Level 3 Practice Programs

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

   Hint =>

   a. The formula to calculate the mean is: mean = sum of all elements/number of elements
   b. Create an int array named heights of size 11 and get 3 digits random height in cms for each player in the range 150 cms to 250 cms
   c. Write the method to Find the sum of all the elements present in the array.
   d. Write the method to find the mean height of the players on the football team
   e. Write the method to find the shortest height of the players on the football team
   f. Write the method to find the tallest height of the players on the football team
   g. Finally display the results

Code

```
import java.util.Random;

public class FootballTeamStats {

    // Method to generate random heights between 150 and 250 for 11 players
    public static int[] generateHeights() {
        int[] heights = new int[11];
        Random rand = new Random();
        for (int i = 0; i < heights.length; i++) {
            heights[i] = rand.nextInt(101) + 150; // 150 to 250
        }
        return heights;
    }

    // Method to calculate the sum of the elements in the array
    public static int calculateSum(int[] arr) {
        int sum = 0;
        for (int height : arr) {
            sum += height;
        }
        return sum;
    }

    // Method to calculate the mean height
    public static double calculateMean(int[] arr) {
        return (double) calculateSum(arr) / arr.length;
    }

    // Method to find the shortest height
    public static int findShortest(int[] arr) {
```

```
        int min = arr[0];
        for (int height : arr) {
            if (height < min) {
                min = height;
            }
        }
        return min;
    }

    // Method to find the tallest height
    public static int findTallest(int[] arr) {
        int max = arr[0];
        for (int height : arr) {
            if (height > max) {
                max = height;
            }
        }
        return max;
    }

    public static void main(String[] args) {
        int[] heights = generateHeights();

        System.out.println("Player Heights:");
        for (int i = 0; i < heights.length; i++) {
            System.out.println("Player " + (i + 1) + ": " + heights[i] + " cm");
        }

        int shortest = findShortest(heights);
        int tallest = findTallest(heights);
        double mean = calculateMean(heights);

        System.out.println("\nShortest Height: " + shortest + " cm");
        System.out.println("Tallest Height: " + tallest + " cm");
        System.out.printf("Mean Height: %.2f cm\n", mean);
    }
}
```

2. 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

Hint =>

a. Method to Find the count of digits in the number
b. Method to Store the digits of the number in a digits array
c. Method to Check if a number is a duck number using the digits array. A duck number is a number that has a non-zero digit present in it

d. Method to check if the number is a armstrong number using the digits array. Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits. Eg: 153 = 1^3 + 5^3 + 3^3
e. Method to find the largest and second largest elements in the digits array. Use ***Integer.MIN_VALUE*** to initialize the variable.
f. Method to find the the smallest and second smallest elements in the digits array. Use ***Integer.MAX_VALUE*** to initialize the variable

Code

```
public class NumberChecker {

    // Method to count the digits in a number
    public static int countDigits(int number) {
        return String.valueOf(Math.abs(number)).length();
    }

    // Method to store the digits in an array
    public static int[] getDigitsArray(int number) {
        String numStr = String.valueOf(Math.abs(number));
        int[] digits = new int[numStr.length()];
        for (int i = 0; i < numStr.length(); i++) {
            digits[i] = numStr.charAt(i) - '0';
        }
        return digits;
    }

    // Method to check if the number is a duck number
    public static boolean isDuckNumber(int[] digits) {
        for (int digit : digits) {
            if (digit == 0) {
                return true;
            }
        }
        return false;
    }
```

```
// Method to check if the number is an Armstrong number
public static boolean isArmstrongNumber(int number, int[] digits) {
    int power = digits.length;
    int sum = 0;
    for (int digit : digits) {
        sum += Math.pow(digit, power);
    }
    return sum == number;
}

// Method to find the largest and second largest digits
public static void findTwoLargest(int[] digits) {
    int max = Integer.MIN_VALUE;
    int secondMax = Integer.MIN_VALUE;

    for (int digit : digits) {
        if (digit > max) {
            secondMax = max;
            max = digit;
        } else if (digit > secondMax && digit != max) {
            secondMax = digit;
        }
    }

    System.out.println("Largest digit: " + max);
    System.out.println("Second largest digit: " + secondMax);
}

// Method to find the smallest and second smallest digits
public static void findTwoSmallest(int[] digits) {
    int min = Integer.MAX_VALUE;
    int secondMin = Integer.MAX_VALUE;
```

```
        for (int digit : digits) {
            if (digit < min) {
                secondMin = min;
                min = digit;
            } else if (digit < secondMin && digit != min) {
                secondMin = digit;
            }
        }

        System.out.println("Smallest digit: " + min);
        System.out.println("Second smallest digit: " + secondMin);
    }

    // Main method to test all functionalities
    public static void main(String[] args) {
        int number = 153; // Change this to test with different numbers

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

        int digitCount = countDigits(number);
        System.out.println("Digit Count: " + digitCount);

        int[] digits = getDigitsArray(number);

        System.out.print("Digits Array: ");
        for (int d : digits) {
            System.out.print(d + " ");
        }
        System.out.println();

        System.out.println("Is Duck Number? " + isDuckNumber(digits));
        System.out.println("Is Armstrong Number? " + isArmstrongNumber(number,
digits));
```

```
        findTwoLargest(digits);
        findTwoSmallest(digits);
    }
}
```

3. 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

Hint =>

a. Method to find the count of digits in the number and a Method to Store the digits of the number in a digits array
b. Method to find the sum of the digits of a number using the digits array
c. Method to find the sum of the squares of the digits of a number using the digits array. Use ***Math.pow()*** method
d. Method to Check if a number is a harshad number using a digits array. A number is called a Harshad number if it is divisible by the sum of its digits. For e.g. 21
e. Method to find the frequency of each digit in the number. Create a 2D array to store the frequency with digit in the first column and frequency in the second column.

Code

```
public class NumberChecker {


    // Method to count the number of digits
    public static int countDigits(int number) {
        return String.valueOf(Math.abs(number)).length();
    }


    // Method to store digits in an array
    public static int[] getDigitsArray(int number) {
        String numStr = String.valueOf(Math.abs(number));
        int[] digits = new int[numStr.length()];
        for (int i = 0; i < numStr.length(); i++) {
            digits[i] = numStr.charAt(i) - '0';
        }
        return digits;
    }


    // Method to find sum of digits
    public static int sumOfDigits(int[] digits) {
```

```
    int sum = 0;
    for (int digit : digits) {
        sum += digit;
    }
    return sum;
}

// Method to find sum of squares of digits
public static int sumOfSquares(int[] digits) {
    int sum = 0;
    for (int digit : digits) {
        sum += Math.pow(digit, 2);
    }
    return sum;
}

// Method to check if number is Harshad (Niven) number
public static boolean isHarshadNumber(int number, int[] digits) {
    int sum = sumOfDigits(digits);
    return number % sum == 0;
}

// Method to find frequency of each digit
public static int[][] digitFrequency(int[] digits) {
    int[] freq = new int[10]; // digits 0-9
    for (int digit : digits) {
        freq[digit]++;
    }

    // Create 2D array: digit, frequency
    int[][] result = new int[10][2];
    for (int i = 0; i < 10; i++) {
        result[i][0] = i;       // digit
        result[i][1] = freq[i]; // frequency
```

```
        }
        return result;
    }

    public static void main(String[] args) {
        int number = 21; // change this number to test others

        System.out.println("Number: " + number);
        int digitCount = countDigits(number);
        System.out.println("Digit Count: " + digitCount);

        int[] digits = getDigitsArray(number);
        System.out.print("Digits: ");
        for (int d : digits) {
            System.out.print(d + " ");
        }
        System.out.println();

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

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

        boolean isHarshad = isHarshadNumber(number, digits);
        System.out.println("Is Harshad Number? " + isHarshad);

        System.out.println("Digit Frequency:");
        int[][] frequency = digitFrequency(digits);
        for (int i = 0; i < frequency.length; i++) {
            if (frequency[i][1] > 0) {
                System.out.println("Digit " + frequency[i][0] + " => " + frequency[i][1] + "
time(s)");
            }
```

```
        }
    }
}
```

4.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

Hint =>

a. Method to find the count of digits in the number and a Method to Store the digits of the number in a digits array
b. Method to reverse the digits array
c. Method to compare two arrays and check if they are equal
d. Method to check if a number is a palindrome using the Digits. A palindrome number is a number that remains the same when its digits are reversed.
e. Method to Check if a number is a duck number using the digits array. A duck number is a number that has a non-zero digit present in it

Code

```
import java.util.Arrays;


public class NumberChecker {


    // Method to count the number of digits
    public static int countDigits(int number) {
        return String.valueOf(Math.abs(number)).length();
    }


    // Method to store digits in an array
    public static int[] getDigitsArray(int number) {
        String numStr = String.valueOf(Math.abs(number));
        int[] digits = new int[numStr.length()];
        for (int i = 0; i < numStr.length(); i++) {
            digits[i] = numStr.charAt(i) - '0';
        }
        return digits;
    }


    // Method to reverse the digits array
```

```
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;
}

// Method to compare two arrays
public static boolean arraysEqual(int[] arr1, int[] arr2) {
    return Arrays.equals(arr1, arr2);
}

// Method to check if number is a palindrome
public static boolean isPalindrome(int number) {
    int[] digits = getDigitsArray(number);
    int[] reversed = reverseArray(digits);
    return arraysEqual(digits, reversed);
}

// Method to check if number is a duck number
public static boolean isDuckNumber(int[] digits) {
    for (int digit : digits) {
        if (digit == 0) {
            return true;
        }
    }
    return false;
}

// Main method to test
public static void main(String[] args) {
    int number = 1210; // Change this to test different numbers
```

```
        System.out.println("Number: " + number);
        System.out.println("Digit Count: " + countDigits(number));

        int[] digits = getDigitsArray(number);
        System.out.print("Digits: ");
        for (int d : digits) {
            System.out.print(d + " ");
        }

        int[] reversed = reverseArray(digits);
        System.out.print("\nReversed Digits: ");
        for (int d : reversed) {
            System.out.print(d + " ");
        }

        System.out.println("\nArrays Equal? " + arraysEqual(digits, reversed));
        System.out.println("Is Palindrome? " + isPalindrome(number));
        System.out.println("Is Duck Number? " + isDuckNumber(digits));
    }
}
```

5. 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

Hint =>

a. Method to Check if a number is prime number. A prime number is a number greater than 1 that has no positive divisors other than 1 and itself.
b. Method to Check if a number is a neon number. A neon number is a number where the sum of digits of the square of the number is equal to the number itself
c. Method to Check if a number is a spy number. A number is called a spy number if the sum of its digits is equal to the product of its digits
d. Method to Check if a number is an automorphic number. An automorphic number is a number whose square ends with the number itself. E.g. 5 is an automorphic number
e. Method to Check if a number is a buzz number. A buzz number is a number that is either divisible by 7 or ends with 7

Code

```
public class NumberChecker {
```

```
// Method to check if a number is prime
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;
}

// Method to check if a number is a neon number
public static boolean isNeon(int number) {
    int square = number * number;
    int sum = 0;
    while (square > 0) {
        sum += square % 10;
        square /= 10;
    }
    return sum == number;
}

// Method to check if a number is a spy number
public static boolean isSpy(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;
```

```
    }

    // Method to check if a number is an automorphic number
    public static boolean isAutomorphic(int number) {
        int square = number * number;
        String numStr = String.valueOf(number);
        String squareStr = String.valueOf(square);
        return squareStr.endsWith(numStr);
    }

    // Method to check if a number is a buzz number
    public static boolean isBuzz(int number) {
        return (number % 7 == 0) || (number % 10 == 7);
    }

    // Main method to test
    public static void main(String[] args) {
        int number = 7; // Try changing this to test other numbers

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

6. 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

Hint =>

a. Method to find factors of a number and return them as an array. Note there are 2 for loops one for the count and another for finding the factor and storing in the array
b. Method to find the greates factor of a Number using the factors array
c. Method to find the sum of the factors using factors array and return the sum
d. Method to find the product of the factors using factors array and return the product

e. Method to find product of cube of the factors using the factors array. Use ***Math.pow()***
f. Method to Check if a number is a perfect number. Perfect numbers are positive integers that are equal to the sum of their proper divisors
g. Method to find the number is a abundant number. A number is called an abundant number if the sum of its proper divisors is greater than the number itself
h. Method to find the number is a deficient number. A number is called a deficient number if the sum of its proper divisors is less than the number itself
i. Method to Check if a number is a strong number. A number is called a strong number if the sum of the factorial of its digits is equal to the number itself

Code

```
import java.util.Arrays;


public class NumberChecker {


    // Method to find all factors of a number and return them as array
    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;
    }


    // Method to get greatest factor (excluding the number itself)
    public static int getGreatestFactor(int[] factors, int number) {
        int max = 0;
        for (int f : factors) {
```

```
        if (f != number && f > max) {
            max = f;
        }
    }
    return max;
}

// Method to find sum of factors
public static int sumOfFactors(int[] factors) {
    int sum = 0;
    for (int f : factors) {
        sum += f;
    }
    return sum;
}

// Method to find product of factors
public static long productOfFactors(int[] factors) {
    long product = 1;
    for (int f : factors) {
        product *= f;
    }
    return product;
}

// Method to find product of cube of factors
public static double cubeProductOfFactors(int[] factors) {
    double product = 1;
    for (int f : factors) {
        product *= Math.pow(f, 3);
    }
    return product;
}
```

```
// Check if perfect number (sum of proper divisors == number)
public static boolean isPerfectNumber(int number, int[] factors) {
    int sum = 0;
    for (int f : factors) {
        if (f != number) sum += f;
    }
    return sum == number;
}

// Check if abundant number (sum of proper divisors > number)
public static boolean isAbundant(int number, int[] factors) {
    int sum = 0;
    for (int f : factors) {
        if (f != number) sum += f;
    }
    return sum > number;
}

// Check if deficient number (sum of proper divisors < number)
public static boolean isDeficient(int number, int[] factors) {
    int sum = 0;
    for (int f : factors) {
        if (f != number) sum += f;
    }
    return sum < number;
}

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

```
    }

    // Check if strong number (sum of factorial of digits == 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;
    }

    // Main method to test
    public static void main(String[] args) {
        int number = 28; // Change this to test other numbers

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

        int[] factors = getFactors(number);
        System.out.println("Factors: " + Arrays.toString(factors));
        System.out.println("Greatest Factor (excluding self): " +
getGreatestFactor(factors, number));
        System.out.println("Sum of Factors: " + sumOfFactors(factors));
        System.out.println("Product of Factors: " + productOfFactors(factors));
        System.out.println("Product of Cubes of Factors: " +
cubeProductOfFactors(factors));

        System.out.println("Is Perfect Number? " + isPerfectNumber(number, factors));
        System.out.println("Is Abundant Number? " + isAbundant(number, factors));
        System.out.println("Is Deficient Number? " + isDeficient(number, factors));
        System.out.println("Is Strong Number? " + isStrongNumber(number));
    }
```

```
}
```

7. 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

Hint =>

a. Write a method to Generate a 6-digit OTP number using Math.random()
b. Create an array to save the OTP numbers generated 10 times
c. Write a method to ensure that the OTP numbers generated are unique. If unique return true else return false

Code

```
import java.util.Arrays;


public class OTPGenerator {


    // Method to generate a 6-digit OTP using Math.random()
    public static int generateOTP() {
        return (int)(100000 + Math.random() * 900000); // 100000 to 999999
    }


    // Method to generate and return an array of 10 OTPs
    public static int[] generateMultipleOTPs(int count) {
        int[] otps = new int[count];
        for (int i = 0; i < count; i++) {
            otps[i] = generateOTP();
        }
        return otps;
    }


    // Method to check if all OTPs in the array are unique
    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;
}

// Main method
public static void main(String[] args) {
    int[] otps = generateMultipleOTPs(10);

    System.out.println("Generated OTPs:");
    for (int otp : otps) {
        System.out.println(otp);
    }

    System.out.println("Are all OTPs unique? " + areOTPsUnique(otps));
}
}
```

8. 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

```
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
```

Hint =>

a. Write a Method to get the name of the month. For this define a month Array to store the names of the months
b. Write a Method to get the number of days in the month. For this define a days Array to store the number of days in each month. For Feb month, check for Leap Year to get the number of days. Also, define a Leap Year Method.
c. Write a method to get the first day of the month using the Gregorian calendar algorithm

   y0 = y − (14 − m) / 12

x = y0 + y0/4 − y0/100 + y0/400

m0 = m + 12 × ((14 − m) / 12) − 2

d0 = (d + x + 31m0 / 12) mod 7

d. Displaying the Calendar requires 2 ***for*** loops.
   i. The first ***for*** loop up to the first day to get the proper indentation. As in the example above 3 spaces from Sun to Thu as to be set as July 1st starts on Fri
   ii. The Second ***for*** loop Displays the days of the month starting from 1 to the number of days. Add proper indentation for single-digit days using ***%3d*** to display the integer right-justified in a field of width 3. Please note to move to the next line after Sat

Code

```
import java.util.Scanner;


public class CalendarDisplay {


    // Array of month names
    static String[] months = {
        "", "January", "February", "March", "April", "May", "June",
        "July", "August", "September", "October", "November", "December"
    };


    // Array of days in each month
    static int[] daysInMonth = {
        0, 31, 28, 31, 30, 31, 30,
        31, 31, 30, 31, 30, 31
    };


    // Check for leap year
    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }


    // Get first day of the month using Gregorian calendar algorithm
    public static int getFirstDayOfMonth(int month, int year) {
        int d = 1; // First day of month
        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 = (d + x + (31 * m0) / 12) % 7;
    return d0; // 0=Sunday, 1=Monday, ..., 6=Saturday
}

// Print the calendar
public static void printCalendar(int month, int year) {
    if (month < 1 || month > 12) {
        System.out.println("Invalid month.");
        return;
    }

    int days = daysInMonth[month];

    // Adjust for leap year if February
    if (month == 2 && isLeapYear(year)) {
        days = 29;
    }

    // Print calendar header
    System.out.printf("     %s %d\n", months[month], year);
    System.out.println(" Su Mo Tu We Th Fr Sa");

    int startDay = getFirstDayOfMonth(month, year);

    // Print initial spaces
    for (int i = 0; i < startDay; i++) {
        System.out.print("   ");
    }

    // Print days
    for (int day = 1; day <= days; day++) {
        System.out.printf("%3d", day);
```

```
            if ((day + startDay) % 7 == 0 || day == days) {
                System.out.println();
            }
        }
    }

    // Main method
    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();

        printCalendar(month, year);
    }
}
```

9 .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()***

Hint =>

a. Take inputs for 2 points x1, y1, and x2, y2
b. Method to find the Euclidean distance between two points and return the distance

   $distance = \sqrt{(x2 - x1)^2} + \sqrt{(y2 - y1)^2}$

c. Write a Method to find the equation of a line given two points and return the equation which includes the slope and the y-intercept

   The equation of a line is given by the equation $y = m * x + b$ Where m is the slope and b is the y-intercept. So firstly compute the slope using the formulae

   $m = (y2 - y1)/(x2 - x1)$

   Post that compute the y-intercept b using the formulae

   $b = y1 - m * x1$

   Finally, return an array having slope m and y-intercept b

   Code

```
import java.util.Scanner;


public class LineGeometry {
```

```
    // Method to calculate Euclidean distance
    public static double calculateDistance(double x1, double y1, double x2, double y2)
{
        double distance = Math.sqrt(Math.pow((x2 - x1), 2) + Math.pow((y2 - y1), 2));
        return distance;
    }

    // Method to calculate slope and y-intercept
    public static double[] calculateLineEquation(double x1, double y1, double x2,
double y2) {
        double[] result = new double[2]; // index 0 = slope, index 1 = intercept

        if (x1 == x2) {
            throw new ArithmeticException("Slope is undefined for vertical lines.");
        }

        double m = (y2 - y1) / (x2 - x1);     // Slope
        double b = y1 - m * x1;               // y-intercept
        result[0] = m;
        result[1] = b;

        return result;
    }

    // Main method
    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 = calculateDistance(x1, y1, x2, y2);
        System.out.printf("Euclidean Distance: %.2f\n", distance);

        try {
            double[] lineEquation = calculateLineEquation(x1, y1, x2, y2);
            System.out.printf("Equation of the line: y = %.2fx + %.2f\n", lineEquation[0],
    lineEquation[1]);
        } catch (ArithmeticException e) {
            System.out.println("Line Equation: Vertical Line (Slope is undefined)");
        }
    }
}
```

10 .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.

Hint =>

a. Take inputs for 3 points x1, y1, x2, y2, and x3, y3
b. Write a Method to find the 3 points that are collinear using the slope formula. The 3 points A(x1,y1), b(x2,y2), and c(x3,y3) are collinear if the slopes formed by 3 points ab, bc, and cd are equal.

   $slope\ AB\ =\ (y2\ -\ y1)/(x2\ -\ x1)$, $slope\ BC\ =\ (y3\ -\ y2)/(x3\ -\ x3)$

   $slope\ AC\ =\ (y3\ -\ y1)/(x3\ -\ x1)$ Points are collinear if

   $slope\ AB\ =\ slope\ BC\ =\ slope\ Ac$
c. The method to find the three points is collinear using the area of the triangle formula. The Three points are collinear if the area of the triangle formed by three points is 0. The area of a triangle is

$$\frac{1}{2}\begin{vmatrix} x_1 - x_2 & x_2 - x_3 \\ y_1 - y_2 & y_2 - y_3 \end{vmatrix}$$

area of triangle formula

$$\frac{1}{2}\begin{vmatrix} 2-4 & 4-6 \\ 4-6 & 6-8 \end{vmatrix} = \frac{1}{2}\begin{vmatrix} -2 & -2 \\ -2 & -2 \end{vmatrix} = \frac{1}{2}(4-4) = 0$$

$area = 0.5 * (x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2))$

Code

```
import java.util.Scanner;

public class CollinearityChecker {

    // Method to check collinearity using slope formula
    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) && (slopeBC == slopeAC);
    }

    // Method to check collinearity using area of triangle formula
    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 == 0.0;
    }

    // Main method
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

determinant $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

```
        System.out.println("Enter coordinates of point A (x1 y1): ");
        double x1 = sc.nextDouble(), y1 = sc.nextDouble();

        System.out.println("Enter coordinates of point B (x2 y2): ");
        double x2 = sc.nextDouble(), y2 = sc.nextDouble();

        System.out.println("Enter coordinates of point C (x3 y3): ");
        double x3 = sc.nextDouble(), y3 = sc.nextDouble();

        boolean collinearSlope = areCollinearBySlope(x1, y1, x2, y2, x3, y3);
        boolean collinearArea = areCollinearByArea(x1, y1, x2, y2, x3, y3);

        System.out.println("\nUsing Slope Method: " + (collinearSlope ? "Collinear" :
    "Not Collinear"));
        System.out.println("Using Area Method: " + (collinearArea ? "Collinear" : "Not
    Collinear"));
    }
}
```

11 .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.

Hint =>

a. Zara decides to give a bonus of 5% to employees whose year of service is more than 5 years or 2% if less than 5 years
b. Create a Method to determine the Salary and years of service and return the same. Use the ***Math.random()*** method to determine the 5-digit salary for each employee and also use the random method to determine the years of service. Define 2D Array to save the salary and years of service.
c. Write a Method to calculate the new salary and bonus based on the logic defined above and return the new 2D Array of the latest salary and bonus amount
d. Write a Method to Calculate the sum of the Old Salary, the Sum of the New Salary, and the Total Bonus Amount and display it in a Tabular Format

Code

```
public class EmployeeBonusCalculator {


  // Method to generate salary and years of service
```

```
    public static int[][] generateEmployeeData(int numEmployees) {
        int[][] data = new int[numEmployees][2]; // [][0] = salary, [][1] = years

        for (int i = 0; i < numEmployees; i++) {
            int salary = 10000 + (int)(Math.random() * 90000); // 5-digit salary
            int years = (int)(Math.random() * 11);              // 0 to 10 years
            data[i][0] = salary;
            data[i][1] = years;
        }

        return data;
    }

    // Method to calculate new salary and bonus
    public static double[][] calculateBonusAndNewSalary(int[][] employeeData) {
        double[][] result = new double[employeeData.length][2]; // [][0] = newSalary, [][1] =
bonus

        for (int i = 0; i < employeeData.length; i++) {
            int oldSalary = employeeData[i][0];
            int years = employeeData[i][1];
            double bonusRate = (years > 5) ? 0.05 : 0.02;
            double bonus = oldSalary * bonusRate;
            double newSalary = oldSalary + bonus;

            result[i][0] = newSalary;
            result[i][1] = bonus;
        }

        return result;
    }

    // Method to display the table and totals
    public static void displayResults(int[][] employeeData, double[][] bonusData) {
```

```
    double totalOldSalary = 0;
    double totalNewSalary = 0;
    double totalBonus = 0;

    System.out.printf("%-10s %-10s %-12s %-12s %-12s\n",
                    "Emp ID", "Service", "Old Salary", "Bonus", "New Salary");
    System.out.println("-------------------------------------------------------------");

    for (int i = 0; i < employeeData.length; i++) {
        int oldSalary = employeeData[i][0];
        int years = employeeData[i][1];
        double bonus = bonusData[i][1];
        double newSalary = bonusData[i][0];

        System.out.printf("%-10d %-10d %-12.2f %-12.2f %-12.2f\n",
                        (i+1), years, (double)oldSalary, bonus, newSalary);

        totalOldSalary += oldSalary;
        totalBonus += bonus;
        totalNewSalary += newSalary;
    }

    System.out.println("-------------------------------------------------------------");
    System.out.printf("%-21s %-12.2f %-12.2f %-12.2f\n",
                    "TOTAL", totalOldSalary, totalBonus, totalNewSalary);
}

// Main Method
public static void main(String[] args) {
    int numEmployees = 10;

    int[][] employeeData = generateEmployeeData(numEmployees);
    double[][] bonusData = calculateBonusAndNewSalary(employeeData);
    displayResults(employeeData, bonusData);
```

```
    }
}
```

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

Grade	Remarks	Marks
A	(Level 4, above agency-normalized standards)	80% and above
B	(Level 3, at agency-normalized standards)	70-79%
C	(Level 2, below, but approaching agency-normalized standards)	60-69%
D	(Level 1, well below agency-normalized standards)	50-59%
E	(Level 1- , too below agency-normalized standards)	40-49%
R	(Remedial standards)	39% and below

Hint =>

a. Take input for the number of students
b. Write a method to generate random 2-digit scores for Physics, Chemistry, and Math (PCM) for the students and return the scores. This method returns a 2D array with PCM scores for all students
c. Write a Method to calculate the total, average, and percentages for each student and return a 2D array with the corresponding values. Please ensure to round off the values to 2 Digits using the ***Math.round()*** method.
d. Finally, write a Method to display the scorecard of all students with their scores, total, average, and percentage in a tabular format using ***"\t"***.

Code

```
import java.util.Scanner;


public class StudentScorecard {


    // Method to generate random 2-digit PCM scores
    public static int[][] generateScores(int numStudents) {
        int[][] scores = new int[numStudents][3]; // [][0]=Phy, [][1]=Chem, [][2]=Math


        for (int i = 0; i < numStudents; i++) {
            scores[i][0] = 10 + (int)(Math.random() * 90); // Physics
            scores[i][1] = 10 + (int)(Math.random() * 90); // Chemistry
            scores[i][2] = 10 + (int)(Math.random() * 90); // Math
```

```
        }
        return scores;
    }

    // Method to calculate total, average and percentage
    public static double[][] calculateResults(int[][] scores) {
        int n = scores.length;
        double[][] results = new double[n][3]; // [][0]=Total, [][1]=Average, 
[][2]=Percentage

        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;

            // Round to 2 decimal places
            results[i][0] = Math.round(total * 100.0) / 100.0;
            results[i][1] = Math.round(average * 100.0) / 100.0;
            results[i][2] = Math.round(percentage * 100.0) / 100.0;
        }
        return results;
    }

    // Method to display scorecard
    public static void displayScorecard(int[][] scores, double[][] results) {

System.out.println("ID\tPhysics\tChemistry\tMaths\tTotal\tAverage\tPercentage");
        System.out.println("-------------------------------------------------------------");

        for (int i = 0; i < scores.length; i++) {
            System.out.print((i + 1) + "\t" + scores[i][0] + "\t" + scores[i][1] + "\t\t" + 
scores[i][2] + "\t");
            System.out.print(results[i][0] + "\t" + results[i][1] + "\t" + results[i][2] + "\n");
        }
    }
```

```
        // Main Method
        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);
        }
    }
```

13.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.

Hint =>

a. Write a Method to create a random matrix taking rows and columns as parameters
b. Write a Method to add two matrices
c. Write a Method to subtract two matrices
d. Write a Method to multiply two matrices

Rectangular matrices [edit]

If

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ x & y & z \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \alpha & \rho \\ \beta & \sigma \\ \gamma & \tau \end{pmatrix},$$

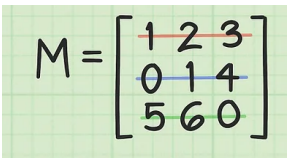
their matrix products are:

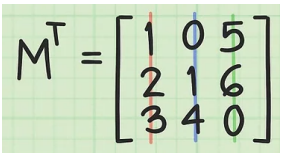
$$\mathbf{AB} = \begin{pmatrix} a & b & c \\ x & y & z \end{pmatrix} \begin{pmatrix} \alpha & \rho \\ \beta & \sigma \\ \gamma & \tau \end{pmatrix} = \begin{pmatrix} a\alpha + b\beta + c\gamma & a\rho + b\sigma + c\tau \\ x\alpha + y\beta + z\gamma & x\rho + y\sigma + z\tau \end{pmatrix},$$

and

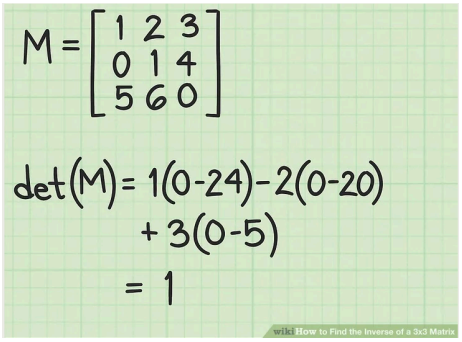
$$\mathbf{BA} = \begin{pmatrix} \alpha & \rho \\ \beta & \sigma \\ \gamma & \tau \end{pmatrix} \begin{pmatrix} a & b & c \\ x & y & z \end{pmatrix} = \begin{pmatrix} \alpha a + \rho x & \alpha b + \rho y & \alpha c + \rho z \\ \beta a + \sigma x & \beta b + \sigma y & \beta c + \sigma z \\ \gamma a + \tau x & \gamma b + \tau y & \gamma c + \tau z \end{pmatrix}.$$

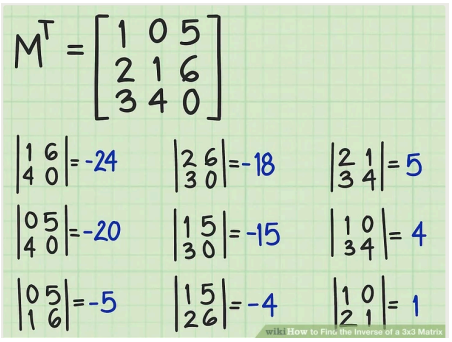
e. Write a Method to find the transpose of a matrix





f. Write a Method to find the determinant of a 2x2 matrix
g. Write a Method to find the determinant of a 3x3 matrix





h. Write a Method to find the inverse of a 2x2 matrix
i. Write a Method to find the inverse of a 3x3 matrix
j. Write a Method to display a matrix

Code

```
import java.util.Random;


public class MatrixOperations {


    // Generate random matrix
    public static double[][] generateMatrix(int rows, int cols) {
        double[][] matrix = new double[rows][cols];
        Random rand = new Random();
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                matrix[i][j] = rand.nextInt(10); // random values 0-9
        return matrix;
    }


    // Display matrix
    public static void displayMatrix(double[][] matrix) {
        for (double[] row : matrix) {
            for (double val : row)
                System.out.printf("%6.2f ", val);
            System.out.println();
        }
    }


    // Matrix Addition
```

```
public static double[][] add(double[][] A, double[][] B) {
    int rows = A.length, 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;
}

// Matrix Subtraction
public static double[][] subtract(double[][] A, double[][] B) {
    int rows = A.length, 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;
}

// Matrix Multiplication
public static double[][] multiply(double[][] A, double[][] B) {
    int r1 = A.length, c1 = A[0].length, c2 = B[0].length;
    double[][] result = new double[r1][c2];
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c2; j++)
            for (int k = 0; k < c1; k++)
                result[i][j] += A[i][k] * B[k][j];
    return result;
}

// Transpose
public static double[][] transpose(double[][] A) {
    int rows = A.length, cols = A[0].length;
    double[][] transposed = new double[cols][rows];
```

```
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            transposed[j][i] = A[i][j];
    return transposed;
}

// Determinant of 2x2
public static double determinant2x2(double[][] m) {
    return m[0][0] * m[1][1] - m[0][1] * m[1][0];
}

// Determinant of 3x3
public static double determinant3x3(double[][] 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]);
}

// Inverse of 2x2
public static double[][] inverse2x2(double[][] m) {
    double det = determinant2x2(m);
    if (det == 0) return null;
    double[][] inv = {
        { m[1][1] / det, -m[0][1] / det },
        { -m[1][0] / det, m[0][0] / det }
    };
    return inv;
}

// Inverse of 3x3 (using adjoint & determinant)
public static double[][] inverse3x3(double[][] m) {
    double 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]) / det;
        inv[0][1] = -(m[0][1]*m[2][2] - m[0][2]*m[2][1]) / det;
        inv[0][2] =  (m[0][1]*m[1][2] - m[0][2]*m[1][1]) / det;

        inv[1][0] = -(m[1][0]*m[2][2] - m[1][2]*m[2][0]) / det;
        inv[1][1] =  (m[0][0]*m[2][2] - m[0][2]*m[2][0]) / det;
        inv[1][2] = -(m[0][0]*m[1][2] - m[0][2]*m[1][0]) / det;

        inv[2][0] =  (m[1][0]*m[2][1] - m[1][1]*m[2][0]) / det;
        inv[2][1] = -(m[0][0]*m[2][1] - m[0][1]*m[2][0]) / det;
        inv[2][2] =  (m[0][0]*m[1][1] - m[0][1]*m[1][0]) / det;

        return inv;
    }

    // Main method to demo all
    public static void main(String[] args) {
        double[][] A = generateMatrix(3, 3);
        double[][] B = generateMatrix(3, 3);

        System.out.println("Matrix A:");
        displayMatrix(A);
        System.out.println("\nMatrix B:");
        displayMatrix(B);

        System.out.println("\nAddition (A + B):");
        displayMatrix(add(A, B));

        System.out.println("\nSubtraction (A - B):");
        displayMatrix(subtract(A, B));

        System.out.println("\nMultiplication (A x B):");
        displayMatrix(multiply(A, B));
```

```
        System.out.println("\nTranspose of Matrix A:");
        displayMatrix(transpose(A));

        System.out.println("\nDeterminant of Matrix A (3x3): " + determinant3x3(A));

        System.out.println("\nInverse of Matrix A (if exists):");
        double[][] inv = inverse3x3(A);
        if (inv != null)
            displayMatrix(inv);
        else
            System.out.println("Matrix A is non-invertible.");
    }
}
```