCount() {

return cardCount;

}

public int getPoints() {

return points;

}

public void displayHand(String owner) {

System.out.println(owner + "'s Hand:");

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

System.out.println(hand[i]);

}

}

private void adjustForAces() {

if (points > 21) {

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

if (hand[i].name.equals("Ace") && hand[i].points == 11) {

hand[i].points = 1;

points -= 10;

if (points <= 21) {

break;

}

}

}

}

}

}