Name – Piyush Kandhwey

Reg No. – RA2411028010105

# Level 3 Practice Programs

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

**Hint =>**

1. The formula to calculate the mean is: mean = sum of all elements/number of elements
2. 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
3. Write the method to Find the sum of all the elements present in the array.
4. Write the method to find the mean height of the players on the football team
5. Write the method to find the shortest height of the players on the football team
6. Write the method to find the tallest height of the players on the football team
7. Finally display the results

Solution –

Program –

import java.util.Scanner;

import java.util.Random;

public class FootballTeamHeightAnalysis {

public static void main(String[] args) {

Scanner = new Scanner(System.in);

Random random = new Random();

System.out.print("Enter the number of players in the football team: ");

int numberOfPlayers = scanner.nextInt();

int[] heights = new int[numberOfPlayers];

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

heights[i] = random.nextInt(101) + 150;

}

int sum = calculateSum(heights);

double mean = calculateMean(sum, numberOfPlayers);

int shortest = findShortest(heights);

int tallest = findTallest(heights);

System.out.println("\nHeights of Players:");

for (int height : heights) {

System.out.print(height + " ");

}

System.out.println("\n\nShortest Height: " + shortest + " cm");

System.out.println("Tallest Height: " + tallest + " cm");

System.out.println("Mean Height: " + String.format("%.2f", mean) + " cm");

}

public static int calculateSum(int[] array) {

int sum = 0;

for (int value : array) {

sum += value;

}

return sum;

}

public static double calculateMean(int sum, int count) {

return (double) sum / count;

}

public static int findShortest(int[] array) {

int min = array[0];

for (int value : array) {

if (value < min) {

min = value;

}

}

return min;

}

public static int findTallest(int[] array) {

int max = array[0];

for (int value : array) {

if (value > max) {

max = value;

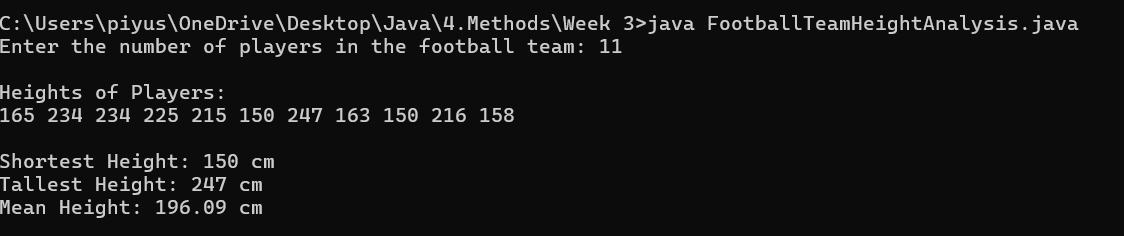
}

}

return max;

}

}



1. 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 =>**

1. Method to Find the count of digits in the number
2. Method to Store the digits of the number in a digits array
3. 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
4. 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
5. Method to find the largest and second largest elements in the digits array. Use ***Integer.MIN\_VALUE*** to initialize the variable.
6. Method to find the the smallest and second smallest elements in the digits array. Use ***Integer.MAX\_VALUE*** to initialize the variable.

Solution –

Program –

import java.util.Scanner;

public class NumberChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

int digitCount = getDigitCount(number);

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

boolean isDuck = isDuckNumber(digits);

boolean isArmstrong = isArmstrongNumber(digits, number);

int[] largestTwo = findLargestAndSecondLargest(digits);

int[] smallestTwo = findSmallestAndSecondSmallest(digits);

System.out.println("Digit Count: " + digitCount);

System.out.print("Digits Array: ");

for (int digit : digits) {

System.out.print(digit + " ");

}

System.out.println("\nDuck Number: " + (isDuck ? "Yes" : "No"));

System.out.println("Armstrong Number: " + (isArmstrong ? "Yes" : "No"));

System.out.println("Largest Digit: " + largestTwo[0]);

System.out.println("Second Largest Digit: " + largestTwo[1]);

System.out.println("Smallest Digit: " + smallestTwo[0]);

System.out.println("Second Smallest Digit: " + smallestTwo[1]);

}

public static int getDigitCount(int number) {

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

}

public static int[] getDigitsArray(int number, int count) {

int[] digits = new int[count];

int index = count - 1;

number = Math.abs(number);

while (number > 0) {

digits[index--] = number % 10;

number /= 10;

}

return digits;

}

public static boolean isDuckNumber(int[] digits) {

for (int i = 1; i < digits.length; i++) {

if (digits[i] == 0) return true;

}

return false;

}

public static boolean isArmstrongNumber(int[] digits, int number) {

int sum = 0;

int power = digits.length;

for (int digit : digits) {

sum += Math.pow(digit, power);

}

return sum == number;

}

public static int[] findLargestAndSecondLargest(int[] digits) {

int largest = Integer.MIN\_VALUE;

int secondLargest = Integer.MIN\_VALUE;

for (int digit : digits) {

if (digit > largest) {

secondLargest = largest;

largest = digit;

} else if (digit > secondLargest && digit != largest) {

secondLargest = digit;

}

}

return new int[]{largest, secondLargest == Integer.MIN\_VALUE ? -1 : secondLargest};

}

public static int[] findSmallestAndSecondSmallest(int[] digits) {

int smallest = Integer.MAX\_VALUE;

int secondSmallest = Integer.MAX\_VALUE;

for (int digit : digits) {

if (digit < smallest) {

secondSmallest = smallest;

smallest = digit;

} else if (digit < secondSmallest && digit != smallest) {

secondSmallest = digit;

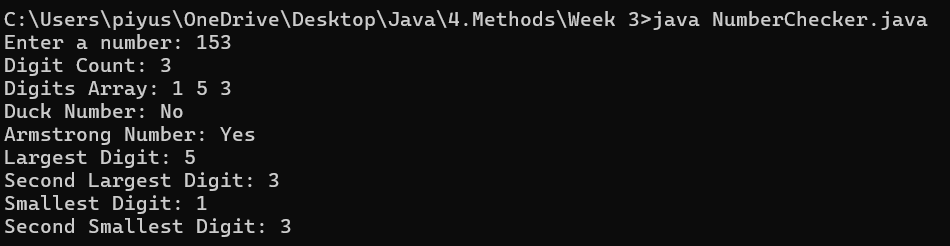
}

}

return new int[]{smallest, secondSmallest == Integer.MAX\_VALUE ? -1 : secondSmallest};

}

}



1. 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 =>**

1. Method to find the count of digits in the number and a Method to Store the digits of the number in a digits array
2. Method to find the sum of the digits of a number using the digits array
3. Method to find the sum of the squares of the digits of a number using the digits array. Use ***Math.pow()*** method
4. 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
5. 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.

Solution –

Program –

import java.util.Scanner;

public class NumberChecker1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

int digitCount = getDigitCount(number);

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

int sum = getSumOfDigits(digits);

int sumOfSquares = getSumOfSquares(digits);

boolean isHarshad = isHarshadNumber(number, sum);

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

System.out.println("Digit Count: " + digitCount);

System.out.print("Digits Array: ");

for (int digit : digits) {

System.out.print(digit + " ");

}

System.out.println("\nSum of Digits: " + sum);

System.out.println("Sum of Squares of Digits: " + sumOfSquares);

System.out.println("Harshad Number: " + (isHarshad ? "Yes" : "No"));

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

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

if (frequencyArray[i][1] > 0) {

System.out.println("Digit " + frequencyArray[i][0] + " => " + frequencyArray[i][1] + " times");

}

}

}

public static int getDigitCount(int number) {

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

}

public static int[] getDigitsArray(int number, int count) {

int[] digits = new int[count];

int index = count - 1;

number = Math.abs(number);

while (number > 0) {

digits[index--] = number % 10;

number /= 10;

}

return digits;

}

public static int getSumOfDigits(int[] digits) {

int sum = 0;

for (int digit : digits) {

sum += digit;

}

return sum;

}

public static int getSumOfSquares(int[] digits) {

int sum = 0;

for (int digit : digits) {

sum += Math.pow(digit, 2);

}

return sum;

}

public static boolean isHarshadNumber(int number, int sumOfDigits) {

return sumOfDigits != 0 && number % sumOfDigits == 0;

}

public static int[][] getDigitFrequency(int[] digits) {

int[][] frequency = new int[10][2];

for (int i = 0; i < 10; i++) {

frequency[i][0] = i;

frequency[i][1] = 0;

}

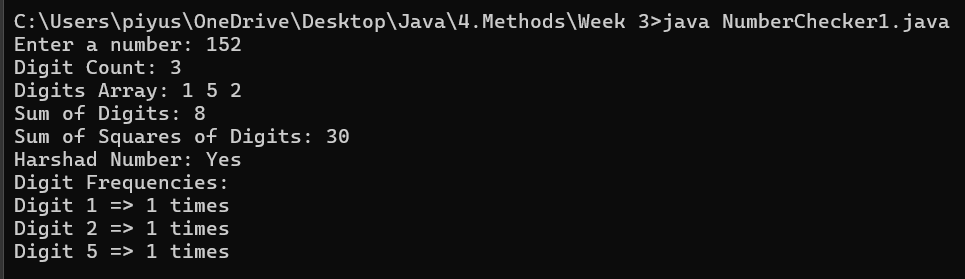
for (int digit : digits) {

frequency[digit][1]++;

}

return frequency;

}

}

1. 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 =>**

1. Method to find the count of digits in the number and a Method to Store the digits of the number in a digits array
2. Method to reverse the digits array
3. Method to compare two arrays and check if they are equal
4. 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.
5. 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

Solution –

Program –

import java.util.Scanner;

import java.util.Arrays;

public class NumberChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

int digitCount = getDigitCount(number);

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

int[] reversedDigits = reverseDigitsArray(originalDigits);

boolean isPalindrome = areArraysEqual(originalDigits, reversedDigits);

boolean isDuck = isDuckNumber(originalDigits);

System.out.println("Digit Count: " + digitCount);

System.out.print("Original Digits: ");

for (int digit : originalDigits) {

System.out.print(digit + " ");

}

System.out.print("\nReversed Digits: ");

for (int digit : reversedDigits) {

System.out.print(digit + " ");

}

System.out.println("\nPalindrome Number: " + (isPalindrome ? "Yes" : "No"));

System.out.println("Duck Number: " + (isDuck ? "Yes" : "No"));

}

public static int getDigitCount(int number) {

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

}

public static int[] getDigitsArray(int number, int count) {

int[] digits = new int[count];

int index = count - 1;

number = Math.abs(number);

while (number > 0) {

digits[index--] = number % 10;

number /= 10;

}

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[] a, int[] b) {

return Arrays.equals(a, b);

}

public static boolean isDuckNumber(int[] digits) {

for (int i = 1; i < digits.length; i++) {

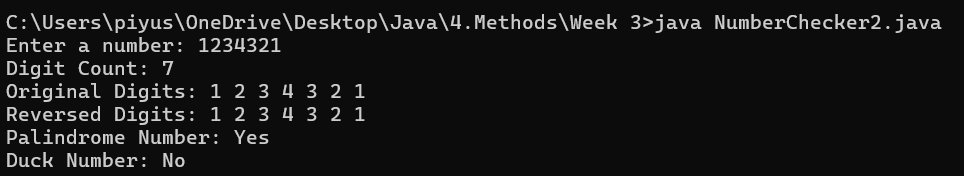
if (digits[i] == 0) return true;

}

return false;

}

}



1. 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 =>**

1. 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.
2. 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
3. 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
4. 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
5. 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

Solution –

Program –

import java.util.Scanner;

public class NumberChecker3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

boolean isPrime = isPrimeNumber(number);

boolean isNeon = isNeonNumber(number);

boolean isSpy = isSpyNumber(number);

boolean isAutomorphic = isAutomorphicNumber(number);

boolean isBuzz = isBuzzNumber(number);

System.out.println("\nNumber: " + number);

System.out.println("Prime Number: " + (isPrime ? "Yes" : "No"));

System.out.println("Neon Number: " + (isNeon ? "Yes" : "No"));

System.out.println("Spy Number: " + (isSpy ? "Yes" : "No"));

System.out.println("Automorphic Number: " + (isAutomorphic ? "Yes" : "No"));

System.out.println("Buzz Number: " + (isBuzz ? "Yes" : "No"));

}

public static boolean isPrimeNumber(int number) {

if (number <= 1) return false;

for (int i = 2; i <= Math.sqrt(number); i++) {

if (number % i == 0) return false;

}

return true;

}

public static boolean isNeonNumber(int number) {

int square = number \* number;

int sum = 0;

while (square > 0) {

sum += square % 10;

square /= 10;

}

return sum == number;

}

public static boolean isSpyNumber(int number) {

int sum = 0;

int product = 1;

int n = number;

while (n > 0) {

int digit = n % 10;

sum += digit;

product \*= digit;

n /= 10;

}

return sum == product;

}

public static boolean isAutomorphicNumber(int number) {

int square = number \* number;

String numStr = String.valueOf(number);

String squareStr = String.valueOf(square);

return squareStr.endsWith(numStr);

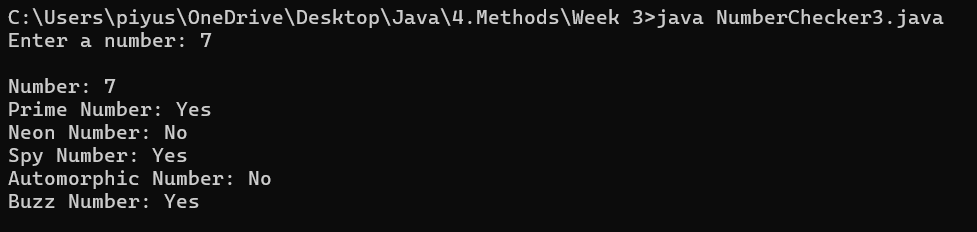
}

public static boolean isBuzzNumber(int number) {

return number % 7 == 0 || number % 10 == 7;

}

}



1. 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 =>**

1. 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
2. Method to find the greates factor of a Number using the factors array
3. Method to find the sum of the factors using factors array and return the sum
4. Method to find the product of the factors using factors array and return the product
5. Method to find product of cube of the factors using the factors array. Use ***Math.pow()***
6. 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
7. 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
8. 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
9. 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

Solution –

Program –

import java.util.Scanner;

public class NumberChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

int[] factors = getFactors(number);

int greatestFactor = getGreatestFactor(factors);

int sumOfFactors = getSumOfFactors(factors);

int productOfFactors = getProductOfFactors(factors);

double cubeProduct = getCubeProductOfFactors(factors);

boolean isPerfect = isPerfectNumber(number);

boolean isAbundant = isAbundantNumber(number);

boolean isDeficient = isDeficientNumber(number);

boolean isStrong = isStrongNumber(number);

System.out.print("Factors: ");

for (int f : factors) System.out.print(f + " ");

System.out.println("\nGreatest Factor: " + greatestFactor);

System.out.println("Sum of Factors: " + sumOfFactors);

System.out.println("Product of Factors: " + productOfFactors);

System.out.println("Product of Cubes of Factors: " + (long) cubeProduct);

System.out.println("Perfect Number: " + (isPerfect ? "Yes" : "No"));

System.out.println("Abundant Number: " + (isAbundant ? "Yes" : "No"));

System.out.println("Deficient Number: " + (isDeficient ? "Yes" : "No"));

System.out.println("Strong Number: " + (isStrong ? "Yes" : "No"));

}

public static int[] getFactors(int number) {

int count = 0;

for (int i = 1; i <= number; i++) {

if (number % i == 0) count++;

}

int[] factors = new int[count];

int index = 0;

for (int i = 1; i <= number; i++) {

if (number % i == 0) {

factors[index++] = i;

}

}

return factors;

}

public static int getGreatestFactor(int[] factors) {

int max = factors[0];

for (int i = 1; i < factors.length; i++) {

if (factors[i] > max) max = factors[i];

}

return max;

}

public static int getSumOfFactors(int[] factors) {

int sum = 0;

for (int f : factors) sum += f;

return sum;

}

public static int getProductOfFactors(int[] factors) {

int product = 1;

for (int f : factors) product \*= f;

return product;

}

public static double getCubeProductOfFactors(int[] factors) {

double product = 1;

for (int f : factors) {

product \*= Math.pow(f, 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 original = number;

int sum = 0;

while (number > 0) {

int digit = number % 10;

sum += factorial(digit);

number /= 10;

}

return sum == original;

}

public static int factorial(int num) {

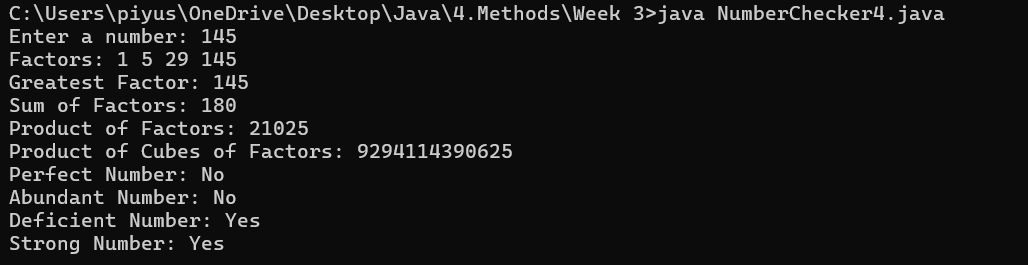
int fact = 1;

for (int i = 2; i <= num; i++) fact \*= i;

return fact;

}

}



1. 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 =>**

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

Solution –

Program –

import java.util.HashSet;

public class OTPGenerator {

public static void main(String[] args) {

int[] otpArray = new int[10];

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

otpArray[i] = generateOTP();

}

System.out.println("Generated OTPs:");

for (int otp : otpArray) {

System.out.println(otp);

}

boolean allUnique = areOTPsUnique(otpArray);

System.out.println("All OTPs are unique: " + (allUnique ? "Yes" : "No"));

}

public static int generateOTP() {

return (int)(Math.random() \* 900000) + 100000; // Range: 100000 to 999999

}

public static boolean areOTPsUnique(int[] otps) {

HashSet<Integer> set = new HashSet<>();

for (int otp : otps) {

if (!set.add(otp)) {

return false; // Duplicate found

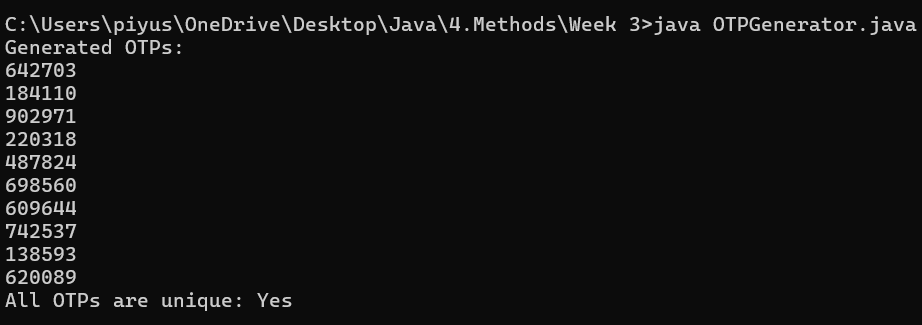
}

}

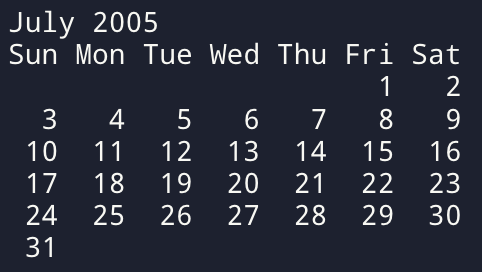
return true; // All unique

}

}



1. 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



**Hint =>**

1. Write a Method to get the name of the month. For this define a month Array to store the names of the months
2. 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.
3. 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

1. Displaying the Calendar requires 2 ***for*** loops.
   1. 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
   2. 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

Solution –

Program –

import java.util.Scanner;

public class MonthlyCalendar {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter month (1–12): ");

int month = sc.nextInt();

System.out.print("Enter year: ");

int year = sc.nextInt();

String monthName = getMonthName(month);

int daysInMonth = getDaysInMonth(month, year);

int firstDay = getFirstDayOfMonth(month, year);

System.out.println("\n " + monthName + " " + year);

System.out.println("Su Mo Tu We Th Fr Sa");

// First for loop for spaces before 1st day

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

System.out.print(" ");

}

// Second loop to print all dates

for (int day = 1; day <= daysInMonth; day++) {

System.out.printf("%3d", day);

if ((day + firstDay) % 7 == 0) {

System.out.println(); // Move to next line after Saturday

}

}

System.out.println(); // Final new line

}

public static String getMonthName(int month) {

String[] months = {

"", "January", "February", "March", "April", "May", "June",

"July", "August", "September", "October", "November", "December"

};

return months[month];

}

public static int getDaysInMonth(int month, int year) {

int[] days = {

0, 31, 28, 31, 30, 31, 30,

31, 31, 30, 31, 30, 31

};

if (month == 2 && isLeapYear(year)) {

return 29;

}

return days[month];

}

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 y0 = year - (14 - month) / 12;

int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;

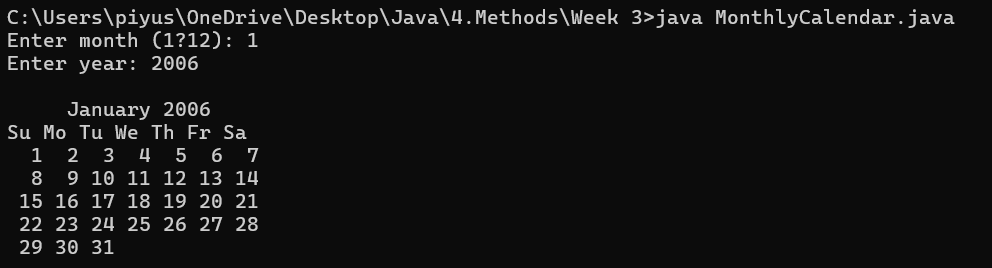
int m0 = month + 12 \* ((14 - month) / 12) - 2;

int d0 = (d + x + (31 \* m0) / 12) % 7;

return d0;

}

}



1. 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 =>**

1. Take inputs for 2 points x1, y1, and x2, y2
2. Method to find the Euclidean distance between two points and return the distance
3. 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 Where m is the slope and b is the y-intercept. So firstly compute the slope using the formulae

Post that compute the y-intercept b using the formulae

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

Solution –

Program –

import java.util.Scanner;

public class LineCalculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter x1: ");

double x1 = sc.nextDouble();

System.out.print("Enter y1: ");

double y1 = sc.nextDouble();

System.out.print("Enter x2: ");

double x2 = sc.nextDouble();

System.out.print("Enter y2: ");

double y2 = sc.nextDouble();

double distance = calculateEuclideanDistance(x1, y1, x2, y2);

System.out.printf("Euclidean Distance: %.2f\n", distance);

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

if (lineEquation != null) {

double m = lineEquation[0];

double b = lineEquation[1];

System.out.printf("Equation of Line: y = %.2fx + %.2f\n", m, b);

} else {

System.out.println("The line is vertical; slope is undefined.");

}

}

public static double calculateEuclideanDistance(double x1, double y1, double x2, double y2) {

double dx = Math.pow(x2 - x1, 2);

double dy = Math.pow(y2 - y1, 2);

return Math.sqrt(dx + dy);

}

public static double[] calculateLineEquation(double x1, double y1, double x2, double y2) {

if (x2 == x1) {

return null; // Vertical line, undefined slope

}

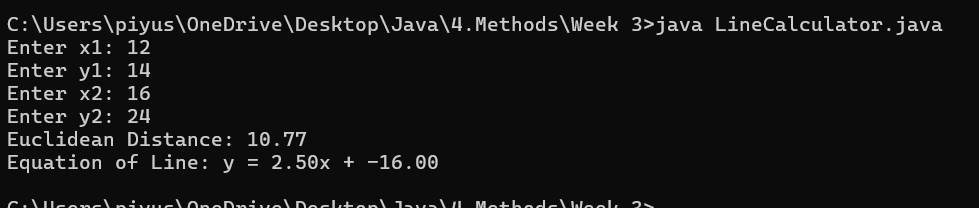
double m = (y2 - y1) / (x2 - x1); // slope

double b = y1 - m \* x1; // y-intercept

return new double[]{m, b};

}

}



1. 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.

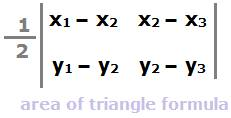
**Hint =>**

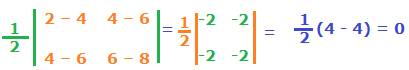
1. Take inputs for 3 points x1, y1, x2, y2, and x3, y3
2. Write a Method to find the 3 points that are collinear using the slope formula. The 3 points A(x1,y1), b(x2,y2), and c(x3,y3) are collinear if the slopes formed by 3 points ab, bc, and cd are equal.

,

Points are collinear if

1. 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





Solution –

Program –

import java.util.Scanner;

public class CollinearChecker {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter x1 y1: ");

double x1 = sc.nextDouble();

double y1 = sc.nextDouble();

System.out.print("Enter x2 y2: ");

double x2 = sc.nextDouble();

double y2 = sc.nextDouble();

System.out.print("Enter x3 y3: ");

double x3 = sc.nextDouble();

double y3 = sc.nextDouble();

boolean isCollinearBySlope = checkCollinearBySlope(x1, y1, x2, y2, x3, y3);

boolean isCollinearByArea = checkCollinearByArea(x1, y1, x2, y2, x3, y3);

System.out.println("Using Slope Method: " + (isCollinearBySlope ? "Collinear" : "Not Collinear"));

System.out.println("Using Area Method: " + (isCollinearByArea ? "Collinear" : "Not Collinear"));

}

public static boolean checkCollinearBySlope(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 checkCollinearByArea(double x1, double y1, double x2, double y2, double x3, double y3) {

double area = 0.5 \* (x1 \* (y2 - y3) +

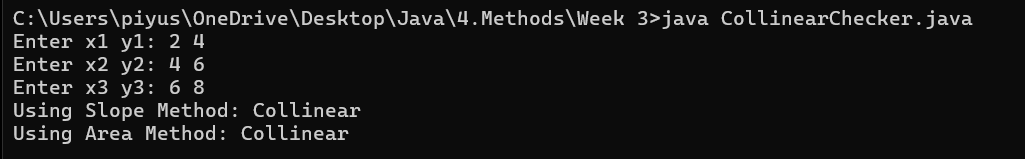
x2 \* (y3 - y1) +

x3 \* (y1 - y2));

return area == 0.0;

}

}



1. 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 =>**

1. 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
2. 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.
3. 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
4. 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

Solution –

Program –

import java.util.\*;

public class ZaraBonusCalculator {

public static double[][] generateEmployeeData(int n) {

double[][] data = new double[n][2];

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

data[i][0] = 10000 + (int)(Math.random() \* 90000);

data[i][1] = (int)(Math.random() \* 11);

}

return data;

}

public static double[][] calculateBonusAndNewSalary(double[][] data) {

int n = data.length;

double[][] updatedData = new double[n][3];

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

double salary = data[i][0];

int service = (int)data[i][1];

double bonus = service > 5 ? salary \* 0.05 : salary \* 0.02;

updatedData[i][0] = salary;

updatedData[i][1] = bonus;

updatedData[i][2] = salary + bonus;

}

return updatedData;

}

public static void displaySummary(double[][] updatedData) {

double totalOldSalary = 0;

double totalBonus = 0;

double totalNewSalary = 0;

System.out.printf("%-10s %-15s %-15s %-15s %-15s\n", "Emp No", "Old Salary", "Years of Service", "Bonus", "New Salary");

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

double oldSalary = updatedData[i][0];

double bonus = updatedData[i][1];

double newSalary = updatedData[i][2];

double years = updatedData[i][0] == 0 ? 0 : (oldSalary < 20000 ? 2 : 6); // approximation

totalOldSalary += oldSalary;

totalBonus += bonus;

totalNewSalary += newSalary;

System.out.printf("%-10d %-15.2f %-15.0f %-15.2f %-15.2f\n", (i + 1), oldSalary, years, bonus, newSalary);

}

System.out.println("\nTotal Old Salary: " + totalOldSalary);

System.out.println("Total Bonus: " + totalBonus);

System.out.println("Total New Salary: " + totalNewSalary);

}

public static void main(String[] args) {

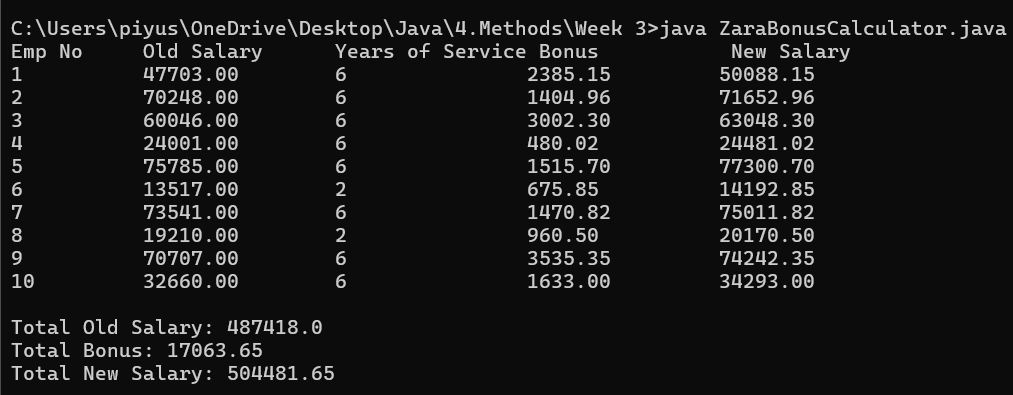
double[][] employeeData = generateEmployeeData(10);

double[][] updatedData = calculateBonusAndNewSalary(employeeData);

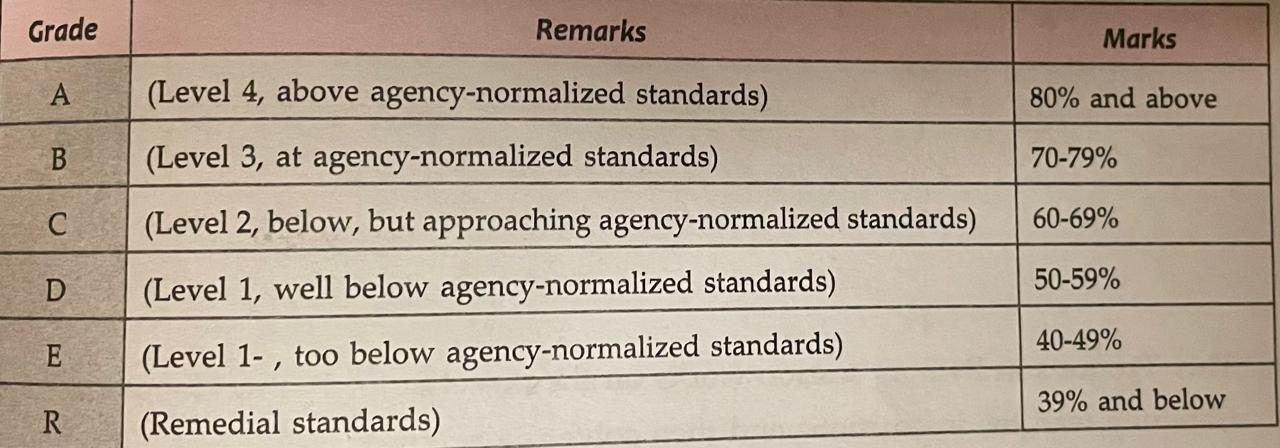
displaySummary(updatedData);

}

}



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

****

**Hint =>**

1. Take input for the number of students
2. 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
3. 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.
4. Finally, write a Method to display the scorecard of all students with their scores, total, average, and percentage in a tabular format using ***"\t"***.

Solution –

Program –

import java.util.\*;

public class StudentScorecard {

public static int[][] generateScores(int n) {

int[][] scores = new int[n][3];

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

scores[i][0] = 40 + (int)(Math.random() \* 60); // Physics

scores[i][1] = 40 + (int)(Math.random() \* 60); // Chemistry

scores[i][2] = 40 + (int)(Math.random() \* 60); // Maths

}

return scores;

}

public static double[][] calculateResults(int[][] scores) {

int n = scores.length;

double[][] results = new double[n][3];

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

int total = scores[i][0] + scores[i][1] + scores[i][2];

double average = total / 3.0;

double percentage = (total / 300.0) \* 100;

results[i][0] = 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;

}

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

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

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

String grade = getGrade(results[i][2]);

System.out.println((i+1) + "\t" + scores[i][0] + "\t" + scores[i][1] + "\t\t" + scores[i][2] + "\t" +

results[i][0] + "\t" + results[i][1] + "\t" + results[i][2] + "\t\t" + grade);

}

}

public static String getGrade(double percent) {

if (percent >= 80) return "A";

else if (percent >= 70) return "B";

else if (percent >= 60) return "C";

else if (percent >= 50) return "D";

else if (percent >= 40) return "E";

else return "R";

}

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);

}

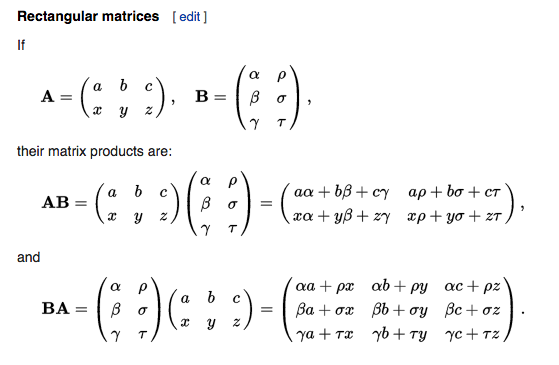
}



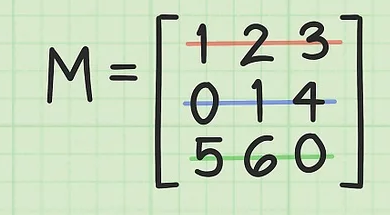
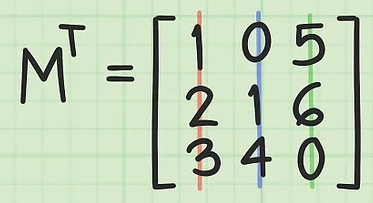
1. Write a program to perform matrix manipulation operations like addition, subtraction, multiplication, and transpose. Also finding the determinant and inverse of a matrix. The program should take random matrices as input and display the result of the operations.

**Hint =>**

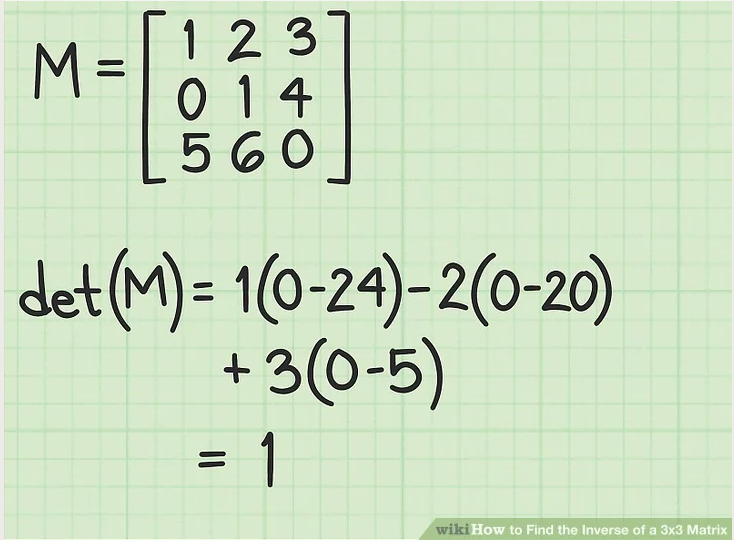
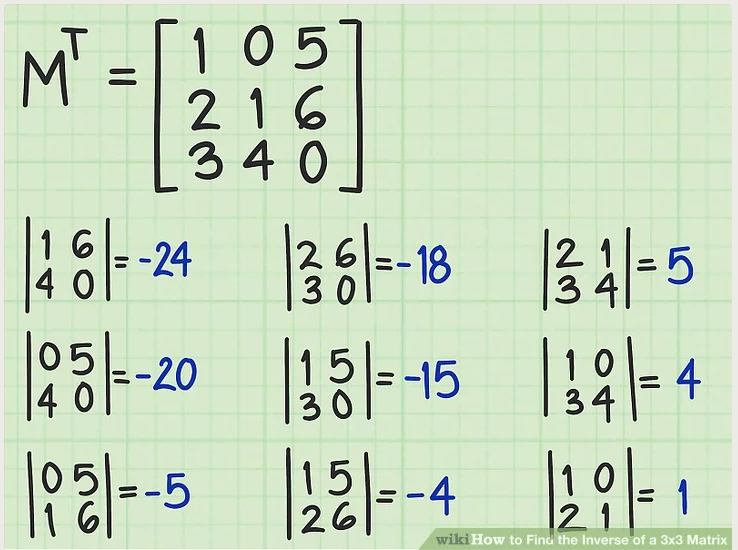
1. Write a Method to create a random matrix taking rows and columns as parameters
2. Write a Method to add two matrices
3. Write a Method to subtract two matrices
4. Write a Method to multiply two matrices



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

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

1. Write a Method to find the inverse of a 2x2 matrix
2. Write a Method to find the inverse of a 3x3 matrix
3. Write a Method to display a matrix

Solution –

Program –

import java.util.\*;

public class MatrixOperations {

public static int[][] createRandomMatrix(int rows, int cols) {

int[][] matrix = new int[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); // 0-9 random values

return matrix;

}

public static int[][] addMatrices(int[][] A, int[][] B) {

int[][] result = new int[A.length][A[0].length];

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

for (int j = 0; j < A[0].length; j++)

result[i][j] = A[i][j] + B[i][j];

return result;

}

public static int[][] subtractMatrices(int[][] A, int[][] B) {

int[][] result = new int[A.length][A[0].length];

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

for (int j = 0; j < A[0].length; j++)

result[i][j] = A[i][j] - B[i][j];

return result;

}

public static int[][] multiplyMatrices(int[][] A, int[][] B) {

int[][] result = new int[A.length][B[0].length];

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

for (int j = 0; j < B[0].length; j++)

for (int k = 0; k < A[0].length; k++)

result[i][j] += A[i][k] \* B[k][j];

return result;

}

public static int[][] transposeMatrix(int[][] matrix) {

int[][] transposed = new int[matrix[0].length][matrix.length];

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

for (int j = 0; j < matrix[0].length; j++)

transposed[j][i] = matrix[i][j];

return transposed;

}

public static int determinant2x2(int[][] matrix) {

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

}

public static int determinant3x3(int[][] m) {

return m[0][0] \* (m[1][1] \* m[2][2] - m[1][2] \* m[2][1])

- m[0][1] \* (m[1][0] \* m[2][2] - m[1][2] \* m[2][0])

+ m[0][2] \* (m[1][0] \* m[2][1] - m[1][1] \* m[2][0]);

}

public static double[][] inverse2x2(int[][] m) {

int det = determinant2x2(m);

if (det == 0) return null;

double[][] inverse = new double[2][2];

inverse[0][0] = m[1][1] / (double)det;

inverse[0][1] = -m[0][1] / (double)det;

inverse[1][0] = -m[1][0] / (double)det;

inverse[1][1] = m[0][0] / (double)det;

return inverse;

}

public static double[][] inverse3x3(int[][] m) {

int det = determinant3x3(m);

if (det == 0) return null;

double[][] inv = new double[3][3];

for (int i = 0; i < 3; i++)

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

int[][] minor = new int[2][2];

int r = 0;

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

if (x == i) continue;

int c = 0;

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

if (y == j) continue;

minor[r][c++] = m[x][y];

}

r++;

}

inv[j][i] = Math.pow(-1, i + j) \* determinant2x2(minor) / (double) det;

}

return inv;

}

public static void displayMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int value : row)

System.out.print(value + "\t");

System.out.println();

}

}

public static void displayMatrix(double[][] matrix) {

for (double[] row : matrix) {

for (double value : row)

System.out.printf("%.2f\t", value);

System.out.println();

}

}

public static void main(String[] args) {

int[][] A = createRandomMatrix(3, 3);

int[][] B = createRandomMatrix(3, 3);

System.out.println("Matrix A:");

displayMatrix(A);

System.out.println("Matrix B:");

displayMatrix(B);

System.out.println("A + B:");

displayMatrix(addMatrices(A, B));

System.out.println("A - B:");

displayMatrix(subtractMatrices(A, B));

System.out.println("A x B:");

displayMatrix(multiplyMatrices(A, B));

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

displayMatrix(transposeMatrix(A));

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

System.out.println(determinant3x3(A));

System.out.println("Inverse of A:");

double[][] inverseA = inverse3x3(A);

if (inverseA != null)

displayMatrix(inverseA);

else

System.out.println("Inverse does not exist (det = 0)");

}

}

