Week 5 - Level 2 - 12 Practice Problems

QUES 1

```
import java.util.Scanner;
class FactorOperations {
    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;
                index++;
            }
        }
        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 double findSumOfSquares(int[] factors) {
    double sumOfSquares = 0;
    for (int factor : factors) {
        sumOfSquares += Math.pow(factor, 2);
    }
    return sumOfSquares;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
```

```
        int number = scanner.nextInt();
        int[] factors = findFactors(number);
        System.out.print("Factors of " + number + ": ");
        for (int factor : factors) {
            System.out.print(factor + " ");
        }
        System.out.println();
        int sum = findSum(factors);
        System.out.println("Sum of factors: " + sum);
        int product = findProduct(factors);
        System.out.println("Product of factors: " + product);
        double sumOfSquares = findSumOfSquares(factors);
        System.out.println("Sum of squares of factors: " +
sumOfSquares);


        scanner.close();
    }
}
```

OUTPUT:

```
Enter a number: 6
Factors of 6: 1 2 3 6
Sum of factors: 12
Product of factors: 36
Sum of squares of factors: 50.0
```

QUES2

```
import java.util.Scanner;
class NaturalNumberSum{
    public static int sumUsingRecursion(int n) {
        if (n == 0) {
            return 0;
        } else {
            return n + sumUsingRecursion(n - 1);
        }
    }
    public static int sumUsingFormula(int n) {
        return n * (n + 1) / 2;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a natural number (n): ");
        int n = scanner.nextInt();
        if (n < 1) {
            System.out.println("Please enter a valid natural
number greater than 0.");
            return;
        }
        int recursiveSum = sumUsingRecursion(n);
        int formulaSum = sumUsingFormula(n);
```

```
        System.out.println("Sum of first " + n + " natural
numbers using recursion: " + recursiveSum);
        System.out.println("Sum of first " + n + " natural
numbers using the formula: " + formulaSum);
        if (recursiveSum == formulaSum) {
            System.out.println("Both methods give the correct
and same result.");
        } else {
            System.out.println("There is an error in the
computations.");
        }

        scanner.close();
    }
}
```

OUTPUT:

```
Enter a natural number (n): 10
Sum of first 10 natural numbers using recursion: 55
Sum of first 10 natural numbers using the formula: 55
Both methods give the correct and same result.
```

QUES 3

```
import java.util.Scanner;
class LeapYear {
    public static boolean isLeapYear(int year) {
        if (year < 1582) {
```

```
            System.out.println("Year must be 1582 or later, as
per the Gregorian calendar.");
            return false;
        }
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400
== 0)) {
            return true;
        } else {
            return false;
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a year: ");
        int year = scanner.nextInt();
        if (isLeapYear(year)) {
            System.out.println(year + " is a Leap Year.");
        } else {
            System.out.println(year + " is not a Leap Year.");
        }

        scanner.close();
    }
}
```

OUTPUT:

Enter a year: 2024

2024 is a Leap Year.

QUES 4

```
class UnitConverter {
    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;
    }
    public static void main(String[] args) {
        double km = 10;
        double miles = 6.21;
```

```
        double meters = 100;
        double feet = 328.084;
        System.out.println(km + " kilometers is equal to " +
convertKmToMiles(km) + " miles.");
        System.out.println(miles + " miles is equal to " +
convertMilesToKm(miles) + " kilometers.");
        System.out.println(meters + " meters is equal to " +
convertMetersToFeet(meters) + " feet.");
        System.out.println(feet + " feet is equal to " +
convertFeetToMeters(feet) + " meters.");
    }
}
```

OUTPUT:

10.0 kilometers is equal to 6.21371 miles.

6.21 miles is equal to 9.9777514 kilometers.

100.0 meters is equal to 328.084 feet.

328.084 feet is equal to 100.0000032 meters.

QUES 5

```
class UnitConverter5 {
    public static double convertYardsToFeet(double yards) {
        double yards2feet = 3;
        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;
    }
    public static void main(String[] args) {
        double yards = 5;
        double feet = 15;
        double meters = 100;
        double inches = 50;
        System.out.println(yards + " yards is equal to " +
convertYardsToFeet(yards) + " feet.");
        System.out.println(feet + " feet is equal to " +
convertFeetToYards(feet) + " yards.");
```

```
        System.out.println(meters + " meters is equal to " +
convertMetersToInches(meters) + " inches.");
        System.out.println(inches + " inches is equal to " +
convertInchesToMeters(inches) + " meters.");
        System.out.println(inches + " inches is equal to " +
convertInchesToCentimeters(inches) + " centimeters.");
    }
}
```

OUTPUT:

5.0 yards is equal to 15.0 feet.

15.0 feet is equal to 5.0 yards.

100.0 meters is equal to 3937.01 inches.

50.0 inches is equal to 1.27 meters.

50.0 inches is equal to 127.0 centimeters.

QUES 6

```
class UnitConverter6 {
    public static double convertFahrenheitToCelsius(double
fahrenheit) {
        double fahrenheit2celsius = (fahrenheit - 32) * 5 / 9;
        return fahrenheit2celsius;
    }
    public static double convertCelsiusToFahrenheit(double
celsius) {
        double celsius2fahrenheit = (celsius * 9 / 5) + 32;
        return celsius2fahrenheit;
    }
```

```
    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 = 1 / 3.78541;
        return liters * liters2gallons;
    }
    public static void main(String[] args) {
        double fahrenheit = 100;
        double celsius = 37;
        double pounds = 150;
        double kilograms = 68;
```

```
        double gallons = 5;
        double liters = 20;
        System.out.println(fahrenheit + " Fahrenheit is equal to
" + convertFahrenheitToCelsius(fahrenheit) + " Celsius.");
        System.out.println(celsius + " Celsius is equal to " +
convertCelsiusToFahrenheit(celsius) + " Fahrenheit.");
        System.out.println(pounds + " pounds is equal to " +
convertPoundsToKilograms(pounds) + " kilograms.");
        System.out.println(kilograms + " kilograms is equal to "
+ convertKilogramsToPounds(kilograms) + " pounds.");
        System.out.println(gallons + " gallons is equal to " +
convertGallonsToLiters(gallons) + " liters.");
        System.out.println(liters + " liters is equal to " +
convertLitersToGallons(liters) + " gallons.");
    }
}
```

OUTPUT:

100.0 Fahrenheit is equal to 37.77777777777778 Celsius.

37.0 Celsius is equal to 98.6 Fahrenheit.

150.0 pounds is equal to 68.0388 kilograms.

68.0 kilograms is equal to 149.91416 pounds.

5.0 gallons is equal to 18.92705 liters.

20.0 liters is equal to 5.28344 gallons.

QUES 7

```
import java.util.Scanner;
class StudentVote{
    public boolean canStudentVote(int age) {
        if (age < 0) {
            return false;
        }
        if (age >= 18) {
            return true;
        } else {
            return false;
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] ages = new int[10];
        StudentVoteChecker checker = new
StudentVoteChecker();
        for (int i = 0; i < 10; i++) {
            System.out.print("Enter age for student " + (i + 1) + ":
");
            ages[i] = scanner.nextInt();
            if (checker.canStudentVote(ages[i])) {
                System.out.println("Student " + (i + 1) + " can
vote.");
```

```
            } else {
                System.out.println("Student " + (i + 1) + " cannot
vote.");
            }
        }
        scanner.close();
    }
}
```

INPUT:

Enter age for student 1: 20

Enter age for student 2: 15

Enter age for student 3: 18

Enter age for student 4: -5

Enter age for student 5: 22

Enter age for student 6: 17

Enter age for student 7: 30

Enter age for student 8: 19

Enter age for student 9: 25

Enter age for student 10: 16

OUTPUT:

Student 1 can vote.

Student 2 cannot vote.

Student 3 can vote.

Student 4 cannot vote.

Student 5 can vote.

Student 6 cannot vote.

Student 7 can vote.

Student 8 can vote.

Student 9 can vote.

Student 10 cannot vote.

QUES 8

```
import java.util.Scanner;
class YoungestAndTallest {
    public static String findYoungest(int[] ages, String[]
names) {
        int youngestAge = ages[0];
        String youngestFriend = names[0];
        for (int i = 1; i < ages.length; i++) {
            if (ages[i] < youngestAge) {
                youngestAge = ages[i];
                youngestFriend = names[i];
            }
        }
        return youngestFriend;
    }
    public static String findTallest(double[] heights, String[]
names) {
        double tallestHeight = heights[0];
        String tallestFriend = names[0];
```

```
        for (int i = 1; i < heights.length; i++) {
            if (heights[i] > tallestHeight) {
                tallestHeight = heights[i];
                tallestFriend = names[i];
            }
        }
        return tallestFriend;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] names = {"Amar", "Akbar", "Anthony"};
        int[] ages = new int[3];
        double[] heights = new double[3];
        for (int i = 0; i < 3; i++) {
            System.out.print("Enter age for " + names[i] + ": ");
            ages[i] = scanner.nextInt();
            System.out.print("Enter height for " + names[i] + " (in
meters): ");
            heights[i] = scanner.nextDouble();
        }
        String youngestFriend = findYoungest(ages, names);
        String tallestFriend = findTallest(heights, names);
        System.out.println("The youngest friend is: " +
youngestFriend);
```

```
        System.out.println("The tallest friend is: " +
tallestFriend);
        scanner.close();
    }
}
```

INPUT:

```
Enter age for Amar: 25
Enter height for Amar (in meters): 1.75
Enter age for Akbar: 22
Enter height for Akbar (in meters): 1.80
Enter age for Anthony: 28
Enter height for Anthony (in meters): 1.70
```

OUTPUT:

```
The youngest friend is: Akbar
The tallest friend is: Akbar
```

QUES 9

```
import java.util.Scanner;
class NumberCheck {
    public static boolean isPositive(int number) {
        return number >= 0;
    }
    public static String isEven(int number) {
        if (number % 2 == 0) {
            return "even";
        } else {
```

```
            return "odd";
        }
    }
    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 scanner = new Scanner(System.in);
        int[] numbers = new int[5];
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter number " + (i + 1) + ": ");
            numbers[i] = scanner.nextInt();
        }
        for (int i = 0; i < numbers.length; i++) {
            int number = numbers[i];
            if (isPositive(number)) {
                System.out.println(number + " is positive and " +
isEven(number) + ".");
            } else {
```

```
            System.out.println(number + " is negative.");
        }
    }
    int result = compare(numbers[0], numbers[4]);
    if (result == 1) {
        System.out.println("The first number is greater than
the last number.");
    } else if (result == 0) {
        System.out.println("The first and last numbers are
equal.");
    } else {
        System.out.println("The first number is less than the
last number.");
    }

    scanner.close();  // Close the scanner
  }
}
```

INPUT:

Enter number 1: 12

Enter number 2: -7

Enter number 3: 9

Enter number 4: 4

Enter number 5: -15

OUTPUT:

12 is positive and even.

-7 is negative.

9 is positive and odd.

4 is positive and even.

-15 is negative.

The first number is greater than the last number.

QUES 10

```
import java.util.Scanner;
class BMICalculator {
    public static void calculateBMI(double[][] data) {
        for (int i = 0; i < data.length; i++) {
            double heightInMeters = data[i][1] / 100.0;
            double bmi = data[i][0] / (heightInMeters *
heightInMeters);
            data[i][2] = bmi;
        }
    }
    public static String determineBMIStatus(double bmi) {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi >= 18.5 && bmi < 24.9) {
            return "Normal weight";
        } else if (bmi >= 25 && bmi < 29.9) {
            return "Overweight";
```

```
        } else {
            return "Obese";
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double[][] data = new double[10][3];
        for (int i = 0; i < 10; i++) {
            System.out.print("Enter weight (kg) for person " + (i
+ 1) + ": ");
            data[i][0] = scanner.nextDouble();

            System.out.print("Enter height (cm) for person " + (i
+ 1) + ": ");
            data[i][1] = scanner.nextDouble();
        }
        calculateBMI(data);
        System.out.println("\nBMI Calculation Results:");
        for (int i = 0; i < 10; i++) {
            double bmi = data[i][2];
            String status = determineBMIStatus(bmi);

            System.out.printf("Person %d - Weight: %.2f kg,
Height: %.2f cm, BMI: %.2f, Status: %s\n",
                    (i + 1), data[i][0], data[i][1], bmi, status);
```

```
        }

        scanner.close();
    }
}
```

INPUT:

```
Enter weight (kg) for person 1: 65
Enter height (cm) for person 1: 170
Enter weight (kg) for person 2: 85
Enter height (cm) for person 2: 180
Enter weight (kg) for person 3: 72
Enter height (cm) for person 3: 160
Enter weight (kg) for person 4: 50
Enter height (cm) for person 4: 155
Enter weight (kg) for person 5: 90
Enter height (cm) for person 5: 175
Enter weight (kg) for person 6: 100
Enter height (cm) for person 6: 160
Enter weight (kg) for person 7: 58
Enter height (cm) for person 7: 165
Enter weight (kg) for person 8: 80
Enter height (cm) for person 8: 178
Enter weight (kg) for person 9: 95
Enter height (cm) for person 9: 185
```

Enter weight (kg) for person 10: 55

Enter height (cm) for person 10: 150

OUTPUT:

BMI Calculation Results:

Person 1 - Weight: 65.00 kg, Height: 170.00 cm, BMI: 22.49, Status: Normal weight

Person 2 - Weight: 85.00 kg, Height: 180.00 cm, BMI: 26.23, Status: Overweight

Person 3 - Weight: 72.00 kg, Height: 160.00 cm, BMI: 28.12, Status: Overweight

Person 4 - Weight: 50.00 kg, Height: 155.00 cm, BMI: 20.81, Status: Normal weight

Person 5 - Weight: 90.00 kg, Height: 175.00 cm, BMI: 29.39, Status: Overweight

Person 6 - Weight: 100.00 kg, Height: 160.00 cm, BMI: 39.06, Status: Obese

Person 7 - Weight: 58.00 kg, Height: 165.00 cm, BMI: 21.26, Status: Normal weight

Person 8 - Weight: 80.00 kg, Height: 178.00 cm, BMI: 25.28, Status: Overweight

Person 9 - Weight: 95.00 kg, Height: 185.00 cm, BMI: 27.74, Status: Overweight

Person 10 - Weight: 55.00 kg, Height: 150.00 cm, BMI: 24.44, Status: Normal weight

QUES 11

```
import java.util.Scanner;
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 {
            return new double[]{};
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter coefficient a: ");
        double a = scanner.nextDouble();
        System.out.print("Enter coefficient b: ");
        double b = scanner.nextDouble();
        System.out.print("Enter coefficient c: ");
        double c = scanner.nextDouble();
```

```
        double[] roots = findRoots(a, b, c);
        if (roots.length == 2) {
            System.out.printf("The two roots are: %.2f and
%.2f\n", roots[0], roots[1]);
        } else if (roots.length == 1) {
            System.out.printf("The root is: %.2f\n", roots[0]);
        } else {
            System.out.println("No real roots exist.");
        }

        scanner.close();
    }
}
```

INPUT:

Enter coefficient a: 1

Enter coefficient b: -3

Enter coefficient c: 2

OUTPUT:

The roots are real and distinct:

Root 1: 2.0

Root 2: 1.0

QUES 12

```
import java.util.Arrays;
class RandomNumberAnalysis {
    public static int[] generate4DigitRandomArray(int size) {
        int[] randomNumbers = new int[size];
        for (int i = 0; i < size; i++) {
            randomNumbers[i] = 1000 + (int)(Math.random() *
9000);
        }
        return randomNumbers;
    }
    public static double[] findAverageMinMax(int[] numbers)
{
        double sum = 0;
        int min = numbers[0];
        int max = numbers[0];
        for (int num : numbers) {
            sum += num;
            min = Math.min(min, num);
            max = Math.max(max, num);
        }
        double average = sum / numbers.length;
        return new double[]{average, min, max};
    }
    public static void main(String[] args) {
```

```
        int[] randomNumbers =
generate4DigitRandomArray(5);
        double[] result =
findAverageMinMax(randomNumbers);
        System.out.println("Generated random 4-digit numbers:
" + Arrays.toString(randomNumbers));
        System.out.printf("Average: %.2f\n", result[0]);
        System.out.println("Minimum value: " + (int)result[1]);
        System.out.println("Maximum value: " + (int)result[2]);
    }
}
```

OUTPUT:

Generated random 4-digit numbers: [7253, 4942, 8435, 6398, 1189]

Average: 5635.40

Minimum value: 1189

Maximum value: 8435