JAVA FUNDAMENTALS 6.1 PRACTICE:

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

Solution:

The one dimensional array name score of type int that can hold 9 values is:

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

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

Solution:

The 2-dimensional array named price of type float that has 10 rows and 3 columns is:

```
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.

Solution:

The 2-dimensional array named matrix of type long that has 4 rows and 3 columns to have all it's values set to 5 is:

```
long[][] matrix = new long[4][3];
for (int i = 0; i < 4; i++) {
   for (int j = 0; j < 3; j++) {
      matrix[i][j] = 5;
   }
}
```

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

Solution:

The one dimensional byte array named values of size 10 is:

```
import java.util.Arrays;
public class Main {
   public static void main(String[] args) {
      byte[] values = new byte[10];
      Arrays.fill(values, (byte) 1);
      for (byte value : values) {
         System.out.print(value + " ");
      }
   }
}
```

5. 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]);
```

Solution:

The output of the following program is: 7654

6. 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();
}
```

Solution:

The output of the following program is:
333
765

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.

Solution:

```java
import java.util.Scanner;

public class TestScores {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] testScores = new int[5];
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter score for test " + (i + 1) + ": ");
            testScores[i] = scanner.nextInt();
        }
        int sum = 0;
        for (int i = 0; i < 5; i++) {
            sum += testScores[i];
        }
        double average = sum / 5.0;
        System.out.println("The average score is: " + average);
        scanner.close();
    }
}
```

Output:

```
Enter score for test 1: 18
Enter score for test 2: 19
Enter score for test 3: 20
Enter score for test 4: 20
Enter score for test 5: 20
The average score is: 19.4

=== Code Execution Successful ===
```

8. In Algebra class we learn about matrices. We learn to add, subtract, and multiply 2x2 matrices and 3x3 matrices.
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 options until they choose quit. The well written program will modularize the process into different methods.

Solution:

```java
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 exit = false;

        while (!exit) {
            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");
            System.out.print("Choose an option: ");
            char choice = scanner.next().charAt(0);

            switch (choice) {
                case 'a':
                    System.out.println("Enter Matrix A:");
                    matrixA = inputMatrix(scanner);
                    break;
                case 'b':
                    System.out.println("Enter Matrix B:");
                    matrixB = inputMatrix(scanner);
                    break;
                case 'c':
                    displayMatrix(addMatrices(matrixA, matrixB));
                    break;
                case 'd':
                    displayMatrix(subtractMatrices(matrixA, matrixB));
                    break;
                case 'e':
                    displayMatrix(multiplyMatrices(matrixA, matrixB));
                    break;
                case 'f':
                    exit = true;
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }

        scanner.close();
    }
    // Method to input a 2x2 matrix
    public static int[][] inputMatrix(Scanner scanner) {
        int[][] matrix = new int[2][2];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("Enter value for position [" + i + "][" + j + "]: ");
                matrix[i][j] = scanner.nextInt();
            }
        }
        return matrix;
    }
    // Method to add two matrices
    public static int[][] addMatrices(int[][] a, int[][] b) {
        int[][] result = new int[2][2];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                result[i][j] = a[i][j] + b[i][j];
            }
        }
        return result;
    }
    // Method to subtract two matrices
    public static int[][] subtractMatrices(int[][] a, int[][] b) {
        int[][] result = new int[2][2];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                result[i][j] = a[i][j] - b[i][j];
            }
        }
        return result;
    }
    // Method to multiply two matrices
    public static int[][] multiplyMatrices(int[][] a, int[][] b) {
        int[][] result = new int[2][2];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                result[i][j] = a[i][0] * b[0][j] + a[i][1] * b[1][j];
            }
        }
        return result;
    }
    // Method to display a matrix
    public static void displayMatrix(int[][] matrix) {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
Choose an option:
=== Session Ended. Please Run the code again ===
```

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

a. Add a method shuffle to the Deck Class. Call the method from the Main class to verify that the deck is indeed shuffled.

b. 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.

c. 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.

d. 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.

A. CARD CLASS:

```
public 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) {
        switch (i) {
            case 1: return "Ace";
            case 2: return "Two";
            case 3: return "Three";
            case 4: return "Four";
            case 5: return "Five";
            case 6: return "Six";
            case 7: return "Seven";
            case 8: return "Eight";
            case 9: return "Nine";
            case 10: return "Ten";
            case 11: return "Jack";
            case 12: return "Queen";
            case 13: return "King";
            default: return "error";
        }
    }
    public int getPoints(String n) {
        if (n.equals("Jack") || n.equals("Queen") || n.equals("King") ||
n.equals("Ten"))
            return 10;
        if (n.equals("Ace"))
            return 11;
        return Integer.parseInt(n);
    }
    public String getSuit(int i) {
        switch (i) {
            case 1: return "Diamonds";
```

```
            case 2: return "Clubs";
            case 3: return "Spades";
            case 4: return "Hearts";
            default: return "error";
        }
    }
}
```

B. DECK CLASS:

```
import java.util.Collections;
import java.util.List;
import java.util.ArrayList;
public class Deck {
    Card[] cardArray = new Card[52];
    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() {
        List<Card> cardList = new
ArrayList<Card>(List.of(cardArray));
        Collections.shuffle(cardList);
        cardArray = cardList.toArray(new Card[0]);
    }
}
```

C. HAND CLASS:

```
import java.util.ArrayList;
public class Hand {
    private ArrayList<Card> hand;
    private int handValue;
    public Hand() {
        hand = new ArrayList<>();
        handValue = 0;
    }
    public void addCard(Card card) {
        hand.add(card);
        handValue += card.points;
        adjustForAces();
    }
    private void adjustForAces() {
        // Adjust Ace values if total is over 21
        while (handValue > 21 && hasAce()) {
            handValue -= 10; // Reduce Ace from 11 to 1
        }
    }
    private boolean hasAce() {
        for (Card card : hand) {
            if (card.name.equals("Ace") && card.points == 11) {
                return true;
            }
        }
        return false;
    }
    public int getValue() {
        return handValue;
    }
    public void display() {
        for (Card card : hand) {
            System.out.println(card);
```

```
        }
        System.out.println("Hand value: " + handValue);
    }
}
```

D. MAIN CLASS:

```
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        Deck deck = new Deck();
        deck.shuffle(); // Shuffle the deck
        Hand playerHand = new Hand();
        Hand dealerHand = new Hand();
        System.out.println("Dealing two cards to the player...");
        playerHand.addCard(deck.cardArray[0]);
        playerHand.addCard(deck.cardArray[1]);
        dealerHand.addCard(deck.cardArray[2]);
        dealerHand.addCard(deck.cardArray[3]);
        boolean playerBusted = false;
        while (playerHand.getValue() <= 21) {
            System.out.println("Player's hand:");
            playerHand.display();
            System.out.print("Do you want another card (y/n)? ");
            char choice = scanner.next().charAt(0);
            if (choice == 'y' || choice == 'Y') {
                playerHand.addCard(deck.cardArray[4]);
                System.out.println("Dealing another card...");
            } else {
                break;
            }
        }
        if (playerHand.getValue() > 21) {
            playerBusted = true;
            System.out.println("Player busts!");
```

```
        }
        while (dealerHand.getValue() < 17) {
            dealerHand.addCard(deck.cardArray[5]);
        }
        System.out.println("Dealer's hand:");
        dealerHand.display();
        if (!playerBusted) {
            if (playerHand.getValue() > dealerHand.getValue() ||
dealerHand.getValue() > 21) {
                System.out.println("Player wins!");
            } else if (playerHand.getValue() < dealerHand.getValue()) {
                System.out.println("Dealer wins!");
            } else {
                System.out.println("It's a tie!");
            }
        }
        scanner.close();
    }
}
```