CodInClub

Powered By BridgeLabz

Best Programming Practice

1. All values as variables including Fixed, User Inputs, and Results
2. Proper naming conventions for all variables
3. Proper Program Name and Class Name
4. Proper Method Name which indicates action taking inputs and providing result

Sample Program 1: Create a program to find the sum of all the digits of a number given by a user using an array and display the sum.

a. Use Math.random() and get a 4-digit random integer number
b. Write a method to count digits in the number
c. Write a method to return an array of digits from a given number.
d. Write a method to Find the sum of the digits of the number in the array
e. Finally, display the sum of the digits of the number

Java

```java
// Create SumOfDigit Class to compute the sum of 4 digits random number
class SumOfDigits {
   // Get a 4 digit random number
   public int get4DigitRandomNumber() {
      return (int) (Math.random() * 9000) + 1000;
   }

   // Find the count of digits in the number
   public int countDigits(int number) {
      int count = 0, temp = number;
      while (temp > 0) {
         count++;
         temp /= 10;
      }
      return count;
   }

   // Store the digits of the number in an array
   public int[] getDigits(int number, int count) {
      int[] digits = new int[count];
      int temp = number;
      for (int i = count - 1; i >= 0; i--) {
         digits[i] = temp % 10;
         temp /= 10;
      }
      return digits;
   }
```

```
    // Find the sum of the elements in an array
    public int sumArray(int[] array) {
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
            sum += array[i];
        }
        return sum;
    }

    public static void main(String[] args) {
        // Get 4 digit random integer number
        SumOfDigits sumOfDigits = new SumOfDigits();
        int number = sumOfDigits.get4DigitRandomNumber();
        System.out.println("The Random Mumber is: " + number);

        // Get the count of digits
        int count = sumOfDigits.countDigits(number);
        System.out.println("The count of digit is: " + count);

        // Get the array of digits from the number
        int[] digits = sumOfDigits.getDigits(number, count);

        // Find the sum of the digits of the number
        int sum = sumOfDigits.sumArray(digits);

        // Display the sum of the digits of the number
        System.out.println("\nSum of Digits: " + sum);
    }
}
```

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

```java
package Day4.LabPractice_L3;
import java.util.Scanner;
import java.util.Random;
public class LP1 {
   public static int[] generateRandomHeights() {
       Random rand = new Random();
       int[] heights = new int[11];
       for (int i = 0; i < heights.length; i++) {
           heights[i] = rand.nextInt(101) + 150;
       }
       return heights;
   }
   public static int findSum(int[] heights) {
       int sum = 0;
       for (int height : heights) {
           sum += height;
       }
       return sum;
   }
   public static double findMeanHeight(int[] heights) {
       return (double) findSum(heights) / heights.length;
   }
```

```
    public static int findShortestHeight(int[] heights) {
        int min = heights[0];
        for (int height : heights) {
            min = Math.min(min, height);
        }
        return min;
    }
    public static int findTallestHeight(int[] heights) {
        int max = heights[0];
        for (int height : heights) {
            max = Math.max(max, height);
        }
        return max;
    }
    public static void main(String[] args) {
        int[] heights = generateRandomHeights();

        System.out.print("Player Heights: ");
        for (int height : heights) {
            System.out.print(height + " ");
        }
        System.out.println();
        System.out.println("Shortest Player Height: " +
findShortestHeight(heights) + " cm");
        System.out.println("Tallest Player Height: " +
findTallestHeight(heights) + " cm");
        System.out.println("Mean Height: " + findMeanHeight(heights) + "
cm");
    }
}
```

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.

```java
package Day4.LabPractice_L3;

import java.util.Scanner;

public class LP2 {

    public static int countDigits(int number) {

        return String.valueOf(Math.abs(number)).length();

    }

    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;

    }

    public static boolean isDuckNumber(int number) {

        String numStr = String.valueOf(Math.abs(number));

        return numStr.contains("0");

    }

    public static boolean isArmstrongNumber(int number) {

        int[] digits = getDigitsArray(number);

        int numDigits = digits.length;

        int sum = 0, original = number;


        for (int digit : digits) {
```

```
            sum += Math.pow(digit, numDigits);
        }
        return sum == original;
    }
    public static int[] findLargestAndSecondLargest(int[] digits) {
        int max1 = Integer.MIN_VALUE, max2 = Integer.MIN_VALUE;
        for (int digit : digits) {
            if (digit > max1) {
                max2 = max1;
                max1 = digit;
            } else if (digit > max2 && digit != max1) {
                max2 = digit;
            }
        }
        return new int[]{max1, max2};
    }
    public static int[] findSmallestAndSecondSmallest(int[] digits) {
        int min1 = Integer.MAX_VALUE, min2 = Integer.MAX_VALUE;
        for (int digit : digits) {
            if (digit < min1) {
                min2 = min1;
                min1 = digit;
            } else if (digit < min2 && digit != min1) {
                min2 = digit;
            }
        }
        return new int[]{min1, min2};
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = input.nextInt();
        int count = countDigits(number);
```

```
        int[] digitsArray = getDigitsArray(number);
        boolean isDuck = isDuckNumber(number);
        boolean isArmstrong = isArmstrongNumber(number);
        int[] largest = findLargestAndSecondLargest(digitsArray);
        int[] smallest = findSmallestAndSecondSmallest(digitsArray);
        System.out.println("Number of digits: " + count);
        System.out.print("Digits in number: ");
        for (int digit : digitsArray) {
            System.out.print(digit + " ");
        }
        System.out.println("\nIs Duck Number: " + isDuck);
        System.out.println("Is Armstrong Number: " + isArmstrong);
        System.out.println("Largest digit: " + largest[0] + ", Second
Largest digit: " + largest[1]);
        System.out.println("Smallest digit: " + smallest[0] + ", Second
Smallest digit: " + smallest[1]);
        input.close();
    }
}
```

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.

```
package Day4.LabPractice_L3;
import java.util.Scanner;
public class LP3 {
```

```java
public static int countDigits(int number) {
    return String.valueOf(Math.abs(number)).length();
}
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;
}
public static int sumOfDigits(int[] digits) {
    int sum = 0;
    for (int digit : digits) {
        sum += digit;
    }
    return sum;
}
public static int sumOfSquares(int[] digits) {
    int sum = 0;
    for (int digit : digits) {
        sum += Math.pow(digit, 2);
    }
    return sum;
}
public static boolean isHarshad(int number) {
    int[] digits = getDigitsArray(number);
    int sum = sumOfDigits(digits);
    return number % sum == 0;
}
public static int[][] digitFrequency(int number) {
    int[] digits = getDigitsArray(number);
    int[][] frequency = new int[10][2];
```

```
        for (int i = 0; i < 10; i++) {
            frequency[i][0] = i;
            frequency[i][1] = 0;
        }
        for (int digit : digits) {
            frequency[digit][1]++;
        }

        return frequency;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = input.nextInt();
        int[] digits = getDigitsArray(number);
        System.out.println("Number of digits: " + countDigits(number));
        System.out.println("Sum of digits: " + sumOfDigits(digits));
        System.out.println("Sum of squares of digits: " +
sumOfSquares(digits));
        System.out.println("Is " + number + " a Harshad number? " +
isHarshad(number));
        System.out.println("Digit Frequency:");
        int[][] frequency = digitFrequency(number);
        for (int i = 0; i < 10; i++) {
            if (frequency[i][1] > 0) {
                System.out.println("Digit " + frequency[i][0] + " appears
" + frequency[i][1] + " times.");
            }
        }
        input.close();
    }
}
```

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

```java
package Day4.LabPractice_L3;
import java.util.Arrays;
import java.util.Scanner;
public class LP4 {
    public static int countDigits(int number) {
        return String.valueOf(Math.abs(number)).length();
    }
    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;
    }
    public static int[] reverseDigitsArray(int[] digits) {
        int[] reversed = new int[digits.length];
        for (int i = 0; i < digits.length; i++) {
            reversed[i] = digits[digits.length - 1 - i];
        }
        return reversed;
    }
    public static boolean areArraysEqual(int[] arr1, int[] arr2) {
        return Arrays.equals(arr1, arr2);
```

```
    }
    public static boolean isPalindrome(int number) {
        int[] digits = getDigitsArray(number);
        int[] reversedDigits = reverseDigitsArray(digits);
        return areArraysEqual(digits, reversedDigits);
    }
    public static boolean isDuckNumber(int number) {
        String numStr = String.valueOf(Math.abs(number));
        return numStr.contains("0") && numStr.charAt(0) != '0';
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = input.nextInt();
        int[] digits = getDigitsArray(number);
        int[] reversedDigits = reverseDigitsArray(digits);
        System.out.println("Number of digits: " + countDigits(number));
        System.out.println("Original Digits Array: " +
Arrays.toString(digits));
        System.out.println("Reversed Digits Array: " +
Arrays.toString(reversedDigits));
        System.out.println("Is the number a palindrome? " +
isPalindrome(number));
        System.out.println("Is the number a Duck Number? " +
isDuckNumber(number));
        input.close();
    }
}
```

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

```
package Day4.LabPractice_L3;
import java.util.Scanner;
public class LP5 {
    public static boolean isPrime(int number) {
        if (number < 2) return false;
        for (int i = 2; i * i <= 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, 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 isBuzzNumber(int number) {
        return number % 7 == 0 || number % 10 == 7;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = input.nextInt();

        System.out.println("Is Prime? " + isPrime(number));
        System.out.println("Is Neon Number? " + isNeonNumber(number));
        System.out.println("Is Spy Number? " + isSpyNumber(number));
        System.out.println("Is Automorphic Number? " +
isAutomorphic(number));
        System.out.println("Is Buzz Number? " + isBuzzNumber(number));

        input.close();
    }
}
```

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

```
package Day4.LabPractice_L3;

import java.util.Scanner;

public class LP6 {

    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 findGreatestFactor(int[] factors) {

        return factors[factors.length - 2];

    }

    public static int sumOfFactors(int[] factors) {

        int sum = 0;

        for (int factor : factors) {
```

```
            sum += factor;
        }
        return sum;
    }
    public static long productOfFactors(int[] factors) {
        long product = 1;
        for (int factor : factors) {
            product *= factor;
        }
        return product;
    }
    public static long productOfCubeOfFactors(int[] factors) {
        long 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, temp = number;

        while (temp > 0) {
            int digit = temp % 10;
            sum += factorial(digit);
            temp /= 10;
        }

        return sum == number;
    }
    private static int factorial(int num) {
        int fact = 1;
        for (int i = 1; i <= num; i++) {
            fact *= i;
        }
        return fact;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
```

```
        int number = input.nextInt();
        int[] factors = findFactors(number);
        System.out.print("Factors: ");
        for (int factor : factors) {
            System.out.print(factor + " ");
        }
        System.out.println();
        System.out.println("Greatest Factor: " +
findGreatestFactor(factors));
        System.out.println("Sum of Factors: " + sumOfFactors(factors));
        System.out.println("Product of Factors: " +
productOfFactors(factors));
        System.out.println("Product of Cube of Factors: " +
productOfCubeOfFactors(factors));
        System.out.println("Is Perfect Number? " +
isPerfectNumber(number));
        System.out.println("Is Abundant Number? " +
isAbundantNumber(number));
        System.out.println("Is Deficient Number? " +
isDeficientNumber(number));
        System.out.println("Is Strong Number? " + isStrongNumber(number));
        input.close();
    }
}
```

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

```
package Day4.LabPractice_L3;
import java.util.HashSet;
import java.util.Random;
```

```java
public class LP7 {
    public static int generateOTP() {
        return 100000 + (int) (Math.random() * 900000); // Ensures a 6-digit number
    }
    public static boolean areOTPsUnique(int[] otps) {
        HashSet<Integer> otpSet = new HashSet<>();
        for (int otp : otps) {
            otpSet.add(otp);
        }
        return otpSet.size() == otps.length; // If set size is equal to array length, all are unique
    }
    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:");
        for (int otp : otps) {
            System.out.println(otp);
        }
        if (areOTPsUnique(otps)) {
            System.out.println("All OTPs are unique.");
        } else {
            System.out.println("Duplicate OTPs found.");
        }
    }
}
```

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

$y0 = y - (14 - m) / 12$

$x = y0 + y0/4 - y0/100 + y0/400$

$m0 = m + 12 \times ((14 - m) / 12) - 2$

$d0 = (d + x + 31m0 / 12) \bmod 7$

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

```
package Day4.LabPractice_L3;
import java.util.Scanner;
public class LP8 {
    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 int getDaysInMonth(int month, int year) {
        int[] days = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        if (month == 2 && isLeapYear(year)) {
            return 29;
        }
        return days[month - 1];
    }
```

```
    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
    public static int getFirstDayOfMonth(int month, int year) {
        int d = 1;
        int y = year - (14 - month) / 12;
        int x = y + y / 4 - y / 100 + y / 400;
        int m = month + 12 * ((14 - month) / 12) - 2;
        return (d + x + (31 * m) / 12) % 7;
    }
    public static void displayCalendar(int month, int year) {
        String monthName = getMonthName(month);
        int daysInMonth = getDaysInMonth(month, year);
        int firstDay = getFirstDayOfMonth(month, year);

        System.out.println("\n" + 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 <= daysInMonth; day++) {
            System.out.printf("%3d ", day);
            if ((firstDay + day) % 7 == 0) {
                System.out.println();
            }
        }
        System.out.println();
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter month (1-12): ");
        int month = input.nextInt();
        System.out.print("Enter year: ");
        int year = input.nextInt();

        if (month < 1 || month > 12 || year < 1) {
            System.out.println("Invalid input! Please enter a valid month (1-12)
and year.");
        } else {
            displayCalendar(month, year);
        }

        input.close();
    }
}
```

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

```
package Day4.LabPractice_L3;
import java.util.Scanner;
public class LP9 {
   public static double calculateDistance(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 slope = (y2 - y1) / (x2 - x1);
       double yIntercept = y1 - (slope * x1);
       return new double[]{slope, yIntercept};
   }
   public static void main(String[] args) {
       Scanner input = new Scanner(System.in);
       System.out.print("Enter x1: ");
       double x1 = input.nextDouble();
       System.out.print("Enter y1: ");
       double y1 = input.nextDouble();
       System.out.print("Enter x2: ");
       double x2 = input.nextDouble();
       System.out.print("Enter y2: ");
       double y2 = input.nextDouble();
       double distance = calculateDistance(x1, y1, x2, y2);
```

```
        System.out.println("Euclidean Distance between points: " +
distance);
        if (x1 == x2) {
            System.out.println("Vertical Line: x = " + x1);
        } else {
            double[] equation = findLineEquation(x1, y1, x2, y2);
            System.out.println("Equation of the line: y = " + equation[0]
+ "x + " + equation[1]);
        }
        input.close();
    }
}
```

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

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

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

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

```
package Day4.LabPractice_L3;
import java.util.Scanner;
public class LP10 {
```

```
   public static boolean isCollinearUsingSlope(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);
   }
   public static boolean isCollinearUsingArea(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 input = new Scanner(System.in);
       System.out.print("Enter x1, y1: ");
       double x1 = input.nextDouble();
       double y1 = input.nextDouble();
       System.out.print("Enter x2, y2: ");
       double x2 = input.nextDouble();
       double y2 = input.nextDouble();
       System.out.print("Enter x3, y3: ");
       double x3 = input.nextDouble();
       double y3 = input.nextDouble();
       boolean collinearSlope = isCollinearUsingSlope(x1, y1, x2, y2, x3,
y3);
       boolean collinearArea = isCollinearUsingArea(x1, y1, x2, y2, x3,
y3);
       if (collinearSlope && collinearArea) {
           System.out.println("The points are collinear.");
       } else {
           System.out.println("The points are not collinear.");
       }

       input.close();
   }
}
```

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

```
package Day4.LabPractice_L3;

import java.util.Random;

public class LP11 {

   public static int[][] generateEmployeeData(int numEmployees) {

       Random random = new Random();

       int[][] employeeData = new int[numEmployees][2];


       for (int i = 0; i < numEmployees; i++) {

           employeeData[i][0] = 10000 + random.nextInt(90000);

           employeeData[i][1] = random.nextInt(11);

       }

       return employeeData;

   }

   public static double[][] calculateBonusAndNewSalary(int[][]
employeeData) {

       double[][] updatedData = new double[employeeData.length][2];


       for (int i = 0; i < employeeData.length; i++) {

           int salary = employeeData[i][0];

           int yearsOfService = employeeData[i][1];


           double bonusPercentage = (yearsOfService > 5) ? 0.05 : 0.02;
// 5% if >5 years, else 2%

           double bonus = salary * bonusPercentage;

           double newSalary = salary + bonus;


           updatedData[i][0] = newSalary;
```

```
            updatedData[i][1] = bonus;
        }
        return updatedData;
    }
    public static void displaySummary(int[][] employeeData, double[][]
updatedData) {
        double totalOldSalary = 0, totalNewSalary = 0, totalBonus = 0;


System.out.println("-----------------------------------------------------
-----------------------------");
        System.out.printf("%-5s %-10s %-15s %-10s %-15s %-10s%n",
                "ID", "Salary", "Years of Service", "Bonus", "New Salary",
"Total Bonus");

System.out.println("-----------------------------------------------------
-----------------------------");


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


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


            System.out.printf("%-5d %-10d %-15d %-10.2f %-15.2f
%-10.2f%n",
                    (i + 1), oldSalary, yearsOfService, bonus, newSalary,
totalBonus);
        }
```

```
System.out.println("-----------------------------------------------------
----------------------------");
        System.out.printf("%-17s %-10.2f %-10s %-10.2f %-15s %-10.2f%n",
                "Total Old Salary:", totalOldSalary, "Total Bonus:",
totalBonus, "Total New Salary:", totalNewSalary);

System.out.println("-----------------------------------------------------
----------------------------");
    }
    public static void main(String[] args) {
        int numEmployees = 10;
        int[][] employeeData = generateEmployeeData(numEmployees);
        double[][] updatedData = calculateBonusAndNewSalary(employeeData);
        displaySummary(employeeData, updatedData);
    }
}
```

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"***.

```
package Day4.LabPractice_L3;
import java.util.Random;
import java.util.Scanner;
public class LP12 {
   public static int[][] generateStudentScores(int numStudents) {
       Random random = new Random();
       int[][] scores = new int[numStudents][3];

       for (int i = 0; i < numStudents; i++) {
           scores[i][0] = 50 + random.nextInt(51);
           scores[i][1] = 50 + random.nextInt(51);
           scores[i][2] = 50 + random.nextInt(51);
       }
       return scores;
   }
   public static double[][] computeResults(int[][] scores) {
       double[][] results = new double[scores.length][3];

       for (int i = 0; i < scores.length; i++) {
           int total = scores[i][0] + scores[i][1] + scores[i][2];
           double average = (double) total / 3;
           double percentage = (double) total / 3;

           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.printf("%-10s %-10s %-10s %-10s %-10s %-10s %-10s%n",
               "Student", "Physics", "Chemistry", "Maths", "Total",
"Average", "Percentage");

System.out.println("-------------------------------------------------------
----------------------------");


       for (int i = 0; i < scores.length; i++) {
           System.out.printf("%-10d %-10d %-10d %-10d %-10.2f %-10.2f
%-10.2f%n",
                   (i + 1), scores[i][0], scores[i][1], scores[i][2],
                   results[i][0], results[i][1], results[i][2]);
       }

System.out.println("-------------------------------------------------------
----------------------------");
   }
   public static void main(String[] args) {
       Scanner input = new Scanner(System.in);
       System.out.print("Enter the number of students: ");
       int numStudents = input.nextInt();
       int[][] scores = generateStudentScores(numStudents);
       double[][] results = computeResults(scores);
       displayScorecard(scores, results);
       input.close();
   }
}
```

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

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

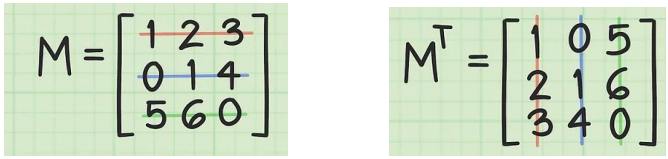
their matrix products are:

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

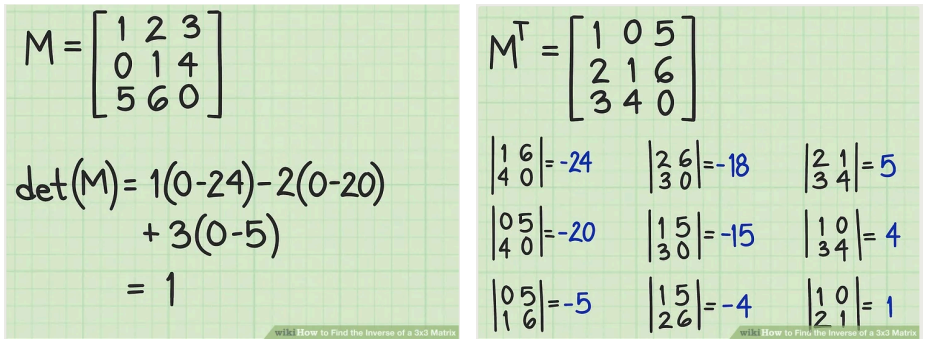
and

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

    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

```java
package Day4.LabPractice_L3;
import java.util.Random;
public class LP13 {
    public static double[][] generateMatrix(int rows, int cols) {
        Random random = new Random();
        double[][] matrix = new double[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = random.nextInt(10) + 1;
            }
        }
        return matrix;
    }
    public static void displayMatrix(double[][] matrix) {
        for (double[] row : matrix) {
            for (double val : row) {
                System.out.printf("%6.2f ", val);
            }
            System.out.println();
        }
        System.out.println();
    }
    public static double[][] addMatrices(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;
    }
```

```
    public static double[][] subtractMatrices(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;
    }
    public static double[][] multiplyMatrices(double[][] A, double[][] B)
{
        int rows = A.length, cols = B[0].length, common = B.length;
        double[][] result = new double[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                for (int k = 0; k < common; k++) {
                    result[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return result;
    }
    public static double[][] transposeMatrix(double[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        double[][] transpose = new double[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }
        return transpose;
```

```
    }
    public static double determinant2x2(double[][] matrix) {
        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
    }
    public static double determinant3x3(double[][] matrix) {
        return 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]);
    }
    public static double[][] inverse2x2(double[][] matrix) {
        double det = determinant2x2(matrix);
        if (det == 0) {
            throw new ArithmeticException("Matrix is singular, cannot be
inverted.");
        }
        double[][] inverse = {
                { matrix[1][1] / det, -matrix[0][1] / det },
                { -matrix[1][0] / det, matrix[0][0] / det }
        };
        return inverse;
    }
    public static double[][] inverse3x3(double[][] matrix) {
        double det = determinant3x3(matrix);
        if (det == 0) {
            throw new ArithmeticException("Matrix is singular, cannot be
inverted.");
        }
        double[][] adj = new double[3][3];

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

```
        adj[0][1] = -(matrix[1][0] * matrix[2][2] - matrix[1][2] *
matrix[2][0]);
        adj[0][2] = matrix[1][0] * matrix[2][1] - matrix[1][1] *
matrix[2][0];

        adj[1][0] = -(matrix[0][1] * matrix[2][2] - matrix[0][2] *
matrix[2][1]);
        adj[1][1] = matrix[0][0] * matrix[2][2] - matrix[0][2] *
matrix[2][0];
        adj[1][2] = -(matrix[0][0] * matrix[2][1] - matrix[0][1] *
matrix[2][0]);

        adj[2][0] = matrix[0][1] * matrix[1][2] - matrix[0][2] *
matrix[1][1];
        adj[2][1] = -(matrix[0][0] * matrix[1][2] - matrix[0][2] *
matrix[1][0]);
        adj[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] = adj[i][j] / det;
            }
        }
        return inverse;
    }
    public static void main(String[] args) {
        double[][] matrixA = generateMatrix(3, 3);
        double[][] matrixB = generateMatrix(3, 3);
        System.out.println("Matrix A:");
        displayMatrix(matrixA);
        System.out.println("Matrix B:");
        displayMatrix(matrixB);
        System.out.println("Matrix Addition:");
```

```
        displayMatrix(addMatrices(matrixA, matrixB));
        System.out.println("Matrix Subtraction:");
        displayMatrix(subtractMatrices(matrixA, matrixB));
        System.out.println("Matrix Multiplication:");
        displayMatrix(multiplyMatrices(matrixA, matrixB));
        System.out.println("Transpose of Matrix A:");
        displayMatrix(transposeMatrix(matrixA));
        System.out.println("Determinant of Matrix A: " +
determinant3x3(matrixA));
        try {
            System.out.println("Inverse of Matrix A:");
            displayMatrix(inverse3x3(matrixA));
        } catch (ArithmeticException e) {
            System.out.println("Matrix A is singular, no inverse
exists.");
        }
    }
}
```