Name – Piyush Kandhwey

Reg No – RA2411028010105

# Level 2 Practice Programs

1. Create a program to find the factors of a number taken as user input, store the factors in an array, and display the factors. Also find the sum, sum of square of factors and product of the factors and display the results

**Hint =>**

1. Take the input for a number
2. Write a ***static*** Method to find the factors of the number and save them in an array and return the array.
3. To find factors and save to array will have two loops. The first loop to find the count and initialize the array with the count. And the second loop save the factors into the array
4. Write a method to find the sum of the factors using factors array
5. Write a method to find the product of the factors using factors array
6. Write a method to find the sum of square of the factors using ***Math.pow()*** method

import java.util.Scanner;

public class FactorCalculator {

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 findSum(int[] factors) {

int sum = 0;

for (int factor : factors) {

sum += factor;

}

return sum;

}

public static int findProduct(int[] factors) {

int product = 1;

for (int factor : factors) {

product \*= factor;

}

return product;

}

public static int findSumOfSquares(int[] factors) {

int sum = 0;

for (int factor : factors) {

sum += (int)Math.pow(factor, 2);

}

return sum;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int number = sc.nextInt();

int[] factors = findFactors(number);

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

for (int f : factors) {

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

}

System.out.println();

int sum = findSum(factors);

int product = findProduct(factors);

int sumOfSquares = findSumOfSquares(factors);

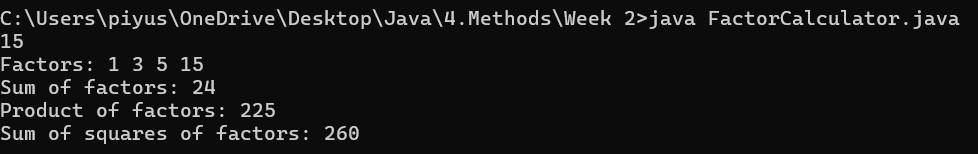
System.out.println("Sum of factors: " + sum);

System.out.println("Product of factors: " + product);

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

}

}



1. Write a program to find the sum of n natural numbers using recursive method and compare the result with the formulae n\*(n+1)/2 and show the result from both computations is correct.

**Hint =>**

1. Take the user input number and check whether it's a Natural number, if not exit
2. Write a Method to find the sum of n natural numbers using **recursion**
3. Write a Method to find the sum of n natural numbers using the formulae n\*(n+1)/2
4. Compare the two results and print the result

import java.util.Scanner;

public class NaturalNumberSumComparison {

public static int sumRecursive(int n) {

if (n == 1) {

return 1;

}

return n + sumRecursive(n - 1);

}

public static int sumFormula(int n) {

return n \* (n + 1) / 2;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

if (n <= 0) {

System.out.println("Please enter a valid natural number greater than 0.");

return;

}

int recursiveSum = sumRecursive(n);

int formulaSum = sumFormula(n);

System.out.println("Sum using recursion: " + recursiveSum);

System.out.println("Sum using formula: " + formulaSum);

if (recursiveSum == formulaSum) {

System.out.println("Both methods produce the same result.");

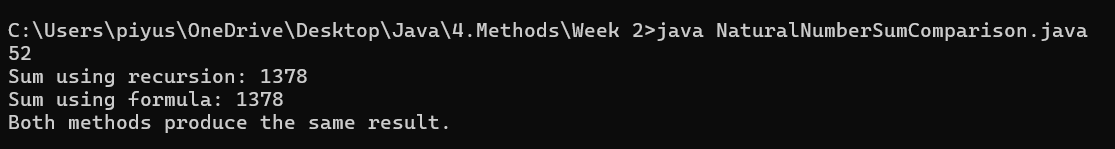
} else {

System.out.println("Mismatch in results.");

}

}

}



1. Write a program that takes a year as input and outputs the Year is a Leap Year or not

**Hint =>**

1. The LeapYear program only works for year >= 1582, corresponding to a year in the Gregorian calendar.
2. Also Leap year is divisible by 4 and not divisible by 100 or divisible by 400
3. Write a method to check for Leap Year using the conditions a and b

import java.util.Scanner;

public class LeapYearChecker {

public static boolean isLeapYear(int year) {

if (year < 1582) {

return false;

}

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int year = sc.nextInt();

if (year < 1582) {

System.out.println("Invalid input. Leap year check only works for year 1582 and later (Gregorian calendar).");

return;

}

if (isLeapYear(year)) {

System.out.println("Year " + year + " is a Leap Year.");

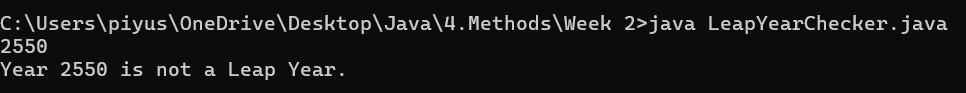
} else {

System.out.println("Year " + year + " is not a Leap Year.");

}

}

}



1. Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double convertKmToMiles(double km)* =>**

1. Method To convert kilometers to miles and return the value. Use the following code double km2miles = 0.621371;
2. Method to convert miles to kilometers and return the value. Use the following code double miles2km = 1.60934;
3. Method to convert meters to feet and return the value. Use the following code to convert double meters2feet = 3.28084;
4. Method to convert feet to meters and return the value. Use the following code to convert double feet2meters = 0.3048;

public class UnitConverter2 {

public static double convertKmToMiles(double km) {

double km2miles = 0.621371;

return km \* km2miles;

}

public static double convertMilesToKm(double miles) {

double miles2km = 1.60934;

return miles \* miles2km;

}

public static double convertMetersToFeet(double meters) {

double meters2feet = 3.28084;

return meters \* meters2feet;

}

public static double convertFeetToMeters(double feet) {

double feet2meters = 0.3048;

return feet \* feet2meters;

}

// Main method for quick testing

public static void main(String[] args) {

double km = 5.0;

double miles = 3.0;

double meters = 10.0;

double feet = 30.0;

System.out.println(km + " km = " + convertKmToMiles(km) + " miles");

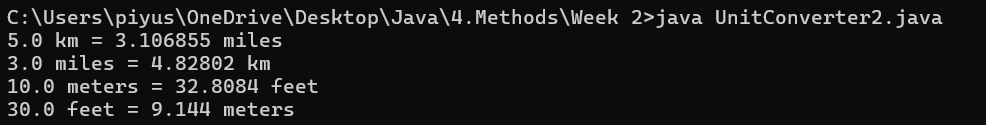
System.out.println(miles + " miles = " + convertMilesToKm(miles) + " km");

System.out.println(meters + " meters = " + convertMetersToFeet(meters) + " feet");

System.out.println(feet + " feet = " + convertFeetToMeters(feet) + " meters");

}

}



1. Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double convertYardsToFeet(double yards)* =>**

1. Method to convert yards to feet and return the value. Use following code to convert double yards2feet = 3;
2. Method to convert feet to yards and return the value. Use following code to convert double feet2yards = 0.333333;
3. Method to convert meters to inches and return the value. Use following code to convert double meters2inches = 39.3701;
4. Method to convert inches to meters and return the value. Use following code to convert double inches2meters = 0.0254;
5. Method to convert inches to centimeters and return the value. Use the following code double inches2cm = 2.54;

public class UnitConverter1 {

public static double convertYardsToFeet(double yards) {

double yards2feet = 3.0;

return yards \* yards2feet;

}

public static double convertFeetToYards(double feet) {

double feet2yards = 0.333333;

return feet \* feet2yards;

}

public static double convertMetersToInches(double meters) {

double meters2inches = 39.3701;

return meters \* meters2inches;

}

public static double convertInchesToMeters(double inches) {

double inches2meters = 0.0254;

return inches \* inches2meters;

}

public static double convertInchesToCentimeters(double inches) {

double inches2cm = 2.54;

return inches \* inches2cm;

}

// Main method to test the conversion methods

public static void main(String[] args) {

System.out.println("5 yards = " + convertYardsToFeet(5) + " feet");

System.out.println("9 feet = " + convertFeetToYards(9) + " yards");

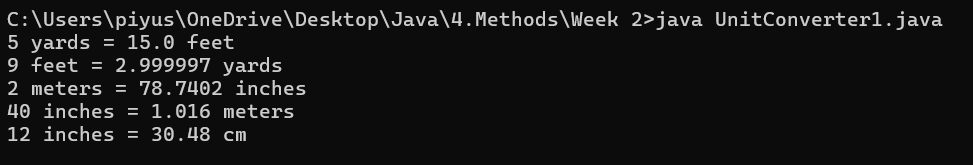
System.out.println("2 meters = " + convertMetersToInches(2) + " inches");

System.out.println("40 inches = " + convertInchesToMeters(40) + " meters");

System.out.println("12 inches = " + convertInchesToCentimeters(12) + " cm");

}

}



1. Extend or Create a ***UnitConvertor*** utility class similar to the one shown in the notes to do the following. Please define ***static*** methods for all the UnitConvertor class methods. E.g.

***public static double convertFarhenheitToCelsius(double farhenheit)* =>**

1. Method to convert Fahrenheit to Celsius and return the value. Use the following code double farhenheit2celsius = (farhenheit - 32) \* 5 / 9;
2. Method to convert Celsius to Fahrenheit and return the value. Use the following code double celsius2farhenheit = (celsius \* 9 / 5) + 32;
3. Method to convert pounds to kilograms and return the value. Use the following code double pounds2kilograms = 0.453592;
4. Method to convert kilograms to pounds and return the value. Use the following code double kilograms2pounds = 2.20462;
5. Method to convert gallons to liters and return the value. Use following code to convert double gallons2liters = 3.78541;
6. Method to convert liters to gallons and return the value. Use following code to convert double liters2gallons = 0.264172;

public class UnitConverter {

public static double convertFahrenheitToCelsius(double fahrenheit) {

return (fahrenheit - 32) \* 5 / 9;

}

public static double convertCelsiusToFahrenheit(double celsius) {

return (celsius \* 9 / 5) + 32;

}

public static double convertPoundsToKilograms(double pounds) {

double pounds2kilograms = 0.453592;

return pounds \* pounds2kilograms;

}

public static double convertKilogramsToPounds(double kilograms) {

double kilograms2pounds = 2.20462;

return kilograms \* kilograms2pounds;

}

public static double convertGallonsToLiters(double gallons) {

double gallons2liters = 3.78541;

return gallons \* gallons2liters;

}

public static double convertLitersToGallons(double liters) {

double liters2gallons = 0.264172;

return liters \* liters2gallons;

}

// Main method to test the conversions

public static void main(String[] args) {

System.out.println("98.6°F = " + convertFahrenheitToCelsius(98.6) + "°C");

System.out.println("37°C = " + convertCelsiusToFahrenheit(37) + "°F");

System.out.println("150 pounds = " + convertPoundsToKilograms(150) + " kg");

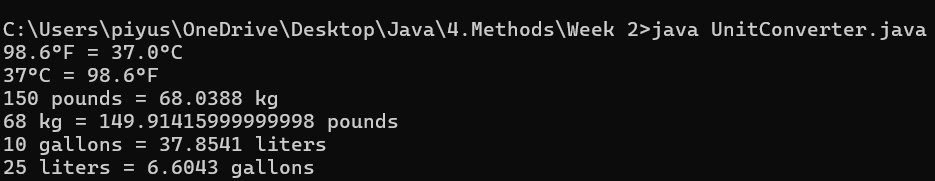
System.out.println("68 kg = " + convertKilogramsToPounds(68) + " pounds");

System.out.println("10 gallons = " + convertGallonsToLiters(10) + " liters");

System.out.println("25 liters = " + convertLitersToGallons(25) + " gallons");

}

}



1. Write a program to take user input for the age of all 10 students in a class and check whether the student can vote depending on his/her age is greater or equal to 18.

**Hint =>**

1. Create a class ***public class StudentVoteChecker*** and define a method ***public boolean canStudentVote(int age)*** which takes in age as a parameter and returns true or false
2. Inside the method firstly validate the age for a negative number, if a negative return is false cannot vote. For valid age check for age is 18 or above return true; else return false;
3. In the main function define an array of 10 integer elements, loop through the array by take user input for the student's age, call canStudentVote() and display the result

import java.util.Scanner;

public class StudentVoteChecker {

public static boolean canStudentVote(int age) {

if (age < 0) {

return false; // invalid age

}

return age >= 18;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[] ages = new int[10];

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

System.out.print("Enter age of student " + (i + 1) + ": ");

ages[i] = sc.nextInt();

boolean eligible = canStudentVote(ages[i]);

if (ages[i] < 0) {

System.out.println("Invalid age entered.");

} else if (eligible) {

System.out.println("Student " + (i + 1) + " is eligible to vote.");

} else {

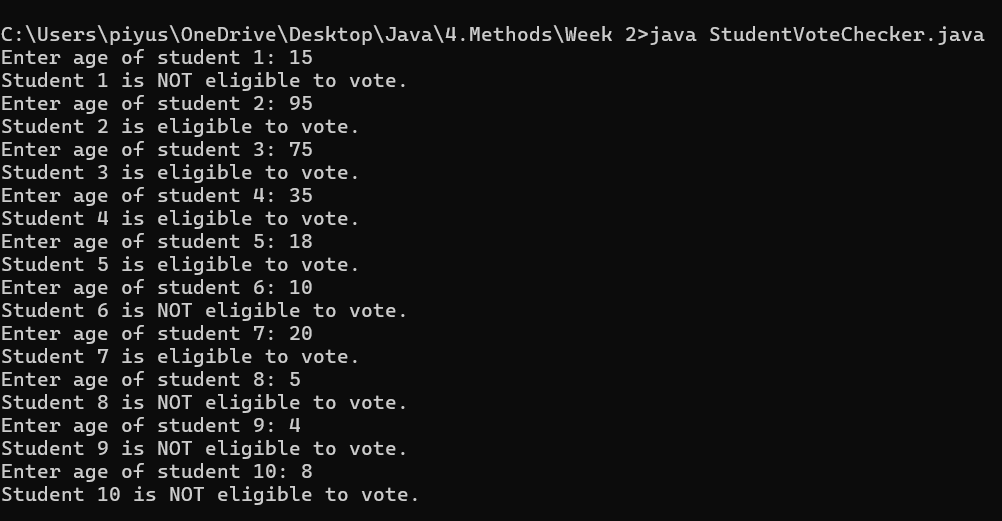
System.out.println("Student " + (i + 1) + " is NOT eligible to vote.");

}

}

}

}



1. Create a program to find the youngest friends among 3 Amar, Akbar and Anthony based on their ages and tallest among the friends based on their heights and display it

**Hint =>**

1. Take user input for age and height for the 3 friends and store it in two arrays each to store the values for age and height of the 3 friends
2. Write a Method to find the youngest of the 3 friends
3. Write a Method to find the tallest of the 3 friends

import java.util.Scanner;

public class FriendComparison {

public static int findYoungest(int[] ages) {

int minIndex = 0;

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

if (ages[i] < ages[minIndex]) {

minIndex = i;

}

}

return minIndex;

}

public static int findTallest(double[] heights) {

int maxIndex = 0;

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

if (heights[i] > heights[maxIndex]) {

maxIndex = i;

}

}

return maxIndex;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String[] names = {"Amar", "Akbar", "Anthony"};

int[] ages = new int[3];

double[] heights = new double[3];

// Take input

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

System.out.print("Enter age of " + names[i] + ": ");

ages[i] = sc.nextInt();

System.out.print("Enter height (in cm) of " + names[i] + ": ");

heights[i] = sc.nextDouble();

}

int youngestIndex = findYoungest(ages);

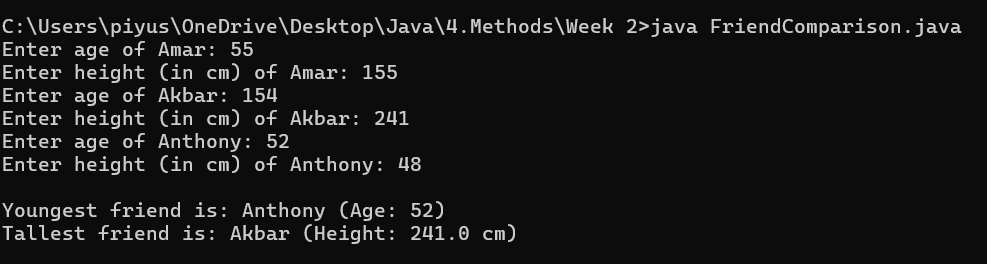
int tallestIndex = findTallest(heights);

System.out.println("\nYoungest friend is: " + names[youngestIndex] + " (Age: " + ages[youngestIndex] + ")");

System.out.println("Tallest friend is: " + names[tallestIndex] + " (Height: " + heights[tallestIndex] + " cm)");

}

}



1. Write a program to take user input for 5 numbers and check whether a number is positive or negative. Further for positive numbers check if the number is even or odd. Finally compare the first and last elements of the array and display if they are equal, greater, or less

**Hint =>**

1. Write a Method to Check whether the number is positive or negative
2. Write a Method to check whether the number is even or odd
3. Write a Method to compare two numbers and return 1 if number1 > number2 or 0 if both are equal or -1 if number1 < number2
4. In the main program, Loop through the array using the length call the method ***isPositive()*** and if positive call method ***isEven()*** and print accordingly
5. If the number is negative, print negative.
6. Finally compare the first and last element of the array by calling the method ***compare()*** and display if they are equal, greater, or less

import java.util.Scanner;

public class NumberAnalysis {

public static boolean isPositive(int number) {

return number >= 0;

}

public static boolean isEven(int number) {

return number % 2 == 0;

}

public static int compare(int num1, int num2) {

if (num1 > num2) return 1;

else if (num1 == num2) return 0;

else return -1;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[] numbers = new int[5];

// Take user input

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

System.out.print("Enter number " + (i + 1) + ": ");

numbers[i] = sc.nextInt();

if (isPositive(numbers[i])) {

System.out.print("Number is Positive and ");

if (isEven(numbers[i])) {

System.out.println("Even");

} else {

System.out.println("Odd");

}

} else {

System.out.println("Number is Negative");

}

}

// Compare first and last element

int result = compare(numbers[0], numbers[4]);

System.out.print("\nComparison between first and last number: ");

if (result == 1) {

System.out.println("First number is greater than the last.");

} else if (result == 0) {

System.out.println("First and last numbers are equal.");

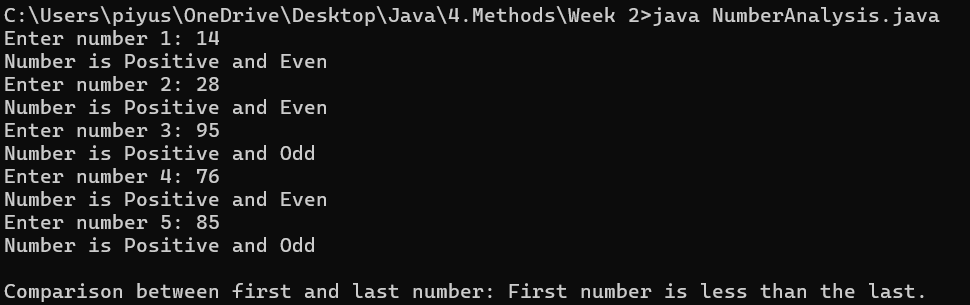
} else {

System.out.println("First number is less than the last.");

}

}

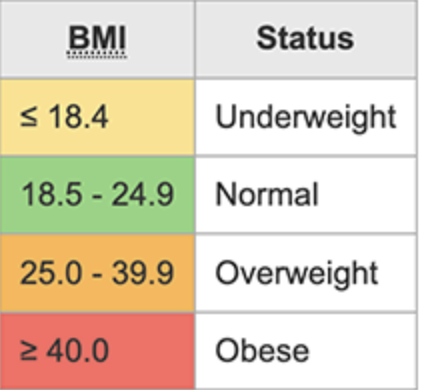
}



1. An organization took up the exercise to find the Body Mass Index (BMI) of all the persons in the team of 10 members. For this create a program to find the BMI and display the height, weight, BMI and status of each individual

**Hint =>**

1. Take user input in double for the weight (in kg) of the person and height (in cm) for the person and and store it in the corresponding 2D array of 10 rows and 3 columns. The First Column storing the weight, the second column storing the height in cm and the third column is the BMI
2. Create a Method to find the BMI of every person and populate the array. Use the formula BMI = weight / (height \* height). Note unit is kg/m^2. For this convert cm to meter
3. Create a Method to determine the BMI status using the logic shown in the figure below. and return the array of all the persons BMI Status.



import java.util.Scanner;

public class BMICalculator {

public static void calculateBMI(double[][] data) {

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

double weight = data[i][0]; // in kg

double heightInCm = data[i][1]; // in cm

double heightInMeters = heightInCm / 100.0;

double bmi = weight / (heightInMeters \* heightInMeters);

data[i][2] = bmi; // Store BMI in the third column

}

}

public static String[] getBMIStatus(double[][] data) {

String[] status = new String[data.length];

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

double bmi = data[i][2];

if (bmi < 18.5) {

status[i] = "Underweight";

} else if (bmi < 25) {

status[i] = "Normal";

} else if (bmi < 40) {

status[i] = "Overweight";

} else {

status[i] = "Obese";

}

}

return status;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

double[][] data = new double[10][3]; // [weight, height(cm), BMI]

// Step 1: Get input

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

System.out.print("Enter weight (kg) of person " + (i + 1) + ": ");

data[i][0] = sc.nextDouble();

System.out.print("Enter height (cm) of person " + (i + 1) + ": ");

data[i][1] = sc.nextDouble();

}

// Step 2: Calculate BMI

calculateBMI(data);

// Step 3: Get BMI status

String[] status = getBMIStatus(data);

// Step 4: Display result

System.out.println("\n--- BMI Results ---");

System.out.println("Person\tWeight(kg)\tHeight(cm)\tBMI\t\tStatus");

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

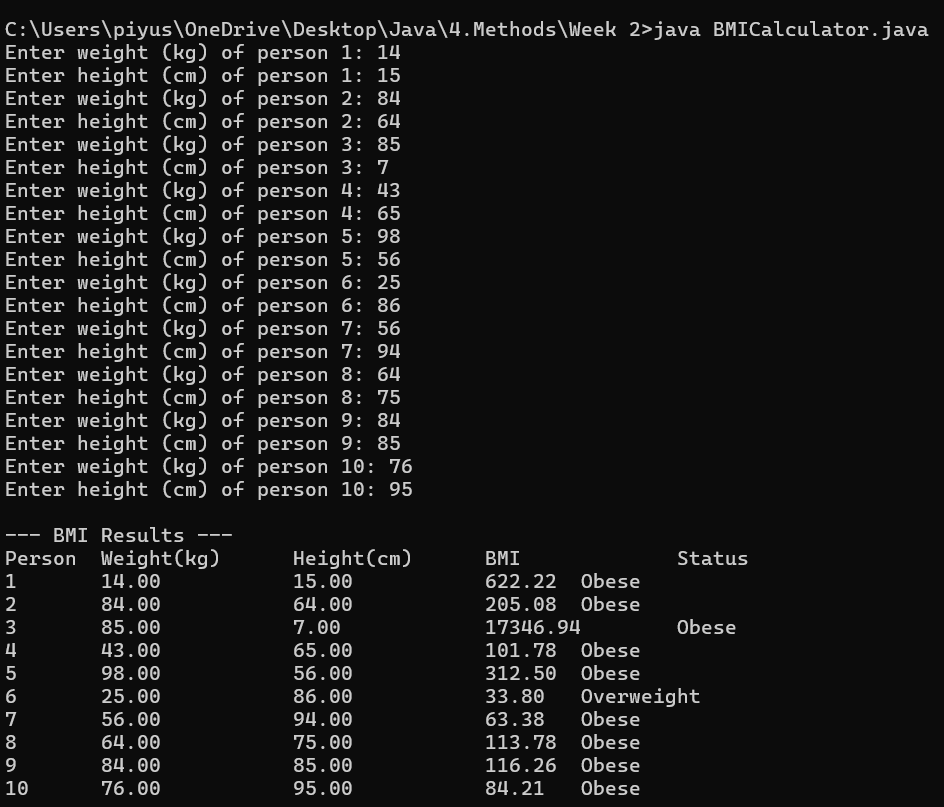
System.out.printf("%d\t%.2f\t\t%.2f\t\t%.2f\t%s\n",

(i + 1), data[i][0], data[i][1], data[i][2], status[i]);

}

}

}



1. Write a program Quadratic to find the roots of the equation . Use Math functions ***Math.pow()*** and ***Math.sqrt()***

**Hint =>**

1. Take a, b, and c as input values to find the roots of x.
2. The roots are computed using the following formulae

If delta is positive the find the two roots using formulae

If delta is zero then there is only one root of x

If delta is negative return empty array or nothing

1. Write a Method to find find the roots of a quadratic equation and return the roots.

import java.util.Scanner;

public class Quadratic {

public static double[] findRoots(double a, double b, double c) {

double delta = Math.pow(b, 2) - 4 \* a \* c;

if (delta > 0) {

double root1 = (-b + Math.sqrt(delta)) / (2 \* a);

double root2 = (-b - Math.sqrt(delta)) / (2 \* a);

return new double[]{root1, root2};

} else if (delta == 0) {

double root = -b / (2 \* a);

return new double[]{root};

} else {

// delta < 0: no real roots

return new double[0];

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter value of a: ");

double a = sc.nextDouble();

System.out.print("Enter value of b: ");

double b = sc.nextDouble();

System.out.print("Enter value of c: ");

double c = sc.nextDouble();

if (a == 0) {

System.out.println("Not a quadratic equation (a cannot be 0).");

return;

}

double[] roots = findRoots(a, b, c);

if (roots.length == 2) {

System.out.printf("Two real and distinct roots: %.2f and %.2f\n", roots[0], roots[1]);

} else if (roots.length == 1) {

System.out.printf("One real root: %.2f\n", roots[0]);

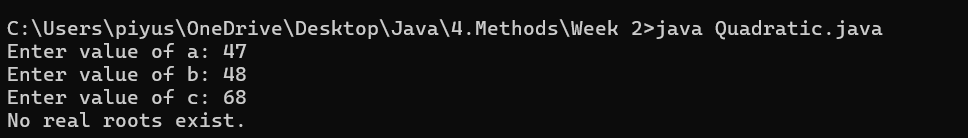
} else {

System.out.println("No real roots exist.");

}

}

}



1. Write a program that generates five 4 digit random values and then finds their average value, and their minimum and maximum value. Use Math.random(), Math.min(), and Math.max().

**Hint =>**

1. Write a method that generates array of 4 digit random numbers given the size as a parameter as shown in the method signature

***public int[] generate4DigitRandomArray(int size)***

1. Write a method to find average, min and max value of an array

***public double[] findAverageMinMax(int[] numbers)***

import java.util.Arrays;

public class RandomStats {

// a. Method to generate an array of 4-digit random numbers

public static int[] generate4DigitRandomArray(int size) {

int[] numbers = new int[size];

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

numbers[i] = (int)(Math.random() \* 9000) + 1000; // 1000 to 9999

}

return numbers;

}

// b. Method to find average, min and max value of an array

public static double[] findAverageMinMax(int[] numbers) {

int min = numbers[0];

int max = numbers[0];

int sum = 0;

for (int num : numbers) {

min = Math.min(min, num);

max = Math.max(max, num);

sum += num;

}

double average = (double) sum / numbers.length;

return new double[]{average, min, max};

}

public static void main(String[] args) {

int[] randomNumbers = generate4DigitRandomArray(5);

System.out.println("Generated 4-digit random numbers: " + Arrays.toString(randomNumbers));

double[] results = findAverageMinMax(randomNumbers);

System.out.printf("Average: %.2f\n", results[0]);

System.out.printf("Minimum: %.0f\n", results[1]);

System.out.printf("Maximum: %.0f\n", results[2]);

}

}

