

Shaik sumiya

192372090

CSA0961- JAVA PROGRAMMING

6.1 PRACTICE:

- 1. Declare a one dimensional array name score of type int that can hold 9 values.**

Code:

```
int[] score = new int[9];
```

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

Code:

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

- 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.**

Code:

```
long[][] matrix = {  
    {5, 5, 5},  
    {5, 5, 5},  
    {5, 5, 5},  
    {5, 5, 5}  
};
```

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

Code:

```
byte[] values = new byte[10];  
Arrays.fill(values, (byte) 1);
```

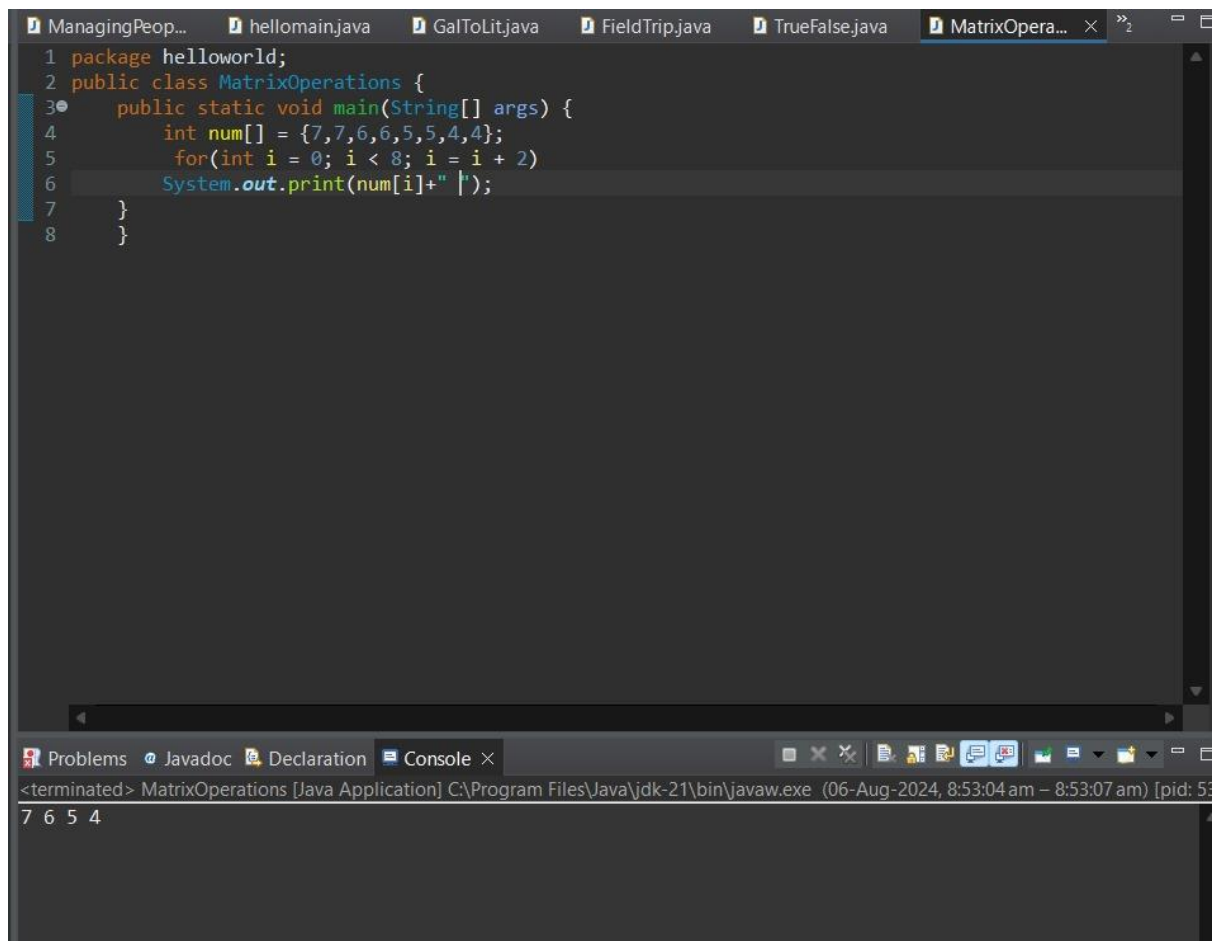
- 5. Without typing in the code determine the output of the following program.**

Code:

```
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 screenshot shows an IDE with a Java file named `MatrixOperations.java`. The code is as follows:

```
1 package helloworld;
2 public class MatrixOperations {
3     public static void main(String[] args) {
4         int num[] = {7,7,6,6,5,5,4,4};
5         for(int i = 0; i < 8; i = i + 2)
6             System.out.print(num[i]+" ");
7     }
8 }
```

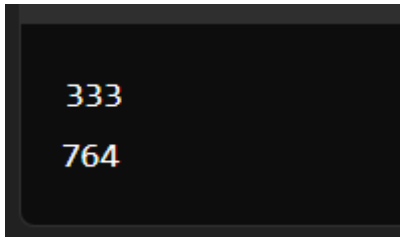
The console output at the bottom shows the result of running the program:

```
<terminated> MatrixOperations [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:53:04 am – 8:53:07 am) [pid: 53]
7 6 5 4
```

6. Without typing in the code determine the output of the following program

Code:

```
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();
}
```



7. 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 students average.

Code:

```
import java.util.Scanner;

public class TestScores {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] scores = new int[5];
        int sum = 0;

        System.out.println("Enter 5 test scores:");
        for (int i = 0; i < 5; i++) {
            scores[i] = scanner.nextInt();
            sum += scores[i];
        }

        double average = sum / 5.0;
        System.out.println("Average score: " + average);
    }
}
```

```

}

package helloworld;
import java.util.Scanner;

public class TestScores {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] scores = new int[5];
        int sum = 0;

        System.out.println("Enter 5 test scores:");
        for (int i = 0; i < 5; i++) {
            scores[i] = scanner.nextInt();
            sum += scores[i];
        }

        double average = sum / 5.0;
        System.out.println("Average score: " + average);
    }
}

```

Problems Javadoc Declaration Console X

<terminated> TestScores [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:46:06 am – 8:46:30 am) [pid: 22792]

Enter 5 test scores:
10
20
15
12
17
Average score: 14.8

8. 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:

Code:

```
import java.util.Scanner;
```

```

public class MatrixOperations {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] matrixA = new int[2][2];
        int[][] matrixB = new int[2][2];
        boolean matricesEntered = false;

        while (true) {
            System.out.println("Menu:");
            System.out.println("a. Enter Matrix A");
            System.out.println("b. Enter Matrix B");
            System.out.println("c. Display A + B");
            System.out.println("d. Display A - B");

```

```

System.out.println("e. Display A * B");
System.out.println("f. Exit");
char choice = scanner.next().charAt(0);

switch (choice) {
    case 'a':
        System.out.println("Enter Matrix A:");
        matrixA = enterMatrix(scanner);
        matricesEntered = true;
        break;
    case 'b':
        System.out.println("Enter Matrix B:");
        matrixB = enterMatrix(scanner);
        matricesEntered = true;
        break;
    case 'c':
        if (matricesEntered) {
            System.out.println("A + B:");
            displayMatrix(addMatrices(matrixA, matrixB));
        } else {
            System.out.println("Enter both matrices first.");
        }
        break;
    case 'd':
        if (matricesEntered) {
            System.out.println("A - B:");
            displayMatrix(subtractMatrices(matrixA, matrixB));
        } else {
            System.out.println("Enter both matrices first.");
        }
        break;
    case 'e':
        if (matricesEntered) {
            System.out.println("A * B:");
            displayMatrix(multiplyMatrices(matrixA, matrixB));
        } else {
            System.out.println("Enter both matrices first.");
        }
        break;
    case 'f':
        System.exit(0);
        break;
    default:
        System.out.println("Invalid choice.");
        break;
}
}
}

```

```

private static int[][] enterMatrix(Scanner scanner) {
    int[][] matrix = new int[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            matrix[i][j] = scanner.nextInt();
        }
    }
    return matrix;
}

```

```

private static void displayMatrix(int[][] matrix) {
    for (int i = 0; i < matrix.length; i++) {
        for (int j = 0; j < matrix[0].length; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}

```

```

private static int[][] addMatrices(int[][] matrixA, int[][] matrixB) {
    int[][] result = new int[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            result[i][j] = matrixA[i][j] + matrixB[i][j];
        }
    }
    return result;
}

```

```

private static int[][] subtractMatrices(int[][] matrixA, int[][] matrixB) {
    int[][] result = new int[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            result[i][j] = matrixA[i][j] - matrixB[i][j];
        }
    }
    return result;
}

```

```

private static int[][] multiplyMatrices(int[][] matrixA, int[][] matrixB) {
    int[][] result = new int[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            result[i][j] = matrixA[i][0] * matrixB[0][j] + matrixA[i][1] * matrixB[1][j];
        }
    }
    return result;
}

```

```

    }
}

<terminated> MatrixOperations [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (06-Aug-2024, 8:54:04 am – 8:55:15 am) [pid: 229]
Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
a
Enter Matrix A:
2
5
6
4
Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
b
Enter Matrix B:
4
2
1
6
Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
c
A + B:
6 7
7 10

```

9. Use the following code to implement a Deck Object. When finished, add the following features:

- Add a method shuffle to the Deck Class. Call the method from the Main class to verify that the deck is indeed shuffled.
- Add a Hand Class that contains an array of 5 Card references. Have the program Deal the Hand two cards and display them for the user. Tell the user how many points they have and ask them if they would like another card or not. Continue to allow the player to add cards until they reach 5 cards, or the total is greater than 21.
- Adjust the Card class to allow Aces to count as 11 to start. If the Hand Class has a value greater than 21, have the Hand Class check for Aces and reduce their point value to 1.
- Have the program create a dealer Hand that the user can play against. The user should try to get as close to 21 without going over in an effort to beat the Dealer. If the Dealer has 16 or more the Dealer should stop taking cards.

Code:

```

import java.util.Arrays;
import java.util.Collections;
import java.util.Scanner;

```

```

public class Main {

```

```

public static void main(String[] args) {
    Deck d = new Deck();
    d.shuffle();
    d.print();

    Hand playerHand = new Hand();
    Hand dealerHand = new Hand();

    // Deal two cards to player and dealer
    playerHand.addCard(d.dealCard());
    playerHand.addCard(d.dealCard());
    dealerHand.addCard(d.dealCard());
    dealerHand.addCard(d.dealCard());

    Scanner scanner = new Scanner(System.in);
    boolean playerTurn = true;

    while (playerTurn) {
        System.out.println("Player's hand: " + playerHand);
        System.out.println("Total points: " + playerHand.getPoints());
        if (playerHand.getPoints() > 21) {
            System.out.println("Player busts! Dealer wins.");
            return;
        }
        System.out.print("Would you like another card? (yes/no): ");
        String response = scanner.nextLine();
        if (response.equalsIgnoreCase("yes")) {
            playerHand.addCard(d.dealCard());
        } else {
            playerTurn = false;
        }
    }

    while (dealerHand.getPoints() < 16) {
        dealerHand.addCard(d.dealCard());
    }

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

    if (dealerHand.getPoints() > 21 || playerHand.getPoints() > dealerHand.getPoints()) {
        System.out.println("Player wins!");
    } else if (playerHand.getPoints() < dealerHand.getPoints()) {
        System.out.println("Dealer wins!");
    } else {
        System.out.println("It's a tie!");
    }
}

```



```

}

class Deck {
    Card[] cardArray = new Card[52];
    int topCardIndex = 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 shuffle() {
        Collections.shuffle(Arrays.asList(cardArray));
        topCardIndex = 0;
    }

    public Card dealCard() {
        if (topCardIndex < cardArray.length) {
            return cardArray[topCardIndex++];
        }
        return null; // No cards left
    }

    public void print() {
        for (Card card : cardArray) {
            System.out.println(card);
        }
    }
}

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;  
}
```

```
private 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";  
}
```

```
private int getPoints(String n) {  
    if (n.equals("Jack") || n.equals("Queen") || n.equals("King") || n.equals("Ten"))  
        return 10;  
    if (n.equals("Two"))  
        return 2;  
    if (n.equals("Three"))  
        return 3;  
    if (n.equals("Four"))  
        return 4;  
    if (n.equals("Five"))  
        return 5;  
    if (n.equals("Six"))  
        return 6;  
    if (n.equals("Seven"))  
        return 7;  
    if (n.equals("Eight"))  
        return 8;  
    if (n.equals("Nine"))  
        return 9;  
    if (n.equals("Ace"))  
        return 11;  
    return -1;  
}
```

```
private 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[] cards = new Card[5];
    private int cardCount = 0;

    public void addCard(Card card) {
        if (cardCount < cards.length) {
            cards[cardCount++] = card;
        }
    }

    public int getPoints() {
        int totalPoints = 0;
        int acesCount = 0;

        for (int i = 0; i < cardCount; i++) {
            totalPoints += cards[i].points;
            if (cards[i].name.equals("Ace")) {
                acesCount++;
            }
        }

        while (totalPoints > 21 && acesCount > 0) {
            totalPoints -= 10;
            acesCount--;
        }

        return totalPoints;
    }

    public String toString() {
        StringBuilder handString = new StringBuilder();
        for (int i = 0; i < cardCount; i++) {
            handString.append(cards[i]).append("\n");
        }
        return handString.toString();
    }
}

```

```
Player's hand: The Ace of Hearts  
The Four of Diamonds  
  
Total points: 15  
Would you like another card? (yes/no): yes  
Player's hand: The Ace of Hearts  
The Four of Diamonds  
The Five of Spades  
  
Total points: 20  
Would you like another card? (yes/no): no  
Dealer's hand: The Four of Clubs  
The Nine of Clubs  
The King of Diamonds  
  
Total points: 23|  
Player wins!  
}
```