Week 5 - Level 3 - 13 Practice Problems

QUES1

```
import java.util.Random;
class FootballTeam {
    public static int findSum(int[] heights) {
        int sum = 0;
        for (int height : heights) {
            sum += height;
        }
        return sum;
    }
    public static double findMean(int[] heights) {
        int sum = findSum(heights);
        return (double) sum / heights.length;
    }
    public static int findShortest(int[] heights) {
        int shortest = heights[0];
        for (int height : heights) {
            if (height < shortest) {
                shortest = height;
            }
        }
        return shortest;
    }
    public static int findTallest(int[] heights) {
        int tallest = heights[0];
        for (int height : heights) {
            if (height > tallest) {
                tallest = height;
            }
        }
        return tallest;
    }
    public static void main(String[] args) {
        int[] heights = new int[11];
        Random rand = new Random();
        for (int i = 0; i < heights.length; i++) {
            heights[i] = rand.nextInt(101) + 150;
        }
        System.out.println("Heights of the players:");
        for (int height : heights) {
            System.out.println(height + " cm");
        }
        int sum = findSum(heights);
        double mean = findMean(heights);
        int shortest = findShortest(heights);
        int tallest = findTallest(heights);
        System.out.println("\nResults:");
        System.out.println("Sum of heights: " + sum + " cm");
        System.out.println("Mean height: " + mean + " cm");
        System.out.println("Shortest height: " + shortest + " cm");
        System.out.println("Tallest height: " + tallest + " cm");
    }
}
```

OUTPUT:

```
Heights of the players:
223 cm
168 cm
191 cm
245 cm
197 cm
161 cm
202 cm
204 cm
252 cm
210 cm
230 cm

Results:
Sum of heights: 2223 cm
Mean height: 202.09090909090907 cm
Shortest height: 161 cm
Tallest height: 252 cm
```

QUES 2

```
import java.util.ArrayList;
import java.util.List;
class NumberChecker {
```

```
public static int countDigits(int number) {
    int count = 0;
    while (number != 0) {
        number /= 10;
        count++;
    }
    return count;
}
public static int[] storeDigits(int number) {
    int digitCount = countDigits(number);
    int[] digits = new int[digitCount];
    int index = 0;
    while (number != 0) {
        digits[index++] = number % 10;
        number /= 10;
    }
    for (int i = 0; i < digitCount / 2; i++) {
        int temp = digits[i];
        digits[i] = digits[digitCount - 1 - i];
        digits[digitCount - 1 - i] = temp;
    }
    return digits;
}
public static boolean isDuckNumber(int number) {
    int[] digits = storeDigits(number);
    for (int digit : digits) {
        if (digit != 0) {
            return true;
        }
    }
    return false;
}
public static boolean isArmstrongNumber(int number) {
    int[] digits = storeDigits(number);
    int digitCount = digits.length;
    int sum = 0;
    for (int digit : digits) {
        sum += Math.pow(digit, digitCount);
    }
    return sum == number;
}
public static int[] findLargestAndSecondLargest(int[] digits) {
    int largest = Integer.MIN_VALUE;
    int secondLargest = Integer.MIN_VALUE;

    for (int digit : digits) {
        if (digit > largest) {
            secondLargest = largest;
            largest = digit;
        } else if (digit > secondLargest && digit != largest) {
            secondLargest = digit;
        }
    }
    return new int[] { largest, secondLargest };
}
public static int[] findSmallestAndSecondSmallest(int[] digits) {
    int smallest = Integer.MAX_VALUE;
    int secondSmallest = Integer.MAX_VALUE;

    for (int digit : digits) {
        if (digit < smallest) {
            secondSmallest = smallest;
            smallest = digit;
        } else if (digit < secondSmallest && digit != smallest) {
            secondSmallest = digit;
        }
    }
    return new int[] { smallest, secondSmallest };
}
public static void main(String[] args) {
    int number = 153;
    System.out.println("Count of digits: " + countDigits(number));
    int[] digits = storeDigits(number);
    System.out.print("Digits: ");
    for (int digit : digits) {
        System.out.print(digit + " ");
    }
    System.out.println();
```

```
        if (isDuckNumber(number)) {
            System.out.println(number + " is a Duck Number.");
        } else {
            System.out.println(number + " is not a Duck Number.");
        }
        if (isArmstrongNumber(number)) {
            System.out.println(number + " is an Armstrong Number.");
        } else {
            System.out.println(number + " is not an Armstrong
Number.");
        }
        int[] largestAndSecondLargest =
findLargestAndSecondLargest(digits);
        System.out.println("Largest digit: " +
largestAndSecondLargest[0]);
        System.out.println("Second largest digit: " +
largestAndSecondLargest[1]);
        int[] smallestAndSecondSmallest =
findSmallestAndSecondSmallest(digits);
        System.out.println("Smallest digit: " +
smallestAndSecondSmallest[0]);
        System.out.println("Second smallest digit: " +
smallestAndSecondSmallest[1]);
    }
}
```

OUTPUT:

```
Count of digits: 3
Digits: 1 5 3
153 is a Duck Number.
```

```
153 is an Armstrong Number.
Largest digit: 5
Second largest digit: 3
Smallest digit: 1
Second smallest digit: 3
```

QUES 3

```
class NumberChecker3 {
    public static int countDigits(int number) {
        int count = 0;
        while (number != 0) {
            number /= 10;
            count++;
        }
        return count;
    }
    public static int[] storeDigits(int number) {
        int digitCount = countDigits(number);
        int[] digits = new int[digitCount];
        int index = 0;
        while (number != 0) {
            digits[index++] = number % 10;
            number /= 10;
        }
        for (int i = 0; i < digitCount / 2; i++) {
            int temp = digits[i];
            digits[i] = digits[digitCount - 1 - i];
            digits[digitCount - 1 - i] = temp;
        }
        return digits;
    }
    public static int sumOfDigits(int number) {
        int sum = 0;
        while (number != 0) {
            sum += number % 10;
            number /= 10;
        }
        return sum;
    }

    // Method to find the sum of the squares of the digits of the number
    public static double sumOfSquaresOfDigits(int number) {
        int[] digits = storeDigits(number);
        double sumOfSquares = 0;
        for (int digit : digits) {
            sumOfSquares += Math.pow(digit, 2);
        }
        return sumOfSquares;
    }
    public static boolean isHarshadNumber(int number) {
        int sum = sumOfDigits(number);
        return sum != 0 && number % sum == 0;
    }
```

```
public static int[][] findDigitFrequency(int number) {
    int[] digits = storeDigits(number);
    int[][] frequency = new int[10][2];

    for (int digit : digits) {
        frequency[digit][0] = digit;
        frequency[digit][1]++;
    }
    int count = 0;
    for (int[] row : frequency) {
        if (row[1] > 0) {
            count++;
        }
    }
    int[][] result = new int[count][2];
    int index = 0;
    for (int[] row : frequency) {
        if (row[1] > 0) {
            result[index++] = row;
        }
    }

    return result;
}
public static void main(String[] args) {
    int number = 21;
```

```
        System.out.println("Count of digits: " + countDigits(number));
        int[] digits = storeDigits(number);
        System.out.print("Digits: ");
        for (int digit : digits) {
            System.out.print(digit + " ");
        }
        System.out.println();
        System.out.println("Sum of digits: " + sumOfDigits(number));
        System.out.println("Sum of squares of digits: " +
sumOfSquaresOfDigits(number));
        if (isHarshadNumber(number)) {
            System.out.println(number + " is a Harshad Number.");
        } else {
            System.out.println(number + " is not a Harshad Number.");
        }
        int[][] frequencies = findDigitFrequency(number);
        System.out.println("Frequency of each digit:");
        for (int[] entry : frequencies) {
            System.out.println("Digit " + entry[0] + ": " + entry[1] + "
times");
        }
    }
}
```

OUTPUT:

```
Count of digits: 2
Digits: 2 1
Sum of digits: 3
Sum of squares of digits: 5.0
21 is a Harshad Number.
Frequency of each digit:
Digit 2: 1 times
Digit 1: 1 times
```

QUES 4

```
class NumberChecker4 {
    public static int countDigits(int number) {
        int count = 0;
        while (number != 0) {
            number /= 10;
            count++;
        }
        return count;
    }
    public static int[] storeDigits(int number) {
        int digitCount = countDigits(number);
        int[] digits = new int[digitCount];
        int index = 0;
        while (number != 0) {
            digits[index++] = number % 10;
            number /= 10;
        }
        for (int i = 0; i < digitCount / 2; i++) {
            int temp = digits[i];
            digits[i] = digits[digitCount - 1 - i];
            digits[digitCount - 1 - i] = temp;
        }
        return digits;
    }
    public static void reverseArray(int[] digits) {
        int start = 0;
        int end = digits.length - 1;
        while (start < end) {
            int temp = digits[start];
            digits[start] = digits[end];
            digits[end] = temp;
            start++;
            end--;
        }
    }
    public static boolean areArraysEqual(int[] array1, int[] array2) {
        if (array1.length != array2.length) {
            return false;
        }
        for (int i = 0; i < array1.length; i++) {
            if (array1[i] != array2[i]) {
                return false;
            }
        }
        return true;
    }
```

```
public static boolean isPalindrome(int number) {
    int[] digits = storeDigits(number);
    int[] reversedDigits = digits.clone();
    reverseArray(reversedDigits);
    return areArraysEqual(digits, reversedDigits);
}
public static boolean isDuckNumber(int number) {
    int[] digits = storeDigits(number);
    for (int digit : digits) {
        if (digit != 0) {
            return true;
        }
    }
    return false;
}
public static void main(String[] args) {
    int number = 101;
    System.out.println("Count of digits: " + countDigits(number));
    int[] digits = storeDigits(number);
    System.out.print("Digits: ");
    for (int digit : digits) {
        System.out.print(digit + " ");
    }
    System.out.println();
    if (isPalindrome(number)) {
        System.out.println(number + " is a Palindrome.");
        } else {
            System.out.println(number + " is not a Palindrome.");
        }
        if (isDuckNumber(number)) {
            System.out.println(number + " is a Duck Number.");
        } else {
            System.out.println(number + " is not a Duck Number.");
        }
        int[] reversedDigits = digits.clone();
        reverseArray(reversedDigits);
        System.out.print("Reversed Digits: ");
        for (int digit : reversedDigits) {
            System.out.print(digit + " ");
        }
        System.out.println();
        if (areArraysEqual(digits, reversedDigits)) {
            System.out.println("The original and reversed arrays are
equal.");
        } else {
            System.out.println("The original and reversed arrays are not
equal.");
        }
    }
}
```

OUTPUT:

```
For number: 12321
Count of digits: 5
```

```
Digits: [1, 2, 3, 2, 1]
Reversed Digits: [1, 2, 3, 2, 1]
Is Palindrome: true
Is Duck Number: true
Are the original digits and reversed digits equal? True
```

QUES 5

```
class NumberChecker5 {
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
    public static boolean isNeon(int number) {
        int square = number * number;
        int sumOfDigits = 0;

        while (square > 0) {
            sumOfDigits += square % 10;
            square /= 10;
        }
        return sumOfDigits == number;
    }
    public static boolean isSpy(int number) {
        int sum = 0;
        int product = 1;
        while (number > 0) {
            int digit = number % 10;
            sum += digit;
            product *= digit;
            number /= 10;
        }
        return sum == product;
    }
    public static boolean isAutomorphic(int number) {
        int square = number * number;
        return
String.valueOf(square).endsWith(String.valueOf(number));
    }
    public static boolean isBuzz(int number) {
        return number % 7 == 0 ||
String.valueOf(number).endsWith("7");
    }
    public static void main(String[] args) {
        int[] testNumbers = {7, 9, 45, 25, 370, 107, 89};
        for (int number : testNumbers) {
            System.out.println("For number: " + number);
            System.out.println("Is Prime: " + isPrime(number));
            System.out.println("Is Neon: " + isNeon(number));
            System.out.println("Is Spy: " + isSpy(number));
            System.out.println("Is Automorphic: " +
isAutomorphic(number));
            System.out.println("Is Buzz: " + isBuzz(number));

            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
    }
}
```

OUTPUT:

```
For number: 7
Is Prime: true
Is Neon: false
Is Spy: false
Is Automorphic: true
Is Buzz: true
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 9
Is Prime: false
Is Neon: false
Is Spy: false
Is Automorphic: false
Is Buzz: false
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 45
```

```
Is Prime: false
Is Neon: false
Is Spy: false
Is Automorphic: false
Is Buzz: false
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 25
Is Prime: false
Is Neon: false
Is Spy: false
Is Automorphic: true
Is Buzz: false
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 370
Is Prime: false
Is Neon: true
Is Spy: false
Is Automorphic: false
Is Buzz: false
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 107
Is Prime: true
Is Neon: false
Is Spy: false
Is Automorphic: false
Is Buzz: true
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
For number: 89
Is Prime: true
Is Neon: false
Is Spy: false
Is Automorphic: false
Is Buzz: false
```

QUES 6

```java
import java.math.BigInteger;
class NumberChecker6 {
    public static int[] findFactors(int number) {
        int count = 0;
        for (int i = 1; i <= number; i++) {
            if (number % i == 0) {
                count++;
            }
        }
        int[] factors = new int[count];
        int index = 0;
        for (int i = 1; i <= number; i++) {
            if (number % i == 0) {
                factors[index++] = i;
            }
        }
        return factors;
    }
```

```
public static int greatestFactor(int number) {
    int[] factors = findFactors(number);
    return factors[factors.length - 1];
}
public static int sumOfFactors(int number) {
    int[] factors = findFactors(number);
    int sum = 0;
    for (int factor : factors) {
        sum += factor;
    }
    return sum;
}
public static int productOfFactors(int number) {
    int[] factors = findFactors(number);
    int product = 1;
    for (int factor : factors) {
        product *= factor;
    }
    return product;
}
public static double productOfCubeOfFactors(int number) {
    int[] factors = findFactors(number);
    double product = 1;
    for (int factor : factors) {
        product *= Math.pow(factor, 3);
    }
    return product;
}
public static boolean isPerfect(int number) {
    int sum = sumOfFactors(number) - number;
    return sum == number;
}
public static boolean isAbundant(int number) {
    int sum = sumOfFactors(number) - number;
    return sum > number;
}
public static boolean isDeficient(int number) {
    int sum = sumOfFactors(number) - number;
    return sum < number;
}
public static boolean isStrong(int number) {
    int originalNumber = number;
    int sumOfFactorialDigits = 0;
    while (number > 0) {
        int digit = number % 10;
        sumOfFactorialDigits += factorial(digit);
        number /= 10;
    }

    return sumOfFactorialDigits == originalNumber;
}
private static int factorial(int number) {
        if (number == 0 || number == 1) {
            return 1;
        }
        int result = 1;
        for (int i = 2; i <= number; i++) {
            result *= i;
        }
        return result;
    }

    public static void main(String[] args) {
        int number = 28;
        int[] factors = findFactors(number);
        System.out.println("Factors of " + number + ": ");
        for (int factor : factors) {
            System.out.print(factor + " ");
        }
        System.out.println();
        System.out.println("Greatest factor: " +
greatestFactor(number));
        System.out.println("Sum of factors: " +
sumOfFactors(number));
        System.out.println("Product of factors: " +
productOfFactors(number));
        System.out.println("Product of cubes of factors: " +
productOfCubeOfFactors(number));
        System.out.println("Is Perfect: " + isPerfect(number));
        System.out.println("Is Abundant: " + isAbundant(number));
        System.out.println("Is Deficient: " + isDeficient(number));
        System.out.println("Is Strong: " + isStrong(number));
    }
}
```

OUTPUT:

```
Factors of 28:
1 2 4 7 14 28
Greatest factor: 28
Sum of factors: 56
Product of factors: 784
Product of cubes of factors: 37012500.0
Is Perfect: true
Is Abundant: true
Is Deficient: false
Is Strong: false
```

QUES 7

```
import java.util.HashSet;
import java.util.Set;
class OTPGenerator {
    public static int generateOTP() {
        int otp = (int) (Math.random() * 900000) + 100000;
        return otp;
    }
    public static boolean areOTPsUnique(int[] otpArray) {
        Set<Integer> otpSet = new HashSet<>();
        for (int otp : otpArray) {
            if (!otpSet.add(otp)) {
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        int[] otpNumbers = new int[10];
        for (int i = 0; i < otpNumbers.length; i++) {
            otpNumbers[i] = generateOTP();
            System.out.println("OTP " + (i + 1) + ": " + otpNumbers[i]);
        }
        boolean areUnique = areOTPsUnique(otpNumbers);
        if (areUnique) {
            System.out.println("\nAll OTPs are unique.");
        } else {
            System.out.println("\nSome OTPs are not unique.");
        }
    }
}
```

OUTPUT:

```
OTP 1: 736249
OTP 2: 871654
OTP 3: 286785
OTP 4: 614532
OTP 5: 573899
OTP 6: 210375
OTP 7: 986715
OTP 8: 783495
OTP 9: 536286
OTP 10: 184793

All OTPs are unique.
```

QUES 8

```
import java.util.Scanner;
class CalendarDisplay {
    public static String getMonthName(int month) {
        String[] months = {
            "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "November",
"December"
        };
        return months[month - 1];
    }
    public static boolean isLeapYear(int year) {
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
{
            return true;
        }
        return false;
    }
    public static int getNumberOfDaysInMonth(int month, int year) {
        int[] daysInMonth = {
```

```
        31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
    };
    if (month == 2 && isLeapYear(year)) {
        return 29;
    }
    return daysInMonth[month - 1];
}
public static int getFirstDayOfMonth(int month, int year) {
    if (month < 3) {
        month += 12;
        year--;
    }
    int y0 = year - (14 - month) / 12;
    int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;
    int m0 = month + 12 * ((14 - month) / 12) - 2;
    int d0 = (1 + x + 31 * m0 / 12) % 7;

    return d0;
}
public static void displayCalendar(int month, int year) {
    String monthName = getMonthName(month);
    int numberOfDays = getNumberOfDaysInMonth(month, year);
    int firstDay = getFirstDayOfMonth(month, year);
    System.out.println("Calendar for " + monthName + " " + year);
    System.out.println("Sun Mon Tue Wed Thu Fri Sat");
    for (int i = 0; i < firstDay; i++) {
            System.out.print("    ");
        }
        for (int day = 1; day <= numberOfDays; day++) {
            System.out.printf("%3d ", day);
            if ((day + firstDay) % 7 == 0) {
                System.out.println();
            }
        }
        System.out.println();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter month (1-12): ");
        int month = scanner.nextInt();
        System.out.print("Enter year: ");
        int year = scanner.nextInt();
        displayCalendar(month, year);

        scanner.close();
    }
}
```

```
OUTPUT:
Enter month (1-12): 7
Enter year: 2005
Calendar for July 2005
Sun Mon Tue Wed Thu Fri Sat
                          1   2
 3    4    5    6    7    8   9
10   11   12   13   14   15  16
17   18   19   20   21   22  23
24   25   26   27   28   29  30
31
```

QUES 9

```
import java.util.Scanner;
class EuclideanDistanceAndLineEquation {
   public static double calculateEuclideanDistance(double x1, double
y1, double x2, double y2) {
       return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1,
2));
   }
   public static double[] calculateLineEquation(double x1, double y1,
double x2, double y2) {
       double[] lineEquation = new double[2];
       double m = (y2 - y1) / (x2 - x1);
       double b = y1 - m * x1;
       lineEquation[0] = m;
       lineEquation[1] = b;
       return lineEquation;
```

```
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter x1: ");
        double x1 = scanner.nextDouble();
        System.out.print("Enter y1: ");
        double y1 = scanner.nextDouble();
        System.out.print("Enter x2: ");
        double x2 = scanner.nextDouble();
        System.out.print("Enter y2: ");
        double y2 = scanner.nextDouble();
        double distance = calculateEuclideanDistance(x1, y1, x2, y2);
        System.out.println("Euclidean Distance between the points: " +
distance);
        double[] lineEquation = calculateLineEquation(x1, y1, x2, y2);
        double slope = lineEquation[0];
        double yIntercept = lineEquation[1];
        System.out.println("The equation of the line is: y = " + slope +
"x + " + yIntercept);

        scanner.close();
    }
}
```

OUTPUT:

```
Enter x1: 1
Enter y1: 2
Enter x2: 4
Enter y2: 6
Euclidean Distance between the points: 5.0
The equation of the line is: y = 1.3333333333333333x + 0.6666666666666667
```

QUES 10

```
import java.util.Scanner;
class CollinearPoints {
    public static boolean areCollinearBySlope(double x1, double y1, double x2, double y2, double x3, double y3) {
        double slopeAB = (y2 - y1) / (x2 - x1);
        double slopeBC = (y3 - y2) / (x3 - x2);
        double slopeAC = (y3 - y1) / (x3 - x1);
        return slopeAB == slopeBC && slopeAB == slopeAC;
    }
    public static boolean areCollinearByArea(double x1, double y1, double x2, double y2, double x3, double y3) {
        double area = 0.5 * (x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2));
        return area == 0;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter x1: ");
        double x1 = scanner.nextDouble();
        System.out.print("Enter y1: ");
        double y1 = scanner.nextDouble();
        System.out.print("Enter x2: ");
        double x2 = scanner.nextDouble();
        System.out.print("Enter y2: ");
        double y2 = scanner.nextDouble();
        System.out.print("Enter x3: ");
        double x3 = scanner.nextDouble();
        System.out.print("Enter y3: ");
        double y3 = scanner.nextDouble();
        boolean collinearBySlope = areCollinearBySlope(x1, y1, x2, y2,
x3, y3);
        System.out.println("Are points collinear by slope method? " +
(collinearBySlope ? "Yes" : "No"));
        boolean collinearByArea = areCollinearByArea(x1, y1, x2, y2,
x3, y3);
        System.out.println("Are points collinear by area method? " +
(collinearByArea ? "Yes" : "No"));


        scanner.close();
    }
}
```

INPUT:

```
Enter x1: 2
Enter y1: 4
Enter x2: 4
Enter y2: 6
Enter x3: 6
Enter y3: 8
```

OUTPUT:

```
Are points collinear by slope method? Yes
Are points collinear by area method? Yes
```

QUES 11

```
import java.util.Random;
class EmployeeBonusCalculator {
    public static double[][] generateEmployeeData(int numEmployees)
{
        Random rand = new Random();
        double[][] employeeData = new double[numEmployees][2];
        for (int i = 0; i < numEmployees; i++) {
            double salary = 10000 + (rand.nextInt(90000));
            int yearsOfService = 1 + rand.nextInt(20);
            employeeData[i][0] = salary;
            employeeData[i][1] = yearsOfService;
        }
        return employeeData;
    }
    public static double[][] calculateNewSalaryAndBonus(double[][]
employeeData) {
        double[][] updatedData = new double[employeeData.length][3];
        for (int i = 0; i < employeeData.length; i++) {
            double salary = employeeData[i][0];
            int yearsOfService = (int) employeeData[i][1];
            double bonus = 0;
            if (yearsOfService > 5) {
                bonus = salary * 0.05;
            } else {
                bonus = salary * 0.02;
            }
            double newSalary = salary + bonus;
            updatedData[i][0] = salary;
            updatedData[i][1] = newSalary;
            updatedData[i][2] = bonus;
        }
        return updatedData;
    }
    public static void calculateTotals(double[][] updatedData) {
        double totalOldSalary = 0;
        double totalNewSalary = 0;
        double totalBonus = 0;

        for (int i = 0; i < updatedData.length; i++) {
            totalOldSalary += updatedData[i][0];
            totalNewSalary += updatedData[i][1];
            totalBonus += updatedData[i][2];
        }
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~");
        System.out.println("Employee No | Old Salary | New Salary |
Bonus");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~");
        for (int i = 0; i < updatedData.length; i++) {
            System.out.printf("%~12d | %~10.2f | %~10.2f | %~10.2f\n",
i + 1, updatedData[i][0], updatedData[i][1], updatedData[i][2]);
        }
```

```
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~");
        System.out.printf("Total       | %.2f    | %.2f    | %.2f\n",
totalOldSalary, totalNewSalary, totalBonus);
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~");
    }
    public static void main(String[] args) {
        int numEmployees = 10;
        double[][] employeeData =
generateEmployeeData(numEmployees);
        double[][] updatedData =
calculateNewSalaryAndBonus(employeeData);
        calculateTotals(updatedData);
    }
}
```

OUTPUT:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Employee No | Old Salary | New Salary | Bonus
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1           | 82325.00   | 86441.25   | 4116.25
2           | 30325.00   | 30931.50   | 606.50
3           | 56678.00   | 59511.00   | 2833.00
4           | 78902.00   | 82147.10   | 3245.10
5           | 10732.00   | 10947.64   | 215.64
6           | 48961.00   | 50686.05   | 1725.05
7           | 19537.00   | 19950.74   | 413.74
8           | 60979.00   | 64028.95   | 3039.95
```

```
9         | 17432.00   | 17780.64   | 348.64
10        | 74102.00   | 77307.10   | 3205.10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Total     | 531081.00  | 551824.57  | 20743.57
```

QUES 12

```
import java.util.Random;
import java.util.Scanner;
class StudentScores {
   public static int[][] generateScores(int numStudents) {
      Random rand = new Random();
      int[][] scores = new int[numStudents][3];
      for (int i = 0; i < numStudents; i++) {
         scores[i][0] = rand.nextInt(50) + 50;
         scores[i][1] = rand.nextInt(50) + 50;
         scores[i][2] = rand.nextInt(50) + 50;
      }
      return scores;
   }
   public static double[][] calculateResults(int[][] scores) {
      double[][] results = new double[scores.length][4];
      for (int i = 0; i < scores.length; i++) {
         int total = scores[i][0] + scores[i][1] + scores[i][2];
         double average = total / 3.0;
         double percentage = (total / 300.0) * 100;
         results[i][0] = total;
         results[i][1] = Math.round(average * 100.0) / 100.0;
```

```
            results[i][2] = Math.round(percentage * 100.0) / 100.0;
        }
        return results;
    }
    public static void displayScorecard(int[][] scores, double[][]
results) {
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~");
        System.out.println("Student
No.\tPhysics\tChemistry\tMath\tTotal\tAverage\tPercentage");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~");
        for (int i = 0; i < scores.length; i++) {
            System.out.printf("%d\t\t", i + 1);
            System.out.printf("%d\t\t%d\t\t%d\t\t", scores[i][0],
scores[i][1], scores[i][2]);
            System.out.printf("%d\t%.2f\t%.2f\n", (int) results[i][0],
results[i][1], results[i][2]);
        }
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of students: ");
        int numStudents = scanner.nextInt();
        int[][] scores = generateScores(numStudents);
        double[][] results = calculateResults(scores);
        displayScorecard(scores, results);
```

```
        scanner.close();
    }
}
```

OUTPUT:

```
Enter the number of students: 5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Student No. Physics Chemistry Math Total Average Percentage
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    1          72        92         63   227   75.67      75.67
    2          54         78        98   230   76.67      76.67
    3          91         56        87   234   78.00      78.00
    4          77         84        61   222   74.00      74.00
    5          63         69        89   221   73.67      73.67
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

QUES 13

```
import java.util.Random;]
class MatrixOperations {
    public static int[][] createRandomMatrix(int rows, int cols) {
        Random rand = new Random();
        int[][] matrix = new int[rows][cols];
       for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = rand.nextInt(10);
            }
        }
```

```
        return matrix;
    }
    public static void displayMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
    }
    public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
        int rows = matrix1.length;
        int cols = matrix1[0].length;
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }
        return result;
    }
    public static int[][] subtractMatrices(int[][] matrix1, int[][]
matrix2) {
        int rows = matrix1.length;
        int cols = matrix1[0].length;
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] - matrix2[i][j];
            }
        }
        return result;
    }
    public static int[][] multiplyMatrices(int[][] matrix1, int[][]
matrix2) {
        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int rows2 = matrix2.length;
        int cols2 = matrix2[0].length;
        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible,
incompatible dimensions.");
            return null;
        }
        int[][] result = new int[rows1][cols2];

        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        return result;
    }
    public static int[][] transposeMatrix(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] result = new int[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[j][i] = matrix[i][j];
            }
        }
        return result;
    }
    public static int determinant2x2(int[][] matrix) {
        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
    }
    public static int determinant3x3(int[][] matrix) {
        int det = matrix[0][0] * (matrix[1][1] * matrix[2][2] -
matrix[1][2] * matrix[2][1])
                - matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2]
* matrix[2][0])
                + matrix[0][2] * (matrix[1][0] * matrix[2][1] -
matrix[1][1] * matrix[2][0]);
        return det;
    }
    public static double[][] inverse2x2(int[][] matrix) {
        int determinant = determinant2x2(matrix);
        if (determinant == 0) {
            System.out.println("Matrix is not invertible (determinant is
0).");
            return null;
        }

        double[][] inverse = new double[2][2];
        inverse[0][0] = matrix[1][1] / (double) determinant;
        inverse[0][1] = -matrix[0][1] / (double) determinant;
        inverse[1][0] = -matrix[1][0] / (double) determinant;
        inverse[1][1] = matrix[0][0] / (double) determinant;

        return inverse;
    }
    public static double[][] inverse3x3(int[][] matrix) {
        int determinant = determinant3x3(matrix);

        if (determinant == 0) {
            System.out.println("Matrix is not invertible (determinant is
0).");
            return null;
        }

        double[][] adjoint = new double[3][3];
        adjoint[0][0] = matrix[1][1] * matrix[2][2] - matrix[1][2] *
matrix[2][1];
```

```
        adjoint[0][1] = matrix[0][2] * matrix[2][1] - matrix[0][1] *
matrix[2][2];
        adjoint[0][2] = matrix[0][1] * matrix[1][2] - matrix[0][2] *
matrix[1][1];
        adjoint[1][0] = matrix[1][2] * matrix[2][0] - matrix[1][0] *
matrix[2][2];
        adjoint[1][1] = matrix[0][0] * matrix[2][2] - matrix[0][2] *
matrix[2][0];
        adjoint[1][2] = matrix[0][2] * matrix[1][0] - matrix[0][0] *
matrix[1][2];
        adjoint[2][0] = matrix[1][0] * matrix[2][1] - matrix[1][1] *
matrix[2][0];
        adjoint[2][1] = matrix[0][1] * matrix[2][0] - matrix[0][0] *
matrix[2][1];
        adjoint[2][2] = matrix[0][0] * matrix[1][1] - matrix[0][1] *
matrix[1][0];

        double[][] inverse = new double[3][3];

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                inverse[i][j] = adjoint[i][j] / determinant;
            }
        }

        return inverse;
    }
    public static void main(String[] args) {
        Random rand = new Random();
        int rows = 3, cols = 3;
        int[][] matrix1 = createRandomMatrix(rows, cols);
        int[][] matrix2 = createRandomMatrix(rows, cols);
        System.out.println("Matrix 1:");
        displayMatrix(matrix1);
        System.out.println("\nMatrix 2:");
        displayMatrix(matrix2);
        System.out.println("\nMatrix 1 + Matrix 2:");
        int[][] sum = addMatrices(matrix1, matrix2);
        displayMatrix(sum);
        System.out.println("\nMatrix 1 - Matrix 2:");
        int[][] difference = subtractMatrices(matrix1, matrix2);
        displayMatrix(difference);
        System.out.println("\nMatrix 1 * Matrix 2:");
        int[][] product = multiplyMatrices(matrix1, matrix2);
        if (product != null) {
            displayMatrix(product);
        }
        System.out.println("\nTranspose of Matrix 1:");
        int[][] transpose = transposeMatrix(matrix1);
        displayMatrix(transpose);
        if (rows == 2 && cols == 2) {
            System.out.println("\nDeterminant of Matrix 1 (2x2): " +
determinant2x2(matrix1));
            System.out.println("Inverse of Matrix 1 (2x2):");
            double[][] inverse2x2 = inverse2x2(matrix1);
            if (inverse2x2 != null) {
                displayMatrix(inverse2x2);
            }
        } else if (rows == 3 && cols == 3) {
            System.out.println("\nDeterminant of Matrix 1 (3x3): " +
determinant3x3(matrix1));
            System.out.println("Inverse of Matrix 1 (3x3):");
            double[][] inverse3x3 = inverse3x3(matrix1);
            if (inverse3x3 != null) {
                displayMatrix(inverse3x3);
            }
        }
    }
}
```

OUTPUT:

```
Matrix 1:
7     1
3     5

Matrix 2:
4     8
6     2

Matrix 1 + Matrix 2:
11    9
9     7

Matrix 1 - Matrix 2:
```

```
3	-7
-3	3

Matrix 1 * Matrix 2:
38	62
42	16

Transpose of Matrix 1:
7	3
1	5

Determinant of Matrix 1: 26

Inverse of Matrix 1:
0.19230769230769232	-0.038461538461538464
-0.11538461538461539	0.2692307692307692
```