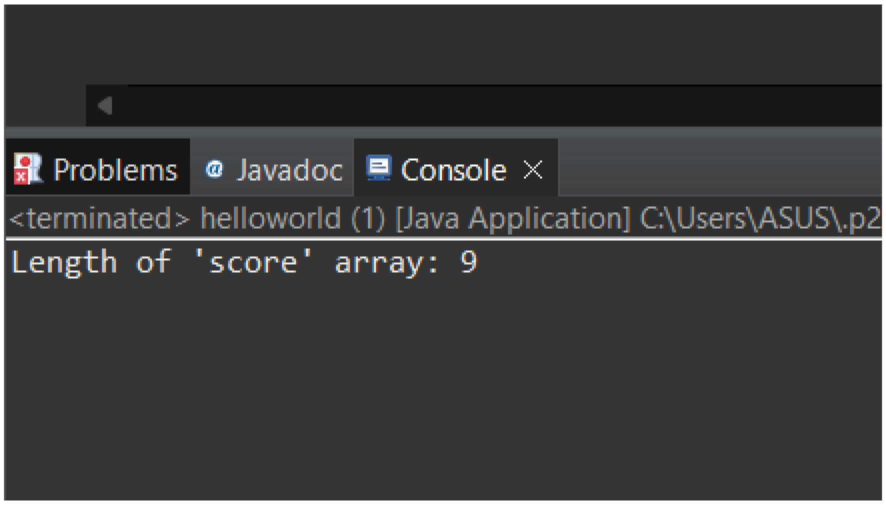
Java 6_1

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

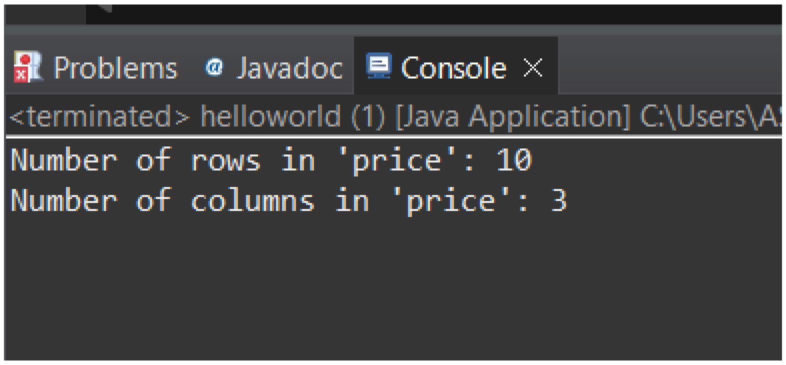
```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

   public static void main(String[] args) {

              package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

   public static void main(String[] args) {

              int[] score = new int[9];

       System.out.println("Length of 'score' array: " + score.length);

       // Output: Length of 'score' array: 9

   }

}
```



```
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.p2
Length of 'score' array: 9
```

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

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

    public static void main(String[] args) {

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

        System.out.println("Number of rows in 'price': " + price.length);

        System.out.println("Number of columns in 'price': " + price[0].length);

        // Output: Number of rows in 'price': 10

        //       Number of columns in 'price': 3

    }

}
```

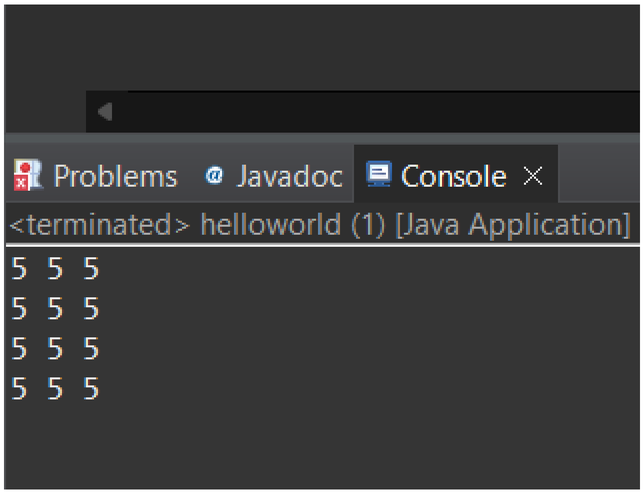


```
<terminated> helloworld (1) [Java Application] C:\Users\A
Number of rows in 'price': 10
Number of columns in 'price': 3
```

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.

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

  public static void main(String[] args) {

        long[][] matrix = new long[4][3];

    for (int row = 0; row < 4; row++) {

      for (int col = 0; col < 3; col++) {

        matrix[row][col] = 5L;

      }

    }

    // Let's print the matrix (just for fun):

    for (int row = 0; row < 4; row++) {

      for (int col = 0; col < 3; col++) {

        System.out.print(matrix[row][col] + " ");

      }

      System.out.println();

    }
```

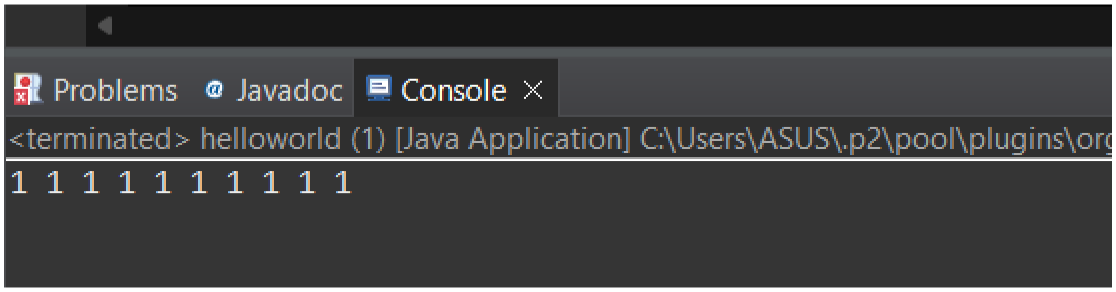
```
        // Output (matrix):
        // 5 5 5
        // 5 5 5
        // 5 5 5
        // 5 5 5
    }
}
```



```
<terminated> helloworld (1) [Java Application]
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.

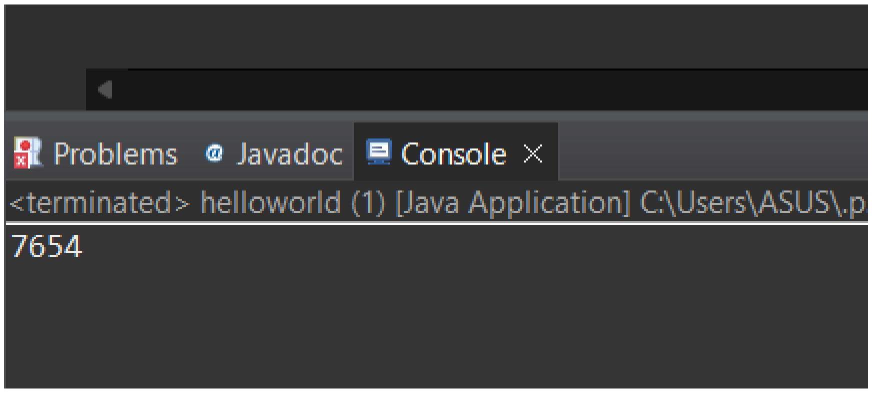
```
package helloworld;


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


public class helloworld {
    public static void main(String[] args) {
            byte[] values = new byte[10];
        Arrays.fill(values, (byte) 1);
        // Let's print the 'values' array:
        for (byte val : values) {
            System.out.print(val + " ");
        }
        // Output: 1 1 1 1 1 1 1 1 1 1
    }
}
```



```
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.p2\pool\plugins\or
1 1 1 1 1 1 1 1 1 1
```

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]);

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

  public static void main(String[] args) {

        int[] num = {7, 7, 6, 6, 5, 5, 4, 4};

     for (int i = 0; i < 8; i = i + 2) {

       System.out.print(num[i]);

     }

     // Output: 7654

   }

 }
```

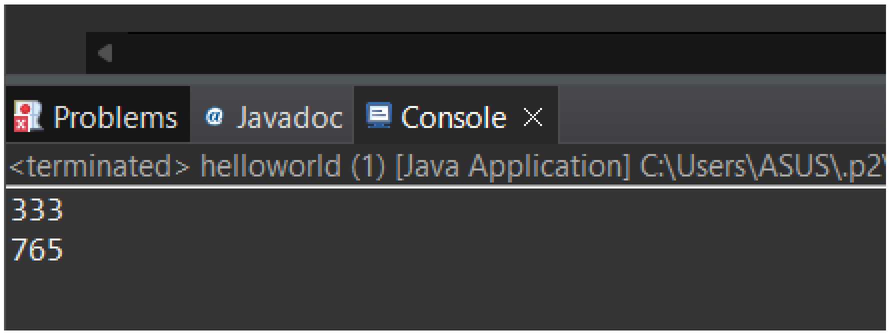


```
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.p
7654
```

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(); }

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

    public static void main(String[] args) {

            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();

            }
```

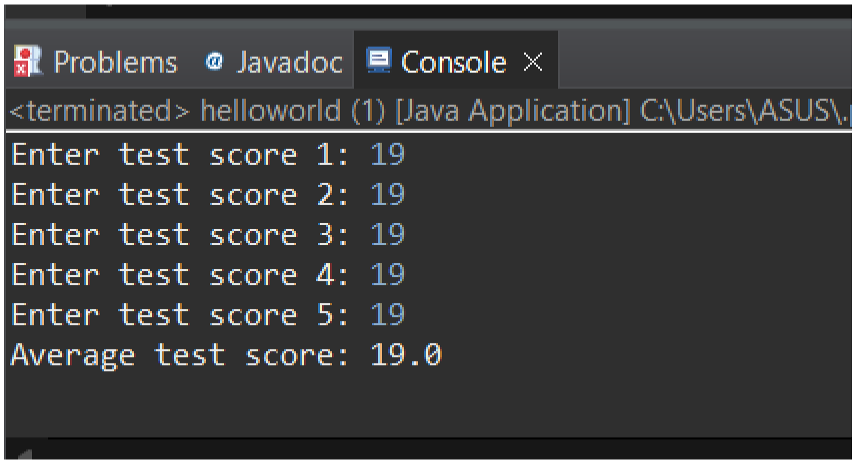
```
        // Output: 7654
    }
}
```



```
Problems  Javadoc  Console
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.p2
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.

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

    public static void main(String[] args) {

              Scanner scanner = new Scanner(System.in);


        // Create an array to store test scores

        double[] testScores = new double[5];


        // Input test scores

        for (int i = 0; i < 5; i++) {
            System.out.print("Enter test score " + (i + 1) + ": ");
            testScores[i] = scanner.nextDouble();
        }

        // Calculate the sum of test scores
        double sum = 0;
        for (double score : testScores) {
            sum += score;
        }

        // Calculate the average
        double average = sum / 5;

        System.out.println("Average test score: " + average);

        scanner.close();
    }
}
```



Problems @ Javadoc Console

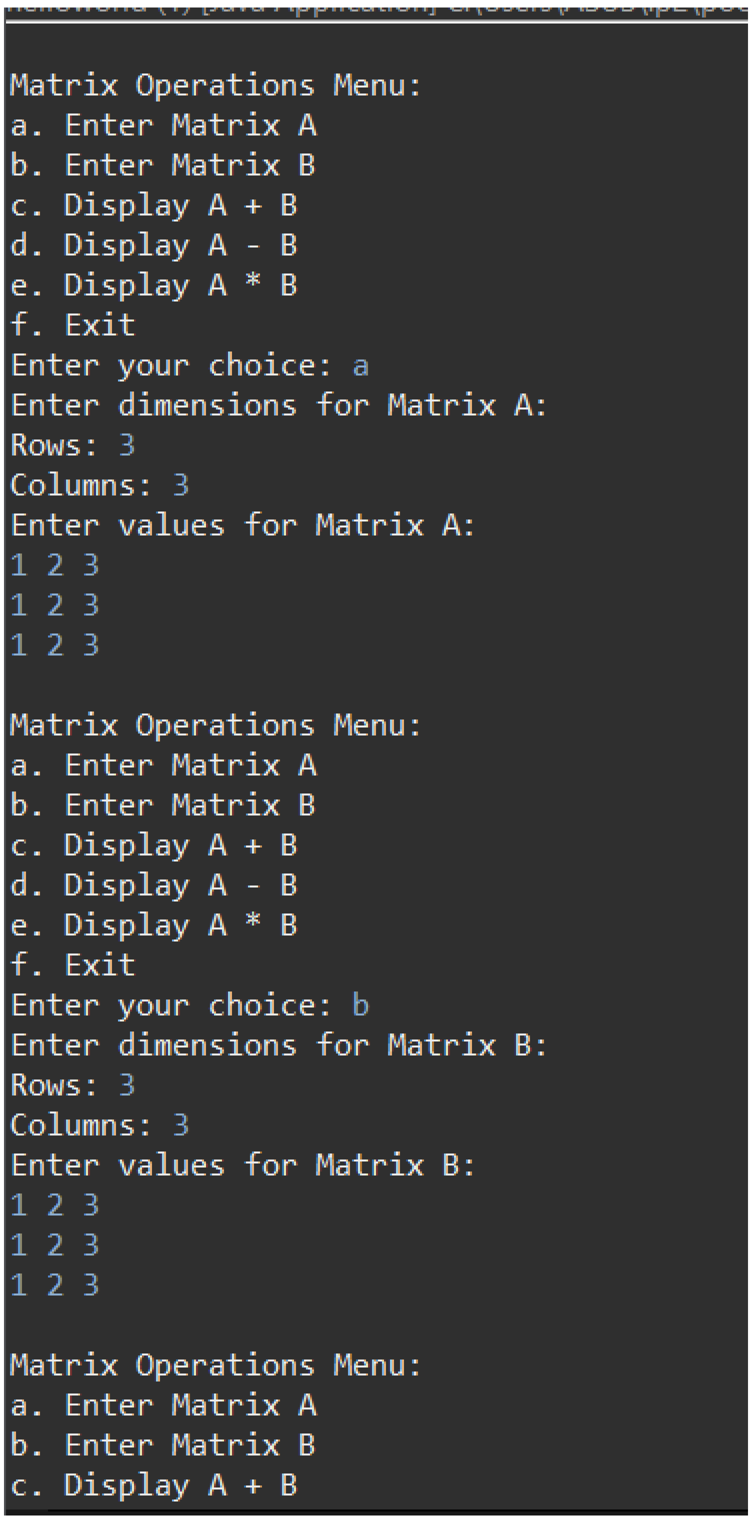
```
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.
Enter test score 1: 19
Enter test score 2: 19
Enter test score 3: 19
Enter test score 4: 19
Enter test score 5: 19
Average test score: 19.0
```

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: � 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�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

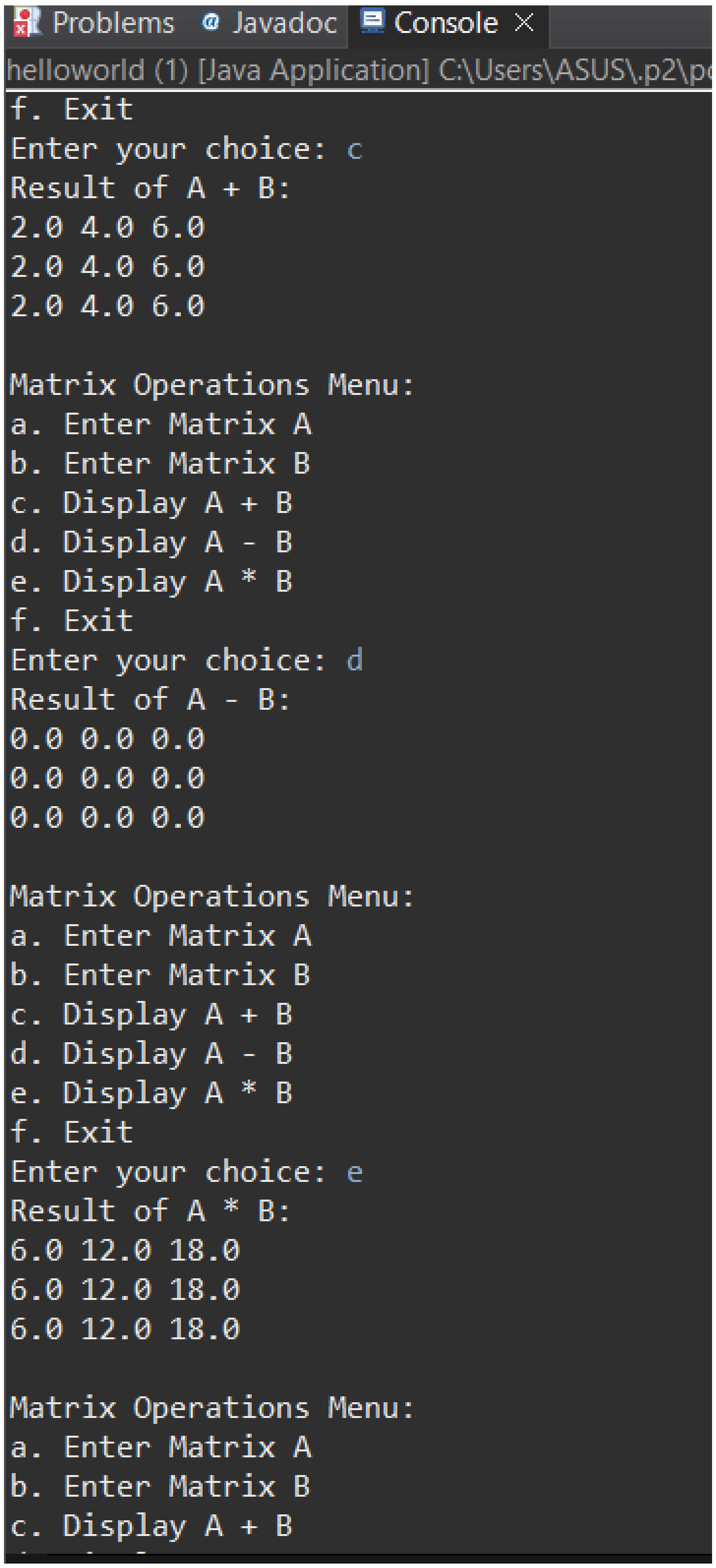
continue with the corresponding positions to get the answer. Subtraction follows this same positional methods. Multiplication of matrices appears to be confusing since it does not follow the positional method used in addition and subtraction. The answer is achieved by taking the row from the first matrix 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 options until they choose quit. The well written program will modularize the process into different methods.

```
package helloworld;


import java.util.Arrays;

import java.util.Scanner;


public class helloworld {

  private static Scanner scanner = new Scanner(System.in);

  private static double[][] matrixA;

  private static double[][] matrixB;


  public static void main(String[] args) {

    displayMenu();

  }


  private static void displayMenu() {

    while (true) {

      System.out.println("\nMatrix Operations 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("Enter your choice: ");
    char option = scanner.next().charAt(0);

    switch (option) {
      case 'a':
        matrixA = enterMatrix("A");
        break;
      case 'b':
        matrixB = enterMatrix("B");
        break;
      case 'c':
        displayResult(addMatrices(matrixA, matrixB), "+");
        break;
      case 'd':
        displayResult(subtractMatrices(matrixA, matrixB), "-");
        break;
      case 'e':
        displayResult(multiplyMatrices(matrixA, matrixB), "*");
        break;
      case 'f':
        System.out.println("Exiting. Fare thee well!");
        scanner.close();
        System.exit(0);
      default:
        System.out.println("Invalid choice. Try again.");
    }
  }
```

```
}

private static double[][] enterMatrix(String name) {
    System.out.println("Enter dimensions for Matrix " + name + ":");
    System.out.print("Rows: ");
    int rows = scanner.nextInt();
    System.out.print("Columns: ");
    int cols = scanner.nextInt();

    double[][] matrix = new double[rows][cols];
    System.out.println("Enter values for Matrix " + name + ":");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = scanner.nextDouble();
        }
    }
    return matrix;
}

private static double[][] addMatrices(double[][] A, double[][] B) {
    int rows = A.length;
    int cols = A[0].length;
    double[][] result = new double[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = A[i][j] + B[i][j];
        }
    }
    return result;
}

private static double[][] subtractMatrices(double[][] A, double[][] B) {
    int rows = A.length;
    int cols = A[0].length;
    double[][] result = new double[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = A[i][j] - B[i][j];
        }
    }
    return result;
}

private static double[][] multiplyMatrices(double[][] A, double[][] B) {
    int rowsA = A.length;
    int colsA = A[0].length;
    int colsB = B[0].length;
    double[][] result = new double[rowsA][colsB];

    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            double sum = 0;
            for (int k = 0; k < colsA; k++) {
                sum += A[i][k] * B[k][j];
            }
            result[i][j] = sum;
            }
        }
        return result;
    }

    private static void displayResult(double[][] result, String operation) {
        System.out.println("Result of A " + operation + " B:");
        for (double[] row : result) {
            for (double val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }
}
```



```
Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
Enter your choice: a
Enter dimensions for Matrix A:
Rows: 3
Columns: 3
Enter values for Matrix A:
1 2 3
1 2 3
1 2 3

Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
Enter your choice: b
Enter dimensions for Matrix B:
Rows: 3
Columns: 3
Enter values for Matrix B:
1 2 3
1 2 3
1 2 3

Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
```



```
f. Exit
Enter your choice: c
Result of A + B:
2.0 4.0 6.0
2.0 4.0 6.0
2.0 4.0 6.0

Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
Enter your choice: d
Result of A - B:
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
d. Display A - B
e. Display A * B
f. Exit
Enter your choice: e
Result of A * B:
6.0 12.0 18.0
6.0 12.0 18.0
6.0 12.0 18.0

Matrix Operations Menu:
a. Enter Matrix A
b. Enter Matrix B
c. Display A + B
```

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. public class Main { public static void main(String args[]){ Deck d = new Deck(); d.print(); } } public class Deck { Card[] cardArray = new Card[52]; Deck(){ //constructor 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 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 1; 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"; } }

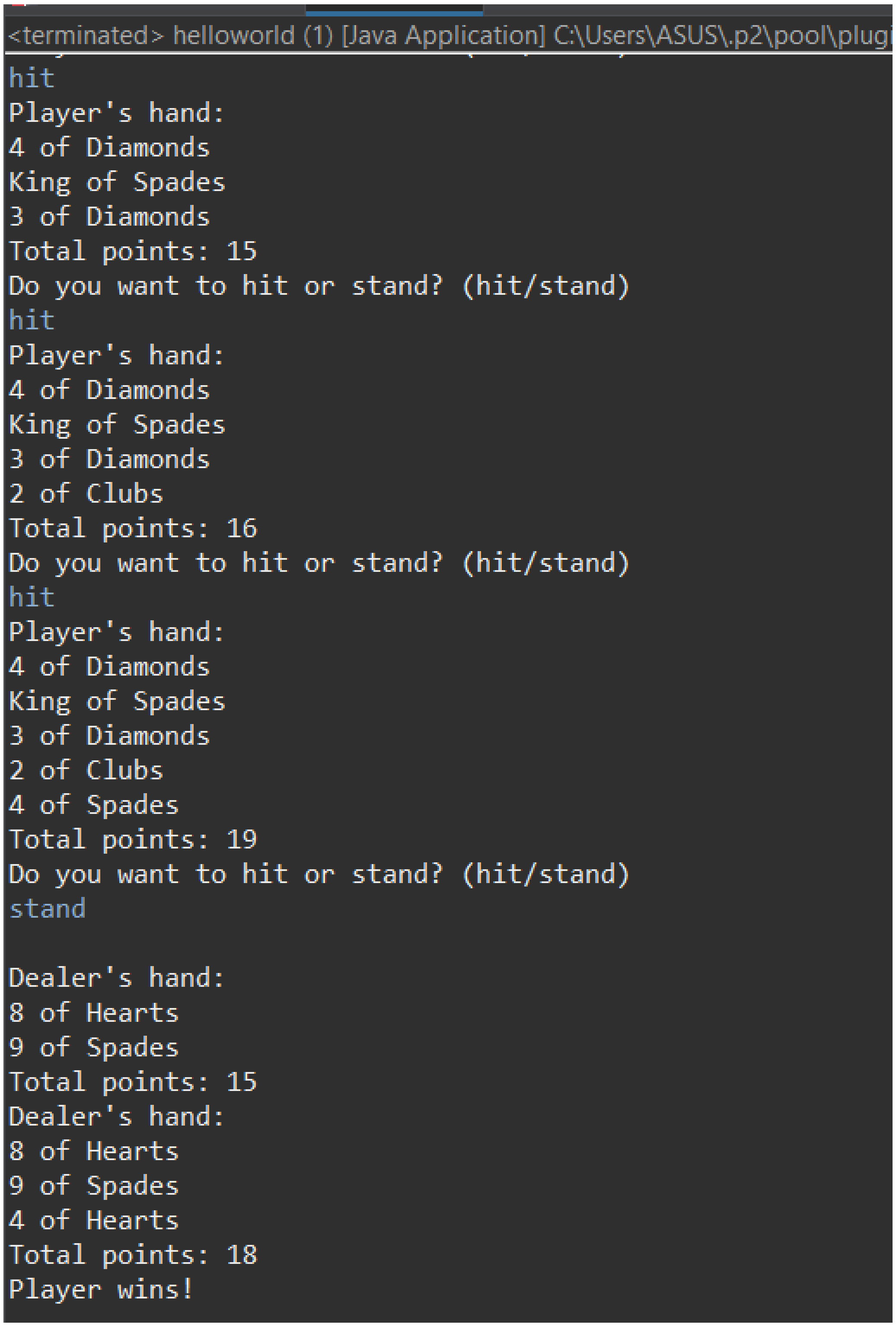
```
package helloworld;


import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Scanner;


public class helloworld {

    public static void main(String[] args) {

        Deck deck = new Deck();

        deck.shuffle();


        Hand playerHand = new Hand();

        Hand dealerHand = new Hand();
```

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

System.out.println("Player's hand:");
playerHand.displayHand();
System.out.println("Total points: " + playerHand.getTotalPoints());

// Implement player's turn (hit or stand)
Scanner scanner = new Scanner(System.in);
boolean playerTurn = true;
while (playerTurn) {
  System.out.println("Do you want to hit or stand? (hit/stand)");
  String decision = scanner.nextLine();
  if (decision.equalsIgnoreCase("hit")) {
    playerHand.addCard(deck.dealCard());
    System.out.println("Player's hand:");
    playerHand.displayHand();
    System.out.println("Total points: " + playerHand.getTotalPoints());
    if (playerHand.getTotalPoints() > 21) {
      System.out.println("Player busts! Dealer wins.");
      return;
    }
  } else {
    playerTurn = false;
  }
}
```

```
        System.out.println("\nDealer's hand:");
        dealerHand.displayHand();
        System.out.println("Total points: " + dealerHand.getTotalPoints());

        // Implement dealer's turn (hit or stand)
        while (dealerHand.getTotalPoints() < 17) {
            dealerHand.addCard(deck.dealCard());
            System.out.println("Dealer's hand:");
            dealerHand.displayHand();
            System.out.println("Total points: " + dealerHand.getTotalPoints());
            if (dealerHand.getTotalPoints() > 21) {
                System.out.println("Dealer busts! Player wins.");
                return;
            }
        }

        // Determine the winner
        int playerPoints = playerHand.getTotalPoints();
        int dealerPoints = dealerHand.getTotalPoints();
        if (playerPoints > dealerPoints) {
            System.out.println("Player wins!");
        } else if (playerPoints < dealerPoints) {
            System.out.println("Dealer wins!");
        } else {
            System.out.println("It's a tie!");
        }
    }
}
```

```
class Deck {
    private List<Card> cardList;

    Deck() {
        cardList = new ArrayList<>();
        for (int suit = 1; suit <= 4; suit++) {
            for (int rank = 1; rank <= 13; rank++) {
                cardList.add(new Card(suit, rank));
            }
        }
    }

    void shuffle() {
        Collections.shuffle(cardList);
    }

    Card dealCard() {
        if (cardList.isEmpty()) {
            throw new IllegalStateException("Deck is empty!");
        }
        return cardList.remove(0);
    }
}

class Card {
    private int suit;
    private int rank;
    Card(int suit, int rank) {
        this.suit = suit;
        this.rank = rank;
    }

    public int getRank() {
        return rank;
    }

    @Override
    public String toString() {
        String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};
        String[] ranks = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};
        return ranks[rank - 1] + " of " + suits[suit - 1];
    }
}

class Hand {
    private List<Card> cards;

    Hand() {
        cards = new ArrayList<>();
    }

    void addCard(Card card) {
        cards.add(card);
    }

    int getTotalPoints() {
        int totalPoints = 0;
        int aceCount = 0;

        for (Card card : cards) {
            int rank = card.getRank();
            if (rank >= 2 && rank <= 10) {
                totalPoints += rank;
            } else if (rank >= 11 && rank <= 13) {
                totalPoints += 10;
            } else if (rank == 1) { // Ace
                aceCount++;
                totalPoints += 11;
            }
        }

        // Adjust for Aces if total points exceed 21
        while (totalPoints > 21 && aceCount > 0) {
            totalPoints -= 10;
            aceCount--;
        }

        return totalPoints;
    }

    void displayHand() {
        for (Card card : cards) {
            System.out.println(card);
        }
    }
```

```
}
```



```
<terminated> helloworld (1) [Java Application] C:\Users\ASUS\.p2\pool\plugi
hit
Player's hand:
4 of Diamonds
King of Spades
3 of Diamonds
Total points: 15
Do you want to hit or stand? (hit/stand)
hit
Player's hand:
4 of Diamonds
King of Spades
3 of Diamonds
2 of Clubs
Total points: 16
Do you want to hit or stand? (hit/stand)
hit
Player's hand:
4 of Diamonds
King of Spades
3 of Diamonds
2 of Clubs
4 of Spades
Total points: 19
Do you want to hit or stand? (hit/stand)
stand

Dealer's hand:
8 of Hearts
9 of Spades
Total points: 15
Dealer's hand:
8 of Hearts
9 of Spades
4 of Hearts
Total points: 18
Player wins!
```