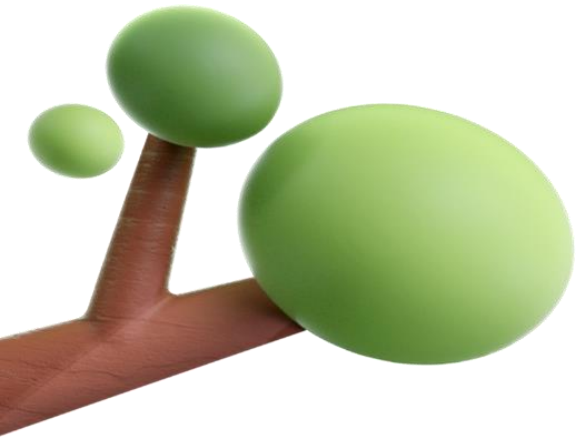


LEVEL 3 ASSIGNMENT

STEP PROGRAM

JAVA METHODS

SUBMITTED BY:

NAME: E.Harikrishna

REG NO: RA2411026010408

CLASS: AJ1

SOURCE CODE

LEVEL 3 QUESTION: 1

Q. Create a program to find the shortest, tallest, and mean height of players present in a football team.

```
import java.util.Random;

public class FootballTeamHeightStats {
    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;
        }

        int sum = findSum(heights);
        double mean = findMean(sum, heights.length);
        int shortest = findShortest(heights);
        int tallest = findTallest(heights);

        System.out.println("Heights of players:");
```

```
    for (int height : heights) {
        System.out.print(height + " ");
    }
    System.out.println("\n\nSum of heights: " + sum);
    System.out.println("Mean height: " + mean);
    System.out.println("Shortest height: " + shortest);
    System.out.println("Tallest height: " + tallest);
}

public static int findSum(int[] arr) {
    int sum = 0;
    for (int num : arr) {
        sum += num;
    }
    return sum;
}

public static double findMean(int sum, int count) {
    return (double) sum / count;
}

public static int findShortest(int[] arr) {
    int min = arr[0];
    for (int num : arr) {
```

```
            if (num < min) {
                min = num;
            }
        }
        return min;
    }

    public static int findTallest(int[] arr) {
        int max = arr[0];
        for (int num : arr) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
}
```

OUTPUT:

```
Heights of players:
189 153 220 236 209 184 244 213 195 163 210

Sum of heights: 2216
Mean height: 201.45454545454547
Shortest height: 153
Tallest height: 244
(base) harikrishna@Harikrishnas-MacBook-Air-10
```

LEVEL3 QUESTION: 2

Q. Extend or Create a NumberChecker utility class and perform following task. Call from main() method the different methods and display results. Make sure all are static methods

```
import java.util.Arrays;


public class NumberChecker {
    public static void main(String[] args) {
        int number = 153;


        int digitCount = countDigits(number);
        int[] digits = getDigitsArray(number, digitCount);
        boolean isDuck = isDuckNumber(digits);
        boolean isArmstrong = isArmstrongNumber(digits, number);
        int[] largestTwo = findTwoLargest(digits);
        int[] smallestTwo = findTwoSmallest(digits);
```

```
        System.out.println("Number: " + number);
        System.out.println("Digits: " + Arrays.toString(digits));
        System.out.println("Number of digits: " + digitCount);
        System.out.println("Is Duck Number: " + isDuck);
        System.out.println("Is Armstrong Number: " + isArmstrong);
        System.out.println("Largest digit: " + largestTwo[0]);
        System.out.println("Second largest digit: " + largestTwo[1]);
        System.out.println("Smallest digit: " + smallestTwo[0]);
        System.out.println("Second smallest digit: " + smallestTwo[1]);
    }

    public static int countDigits(int number) {
        int count = 0;
        int temp = number;
        while (temp > 0) {
            temp /= 10;
            count++;
        }
        return count;
    }
```

```
public static int[] getDigitsArray(int number, int count) {
    int[] digits = new int[count];
    int index = count - 1;
    while (number > 0) {
        digits[index--] = number % 10;
        number /= 10;
    }
    return digits;
}

public static boolean isDuckNumber(int[] digits) {
    for (int digit : digits) {
        if (digit == 0) {
            return true;
        }
    }
    return false;
}

public static boolean isArmstrongNumber(int[] digits, int number) {
    int sum = 0;
    int power = digits.length;
```

```
    for (int digit : digits) {
        sum += Math.pow(digit, power);
    }
    return sum == number;
}

public static int[] findTwoLargest(int[] digits) {
    int first = Integer.MIN_VALUE;
    int second = Integer.MIN_VALUE;
    for (int digit : digits) {
        if (digit > first) {
            second = first;
            first = digit;
        } else if (digit > second && digit != first) {
            second = digit;
        }
    }
    return new int[]{first, second};
}

public static int[] findTwoSmallest(int[] digits) {
    int first = Integer.MAX_VALUE;
```

```
        int second = Integer.MAX_VALUE;
        for (int digit : digits) {
            if (digit < first) {
                second = first;
                first = digit;
            } else if (digit < second && digit != first) {
                second = digit;
            }
        }
        return new int[]{first, second};
    }
}
```

OUTPUT:

```
(base) harikrishna@Harikrishnas-MacB
Number: 153
Digits: [1, 5, 3]
Number of digits: 3
Is Duck Number: false
Is Armstrong Number: true
Largest digit: 5
Second largest digit: 3
Smallest digit: 1
Second smallest digit: 3
(base) harikrishna@Harikrishnas-MacBo
```

LEVEL 3 QUESTION: 3

Q. Extend or Create a NumberChecker utility class and perform following task. Call from main() method the different methods and display results. Make sure all are static methods

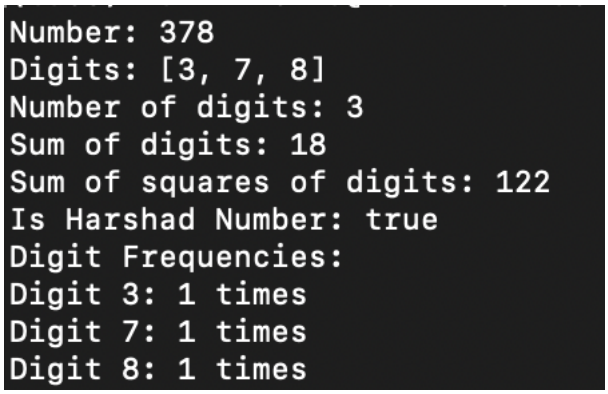
```
import java.util.Arrays;

public class NumberCheck1 {

   public static void main(String[] args) {

      int number = 378;

      int digitCount = countDigits(number);

      int[] digits = getDigitsArray(number, digitCount);

      int sumOfDigits = getSumOfDigits(digits);

      int sumOfSquares = getSumOfSquares(digits);

      boolean isHarshad = isHarshadNumber(number, sumOfDigits);

      int[][] frequency = getDigitFrequency(digits);

      System.out.println("Number: " + number);

      System.out.println("Digits: " + Arrays.toString(digits));

      System.out.println("Number of digits: " + digitCount);

      System.out.println("Sum of digits: " + sumOfDigits);

      System.out.println("Sum of squares of digits: " + sumOfSquares);

      System.out.println("Is Harshad Number: " + isHarshad);

      System.out.println("Digit Frequencies:");

      for (int[] pair : frequency) {
```

```
        if (pair[1] > 0) {
            System.out.println("Digit " + pair[0] + ": " + pair[1] + " times");
        }
    }
}
public static int countDigits(int number) {
    int count = 0;
    int temp = number;
    while (temp > 0) {
        temp /= 10;
        count++;
    }
    return count;
}
public static int[] getDigitsArray(int number, int count) {
    int[] digits = new int[count];
    int index = count - 1;
    while (number > 0) {
        digits[index--] = number % 10;
        number /= 10;
    }
    return digits;
```

```
}
public static int getSumOfDigits(int[] digits) {
    int sum = 0;
    for (int digit : digits) {
        sum += digit;
    }
    return sum;
}
public static int getSumOfSquares(int[] digits) {
    int sum = 0;
    for (int digit : digits) {
        sum += Math.pow(digit, 2);
    }
    return sum;
}
public static boolean isHarshadNumber(int number, int sumOfDigits) {
    return number % sumOfDigits == 0;
}
public static int[][] getDigitFrequency(int[] digits) {
    int[][] frequency = new int[10][2];
    for (int i = 0; i < 10; i++) {
        frequency[i][0] = i;
```

```
    }
    for (int digit : digits) {
      frequency[digit][1]++;
    }
    return frequency;
  }
}
```

OUTPUT:



LEVEL 3 QUESTION: 4

Q. Extend or Create a NumberChecker utility class and perform following task. Call from main() method the different methods and display results. Make sure all are static methods

```
import java.util.Arrays;
```

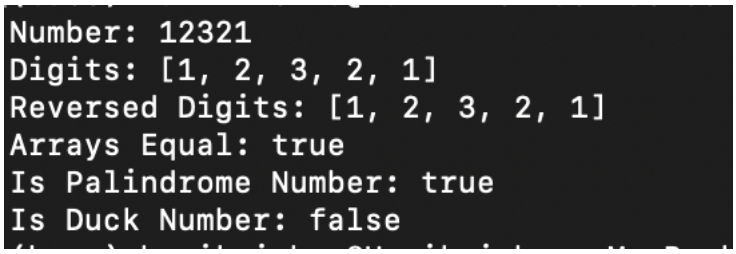
```
public class NumberChecker {
    public static void main(String[] args) {
        int number = 12321;

        int digitCount = countDigits(number);
        int[] digits = getDigitsArray(number, digitCount);
        int[] reversedDigits = reverseArray(digits);
        boolean arraysEqual = areArraysEqual(digits, reversedDigits);
        boolean isPalindrome = isPalindromeNumber(digits);
        boolean isDuck = isDuckNumber(digits);

        System.out.println("Number: " + number);
        System.out.println("Digits: " + Arrays.toString(digits));
        System.out.println("Reversed Digits: " + Arrays.toString(reversedDigits));
        System.out.println("Arrays Equal: " + arraysEqual);
        System.out.println("Is Palindrome Number: " + isPalindrome);
        System.out.println("Is Duck Number: " + isDuck);
    }

    public static int countDigits(int number) {
        int count = 0;
        int temp = number;
```

```
    while (temp > 0) {
        temp /= 10;
        count++;
    }
    return count;
}

public static int[] getDigitsArray(int number, int count) {
    int[] digits = new int[count];
    int index = count - 1;
    while (number > 0) {
        digits[index--] = number % 10;
        number /= 10;
    }
    return digits;
}

public static int[] reverseArray(int[] arr) {
    int[] reversed = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        reversed[i] = arr[arr.length - 1 - i];
    }
```

```
    return reversed;
}

public static boolean areArraysEqual(int[] arr1, int[] arr2) {
    if (arr1.length != arr2.length) return false;
    for (int i = 0; i < arr1.length; i++) {
        if (arr1[i] != arr2[i]) return false;
    }
    return true;
}

public static boolean isPalindromeNumber(int[] digits) {
    int[] reversed = reverseArray(digits);
    return areArraysEqual(digits, reversed);
}

public static boolean isDuckNumber(int[] digits) {
    for (int i = 1; i < digits.length; i++) {
        if (digits[i] == 0) return true;
    }
    return false;
}
```

}

OUTPUT:



LEVEL 3 QUESTION: 5

Q. Extend or Create a NumberChecker utility class and perform following task. Call from main() method the different methods and display results. Make sure all are static methods

```
public class NumberChecker {

   public static void main(String[] args) {

      int number = 7;


      boolean isPrime = isPrimeNumber(number);

      boolean isNeon = isNeonNumber(number);

      boolean isSpy = isSpyNumber(number);

      boolean isAutomorphic = isAutomorphicNumber(number);

      boolean isBuzz = isBuzzNumber(number);


      System.out.println("Number: " + number);

      System.out.println("Is Prime Number: " + isPrime);
```

```
    System.out.println("Is Neon Number: " + isNeon);
    System.out.println("Is Spy Number: " + isSpy);
    System.out.println("Is Automorphic Number: " + isAutomorphic);
    System.out.println("Is Buzz Number: " + isBuzz);
}

public static boolean isPrimeNumber(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 isNeonNumber(int number) {
    int square = number * number;
    int sum = 0;
    while (square > 0) {
        sum += square % 10;
        square /= 10;
    }
    return sum == number;
```

```
}

public static boolean isSpyNumber(int number) {
    int sum = 0;
    int product = 1;
    int temp = number;
    while (temp > 0) {
        int digit = temp % 10;
        sum += digit;
        product *= digit;
        temp /= 10;
    }
    return sum == product;
}

public static boolean isAutomorphicNumber(int number) {
    int square = number * number;
    return String.valueOf(square).endsWith(String.valueOf(number));
}

public static boolean isBuzzNumber(int number) {
    return number % 7 == 0 || number % 10 == 7;
```

```
    }
}
```

OUTPUT:

```
Number: 7
Is Prime Number: true
Is Neon Number: false
Is Spy Number: true
Is Automorphic Number: false
Is Buzz Number: true
```

LEVEL 3 QUESTION: 6

Q. Extend or Create a NumberChecker utility class and perform following task. Call from main() method the different methods and display results. Make sure all are static methods

```
public class NumberChecker {
    public static void main(String[] args) {
        int number = 28;

        int[] factors = getFactors(number);
        int greatestFactor = getGreatestFactor(factors);
        int sum = getSumOfFactors(factors);
        int product = getProductOfFactors(factors);
```

```
    double cubeProduct = getCubeProductOfFactors(factors);
    boolean isPerfect = isPerfectNumber(number);
    boolean isAbundant = isAbundantNumber(number);
    boolean isDeficient = isDeficientNumber(number);
    boolean isStrong = isStrongNumber(number);

    System.out.println("Number: " + number);
    System.out.print("Factors: ");
    for (int f : factors) {
        System.out.print(f + " ");
    }
    System.out.println("\nGreatest Factor: " + greatestFactor);
    System.out.println("Sum of Factors: " + sum);
    System.out.println("Product of Factors: " + product);
    System.out.println("Product of Cubes of Factors: " + cubeProduct);
    System.out.println("Is Perfect Number: " + isPerfect);
    System.out.println("Is Abundant Number: " + isAbundant);
    System.out.println("Is Deficient Number: " + isDeficient);
    System.out.println("Is Strong Number: " + isStrong);
}

public static int[] getFactors(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 getGreatestFactor(int[] factors) {
    int max = factors[0];
    for (int i = 1; i < factors.length; i++) {
        if (factors[i] > max) max = factors[i];
    }
    return max;
}
```

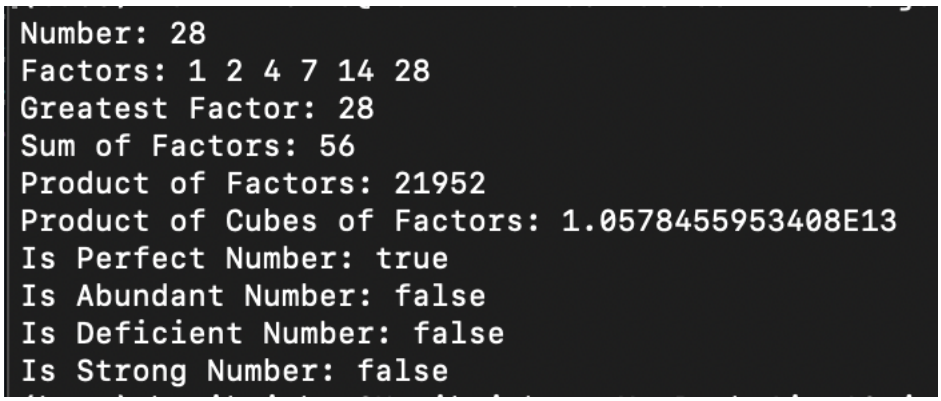
```
public static int getSumOfFactors(int[] factors) {
    int sum = 0;
    for (int factor : factors) {
        sum += factor;
    }
    return sum;
}

public static int getProductOfFactors(int[] factors) {
    int product = 1;
    for (int factor : factors) {
        product *= factor;
    }
    return product;
}

public static double getCubeProductOfFactors(int[] factors) {
    double product = 1;
    for (int factor : factors) {
        product *= Math.pow(factor, 3);
    }
```

```
    return product;
}

public static boolean isPerfectNumber(int number) {
    int sum = 0;
    for (int i = 1; i < number; i++) {
        if (number % i == 0) sum += i;
    }
    return sum == number;
}

public static boolean isAbundantNumber(int number) {
    int sum = 0;
    for (int i = 1; i < number; i++) {
        if (number % i == 0) sum += i;
    }
    return sum > number;
}

public static boolean isDeficientNumber(int number) {
    int sum = 0;
    for (int i = 1; i < number; i++) {
```

```
        if (number % i == 0) sum += i;
    }
    return sum < number;
}

public static boolean isStrongNumber(int number) {
    int sum = 0;
    int temp = number;
    while (temp > 0) {
        int digit = temp % 10;
        sum += factorial(digit);
        temp /= 10;
    }
    return sum == number;
}

public static int factorial(int n) {
    int fact = 1;
    for (int i = 2; i <= n; i++) {
        fact *= i;
    }
    return fact;
```

```
    }
}
```

OUTPUT:



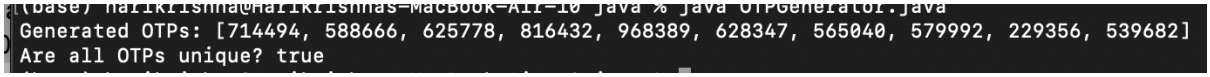
LEVEL 3 QUESTION: 7

Q. Write a program to generate a six-digit OTP number using Math.random() method. Validate the numbers are unique by generating the OTP number 10 times and ensuring all the 10 OTPs are not the same

```
import java.util.Arrays;

public class OTPGenerator {
    public static void main(String[] args) {
        int[] otps = new int[10];

        for (int i = 0; i < 10; i++) {
            otps[i] = generateOTP();
```

```
        }

        System.out.println("Generated OTPs: " + Arrays.toString(otps));
        boolean allUnique = areOTPsUnique(otps);
        System.out.println("Are all OTPs unique? " + allUnique);
    }

    public static int generateOTP() {
        return (int)(Math.random() * 900000) + 100000; // generates between 100000 and 999999
    }

    public static boolean areOTPsUnique(int[] otps) {
        for (int i = 0; i < otps.length; i++) {
            for (int j = i + 1; j < otps.length; j++) {
                if (otps[i] == otps[j]) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

OUTPUT:



LEVEL 3 QUESTION: 8

Q. Create a program to display a calendar for a given month and year. The program should take the month and year as input from the user and display the calendar for that month. E.g. for 07 2005 user input, the program should display the calendar as shown below

```
import java.util.Scanner;

public class CalendarDisplay { static final String[] MONTHS =
{ "January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December" };

static final int[] DAYS_IN_MONTH = {
  31, 28, 31, 30, 31, 30,
  31, 31, 30, 31, 30, 31
};

static boolean isLeapYear(int year) {
  return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
```

```
static int getDaysInMonth(int month, int year) {
  if (month == 1 && isLeapYear(year)) return 29;
  return DAYS_IN_MONTH[month];
}

static int getStartDay(int month, int year) {
  int y = year, m = month + 1, d = 1;
  int y0 = y - (14 - m) / 12;
  int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;
  int m0 = m + 12 * ((14 - m) / 12) - 2;
  return (d + x + (31 * m0) / 12) % 7;
}

static void displayCalendar(int month, int year) {
  System.out.printf("\n %s %d\n", MONTHS[month], year);
  System.out.println("Sun Mon Tue Wed Thu Fri Sat");
  int startDay = getStartDay(month, year);
  int numberOfDays = getDaysInMonth(month, year);
  for (int i = 0; i < startDay; i++) System.out.print("    ");
  for (int day = 1; day <= numberOfDays; day++) {
    System.out.printf("%3d ", day);
    if ((day + startDay) % 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() - 1;
    System.out.print("Enter year: ");
    int year = scanner.nextInt();
    displayCalendar(month, year);
}


}
```

OUTPUT:

```
Enter month (1-12): 12
Enter year: 2005

  December 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
```

LEVEL 3 QUESTION: 9

Q. Write a program Euclidean distance between two points as well as the equation of the line using those two points. Use Math functions Math.pow() and Math.sqrt()

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

```
public class LineGeometry {

    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 = findEuclideanDistance(x1, y1, x2, y2);
```

```
        double[] lineEquation = findLineEquation(x1, y1, x2, y2);

        System.out.printf("Euclidean Distance: %.2f\n", distance);
        System.out.printf("Line Equation: y = %.2fx + %.2f\n", lineEquation[0], lineEquation[1]);
    }

    public static double findEuclideanDistance(double x1, double y1, double x2, double y2) {
        return Math.sqrt(Math.pow((x2 - x1), 2) + Math.pow((y2 - y1), 2));
    }

    public static double[] findLineEquation(double x1, double y1, double x2, double y2) {
        double m = (y2 - y1) / (x2 - x1);
        double b = y1 - m * x1;
        return new double[] { m, b };
    }
```

}

OUTPUT:

```
Enter x1: 7
Enter y1: 4
Enter x2: 6
Enter y2: 8
Euclidean Distance: 4.12
Line Equation: y = -4.00x + 32.00
```

LEVEL 3 QUESTION: 10

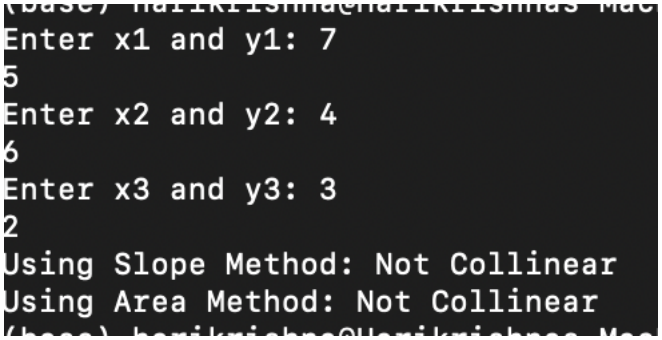
Q. Write a program to find the 3 points that are collinear using the slope formulae and area of triangle formulae. check A (2, 4), B (4, 6) and C (6, 8) are Collinear for sampling.

```
import java.util.Scanner;

public class CollinearPoints {

public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);

  System.out.print("Enter x1 and y1: ");
  int x1 = scanner.nextInt();
  int y1 = scanner.nextInt();

  System.out.print("Enter x2 and y2: ");
  int x2 = scanner.nextInt();
  int y2 = scanner.nextInt();
```

```
    System.out.print("Enter x3 and y3: ");
    int x3 = scanner.nextInt();
    int y3 = scanner.nextInt();

    boolean collinearBySlope = areCollinearBySlope(x1, y1, x2, y2, x3, y3);
    boolean collinearByArea = areCollinearByArea(x1, y1, x2, y2, x3, y3);

    System.out.println("Using Slope Method: " + (collinearBySlope ? "Collinear" : "Not Collinear"));
    System.out.println("Using Area Method: " + (collinearByArea ? "Collinear" : "Not Collinear"));
}

public static boolean areCollinearBySlope(int x1, int y1, int x2, int y2, int x3, int y3) {
    int slopeAB_num = (y2 - y1) * (x3 - x2);
    int slopeBC_num = (y3 - y2) * (x2 - x1);
    return slopeAB_num == slopeBC_num;
}

public static boolean areCollinearByArea(int x1, int y1, int x2, int y2, int x3, int y3) {
    double area = 0.5 * (x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2));
    return area == 0.0;
```

```
}

}
```

OUTPUT:



LEVEL 3 QUESTION: 11

Q. Create a program to find the bonus of 10 employees based on their years of service as well as the total bonus amount the 10-year-old company Zara has to pay as a bonus, along with the old and new salary.

```
import java.util.Random;

public class ZaraBonus {

   public static void main(String[] args) {
```

```
    int[][] salaryAndYears = determineSalaryAndService();
    int[][] newSalaryAndBonus = calculateNewSalaryAndBonus(salaryAndYears);
    calculateAndDisplaySum(salaryAndYears, newSalaryAndBonus);
  }

  public static int[][] determineSalaryAndService() {
    Random rand = new Random();
    int[][] salaryAndYears = new int[10][2];

    for (int i = 0; i < 10; i++) {
      salaryAndYears[i][0] = 10000 + rand.nextInt(90000);
      salaryAndYears[i][1] = 1 + rand.nextInt(20);
    }

    return salaryAndYears;
  }
```

```
  public static int[][] calculateNewSalaryAndBonus(int[][] salaryAndYears) {
    int[][] newSalaryAndBonus = new int[10][3];

    for (int i = 0; i < 10; i++) {
      int oldSalary = salaryAndYears[i][0];
      int yearsOfService = salaryAndYears[i][1];
      double bonusPercent = yearsOfService > 5 ? 0.05 : 0.02;
      int bonusAmount = (int) (oldSalary * bonusPercent);
      int newSalary = oldSalary + bonusAmount;

      newSalaryAndBonus[i][0] = oldSalary;
      newSalaryAndBonus[i][1] = newSalary;
      newSalaryAndBonus[i][2] = bonusAmount;
    }

    return newSalaryAndBonus;
  }
```

```
public static void calculateAndDisplaySum(int[][] salaryAndYears, int[][] newSalaryAndBonus) {
    int oldSalarySum = 0, newSalarySum = 0, totalBonus = 0;

    for (int i = 0; i < 10; i++) {
        oldSalarySum += salaryAndYears[i][0];
        newSalarySum += newSalaryAndBonus[i][1];
        totalBonus += newSalaryAndBonus[i][2];
    }

    System.out.println("Old Salary Sum: " + oldSalarySum);
    System.out.println("New Salary Sum: " + newSalarySum);
    System.out.println("Total Bonus: " + totalBonus);
    System.out.println("\nEmployee Report:");
    System.out.println("Old Salary | New Salary | Bonus");

    for (int i = 0; i < 10; i++) {
        System.out.println(newSalaryAndBonus[i][0] + " | " + newSalaryAndBonus[i][1] + " | " + newSalaryAndBonus[i][2]);
```

```
    }
  }
}
```

OUTPUT:

```
Employee Report:
Old Salary | New Salary | Bonus
24685 | 25919 | 1234
29887 | 31381 | 1494
14012 | 14292 | 280
43742 | 44616 | 874
54051 | 55132 | 1081
28681 | 30115 | 1434
29258 | 30720 | 1462
75994 | 79793 | 3799
34573 | 36301 | 1728
27741 | 29128 | 1387
```

LEVEL 3 QUESTION: 12

Q. Create a program to take input marks of students in 3 subjects physics, chemistry, and maths. Compute the total, average, and the percentage score

```
import java.util.Scanner;

import java.util.Random;


public class StudentScorecard {
```

```
static int[][] generateScores(int n) {
    Random rand = new Random();
    int[][] scores = new int[n][3];
    for (int i = 0; i < n; i++) {
        scores[i][0] = 40 + rand.nextInt(61);
        scores[i][1] = 40 + rand.nextInt(61);
        scores[i][2] = 40 + rand.nextInt(61);
    }
    return scores;
}

static double[][] calculateResults(int[][] scores) {
    int n = scores.length;
    double[][] results = new double[n][3];
    for (int i = 0; i < n; 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;
}

static String getGrade(double percentage) {
    if (percentage >= 80) return "A";
    else if (percentage >= 70) return "B";
    else if (percentage >= 60) return "C";
    else if (percentage >= 50) return "D";
    else if (percentage >= 40) return "E";
    else return "R";
}

static void displayScorecard(int[][] scores, double[][] results) {
```

```
System.out.println("ID\tPhysics\tChemistry\tMath\tTotal\tAverage\tPercentage\tGrade");
        for (int i = 0; i < scores.length; i++) {

System.out.printf("%d\t%d\t%d\t\t%d\t%.0f\t%.2f\t%.2f\t\t%s\n",
                    (i + 1),
                    scores[i][0],
                    scores[i][1],
                    scores[i][2],
                    results[i][0],
                    results[i][1],
                    results[i][2],
                    getGrade(results[i][2]));
        }
    }

    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        int[][] scores = generateScores(n);
        double[][] results = calculateResults(scores);
        displayScorecard(scores, results);
    }
}
```

OUTPUT:

```
Enter number of students: 50
ID      Physics Chemistry       Math    Total   Average Percentage      Grade
1       68      88              71      227     75.67   75.67           B
2       52      73              68      193     64.33   64.33           C
3       71      95              45      211     70.33   70.33           B
4       96      63              49      208     69.33   69.33           C
5       60      91              59      210     70.00   70.00           B
6       87      41              72      200     66.67   66.67           C
7       63      84              68      215     71.67   71.67           B
8       56      49              72      177     59.00   59.00           D
9       53      69              42      164     54.67   54.67           D
10      52      49              94      195     65.00   65.00           C
11      99      88              88      275     91.67   91.67           A
12      44      54              40      138     46.00   46.00           E
13      48      67              44      159     53.00   53.00           D
14      92      74              97      263     87.67   87.67           A
15      69      47              55      171     57.00   57.00           D
16      48      46              47      141     47.00   47.00           E
17      65      61              72      198     66.00   66.00           C
18      68      46              41      155     51.67   51.67           D
19      53      88              85      226     75.33   75.33           B
20      87      68              41      196     65.33   65.33           C
21      100     91              59      250     83.33   83.33           A
22      69      67              57      193     64.33   64.33           C
23      73      78              69      220     73.33   73.33           B
24      66      57              46      169     56.33   56.33           D
25      59      60              57      176     58.67   58.67           D
26      66      54              51      171     57.00   57.00           D
27      89      80              94      263     87.67   87.67           A
28      82      63              91      236     78.67   78.67           B
29      44      91              74      209     69.67   69.67           C
30      46      47              74      167     55.67   55.67           D
31      72      67              73      212     70.67   70.67           B
32      55      62              68      185     61.67   61.67           C
33      41      100             74      215     71.67   71.67           B
34      89      84              45      218     72.67   72.67           B
35      55      81              95      231     77.00   77.00           B
36      89      53              86      228     76.00   76.00           B
37      53      89              60      202     67.33   67.33           C
38      68      59              97      224     74.67   74.67           B
39      99      53              47      199     66.33   66.33           C
40      47      50              60      157     52.33   52.33           D
41      67      83              47      197     65.67   65.67           C
42      53      82              54      189     63.00   63.00           C
43      68      61              58      187     62.33   62.33           C
44      91      99              43      233     77.67   77.67           B
45      76      55              76      207     69.00   69.00           C
46      64      53              93      210     70.00   70.00           B
47      84      49              92      225     75.00   75.00           B
48      75      63              89      227     75.67   75.67           B
49      77      57              99      233     77.67   77.67           B
50      41      60              65      166     55.33   55.33           D
```

LEVEL 3 QUESTION: 13

Q. Write a program to perform matrix manipulation operations like addition, subtraction, multiplication, and transpose. Also finding the determinant and inverse of a matrix. The program should take random matrices as input and display the result of the operations.

```
import java.util.Random;

public 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 int[][] addMatrices(int[][] a, int[][] b) {
  int[][] result = new int[a.length][a[0].length];
  for (int i = 0; i < a.length; i++)
    for (int j = 0; j < a[0].length; j++)
      result[i][j] = a[i][j] + b[i][j];
  return result;
}

public static int[][] subtractMatrices(int[][] a, int[][] b) {
  int[][] result = new int[a.length][a[0].length];
  for (int i = 0; i < a.length; i++)
    for (int j = 0; j < a[0].length; j++)
      result[i][j] = a[i][j] - b[i][j];
  return result;
}
```

```
public static int[][] multiplyMatrices(int[][] a, int[][] b) {
  int[][] result = new int[a.length][b[0].length];
  for (int i = 0; i < a.length; i++)
    for (int j = 0; j < b[0].length; j++)
      for (int k = 0; k < a[0].length; k++)
        result[i][j] += a[i][k] * b[k][j];
  return result;
}

public static int[][] transposeMatrix(int[][] matrix) {
  int[][] transposed = new int[matrix[0].length][matrix.length];
  for (int i = 0; i < matrix.length; i++)
    for (int j = 0; j < matrix[0].length; j++)
      transposed[j][i] = matrix[i][j];
  return transposed;
}

public static int determinant2x2(int[][] matrix) {
  return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
}

public static int determinant3x3(int[][] m) {
  return m[0][0] * (m[1][1] * m[2][2] - m[1][2] * m[2][1])
    - m[0][1] * (m[1][0] * m[2][2] - m[1][2] * m[2][0])
    + m[0][2] * (m[1][0] * m[2][1] - m[1][1] * m[2][0]);
}
```

```
public static double[][] inverse2x2(int[][] m) {
  int det = determinant2x2(m);
  if (det == 0) return null;
  double[][] inv = new double[2][2];
  inv[0][0] = m[1][1] / (double) det;
  inv[0][1] = -m[0][1] / (double) det;
  inv[1][0] = -m[1][0] / (double) det;
  inv[1][1] = m[0][0] / (double) det;
  return inv;
}

public static double[][] inverse3x3(int[][] m) {
  int det = determinant3x3(m);
  if (det == 0) return null;
  double[][] inv = new double[3][3];
  inv[0][0] = (m[1][1]*m[2][2] - m[1][2]*m[2][1]) / (double) det;
  inv[0][1] = (m[0][2]*m[2][1] - m[0][1]*m[2][2]) / (double) det;
  inv[0][2] = (m[0][1]*m[1][2] - m[0][2]*m[1][1]) / (double) det;
  inv[1][0] = (m[1][2]*m[2][0] - m[1][0]*m[2][2]) / (double) det;
  inv[1][1] = (m[0][0]*m[2][2] - m[0][2]*m[2][0]) / (double) det;
  inv[1][2] = (m[0][2]*m[1][0] - m[0][0]*m[1][2]) / (double) det;
  inv[2][0] = (m[1][0]*m[2][1] - m[1][1]*m[2][0]) / (double) det;
  inv[2][1] = (m[0][1]*m[2][0] - m[0][0]*m[2][1]) / (double) det;
  inv[2][2] = (m[0][0]*m[1][1] - m[0][1]*m[1][0]) / (double) det;
  return inv;
}
```

```
public static void displayMatrix(int[][] matrix) {
  for (int[] row : matrix) {
    for (int val : row)
      System.out.print(val + " ");
    System.out.println();
  }
}

public static void displayMatrix(double[][] matrix) {
  for (double[] row : matrix) {
    for (double val : row)
      System.out.printf("%.2f ", val);
    System.out.println();
  }
}

public static void main(String[] args) {
  int[][] matrixA = createRandomMatrix(3, 3);
  int[][] matrixB = createRandomMatrix(3, 3);

  System.out.println("Matrix A:");
  displayMatrix(matrixA);
  System.out.println("Matrix B:");
  displayMatrix(matrixB);

  System.out.println("A + B:");
  displayMatrix(addMatrices(matrixA, matrixB));
```

```
    System.out.println("A - B:");
    displayMatrix(subtractMatrices(matrixA, matrixB));

    System.out.println("A * B:");
    displayMatrix(multiplyMatrices(matrixA, matrixB));

    System.out.println("Transpose of A:");
    displayMatrix(transposeMatrix(matrixA));

    System.out.println("Determinant of A:");
    System.out.println(determinant3x3(matrixA));

    System.out.println("Inverse of A:");
    double[][] invA = inverse3x3(matrixA);
    if (invA != null)
      displayMatrix(invA);
    else
      System.out.println("Matrix A is non-invertible.");
  }


}
```

OUTPUT:

```
Matrix A:
9 3 0
5 0 3
5 9 2
Matrix B:
2 4 1
8 9 8
5 4 4
A + B:
11 7 1
13 9 11
10 13 6
A - B:
7 -1 -1
-3 -9 -5
0 5 -2
A * B:
42 63 33
25 32 17
92 109 85
Transpose of A:
9 5 5
3 0 9
0 3 2
Determinant of A:
-228
Inverse of A:
0.12 0.03 -0.04
-0.02 -0.08 0.12
-0.20 0.29 0.07
```