Java Level 3 Methods - Programs

1. Football Team Heights

```
public class FootballTeamHeights {
    public static int[] generateHeights() {
        int[] heights = new int[11];
        for (int i = 0; i < heights.length; i++) {
            heights[i] = (int)(Math.random() * 101) + 150;
        }
        return heights;
    }

    public static int sum(int[] array) {
        int sum = 0;
        for (int i : array) sum += i;
        return sum;
    }

    public static double mean(int[] array) {
        return (double)sum(array) / array.length;
    }

    public static int min(int[] array) {
        int min = array[0];
        for (int i : array) if (i < min) min = i;
        return min;
    }

    public static int max(int[] array) {
        int max = array[0];
        for (int i : array) if (i > max) max = i;
        return max;
    }

    public static void main(String[] args) {
        int[] heights = generateHeights();
        System.out.print("Player Heights: ");
        for (int h : heights) System.out.print(h + " ");
        System.out.println("\nShortest Height: " + min(heights));
        System.out.println("Tallest Height: " + max(heights));
        System.out.println("Mean Height: " + mean(heights));
    }
}
```

Java Level 3 Methods - Programs

2. Number Checker Utility (Duck, Armstrong, Largest/Smallest)

```
public class NumberCheckerUtility1 {
    public static int countDigits(int number) {
        int count = 0;
        while (number > 0) {
            number /= 10;
            count++;
        }
        return count;
    }

    public static int[] getDigits(int number) {
        int count = countDigits(number);
        int[] digits = new int[count];
        for (int i = count - 1; i >= 0; i--) {
            digits[i] = 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 number) {
        int[] digits = getDigits(number);
        int sum = 0;
        for (int digit : digits) {
            sum += Math.pow(digit, digits.length);
        }
        return sum == number;
    }

    public static int[] findTwoLargest(int[] digits) {
        int largest = Integer.MIN_VALUE, second = Integer.MIN_VALUE;
        for (int n : digits) {
            if (n > largest) {
                second = largest;
                largest = n;
            } else if (n > second && n != largest) {
                second = n;
            }
        }
        return new int[]{largest, second};
    }
```

```
    public static int[] findTwoSmallest(int[] digits) {
        int smallest = Integer.MAX_VALUE, second = Integer.MAX_VALUE;
        for (int n : digits) {
            if (n < smallest) {
                second = smallest;
                smallest = n;
            } else if (n < second && n != smallest) {
                second = n;
            }
        }
        return new int[]{smallest, second};
    }

    public static void main(String[] args) {
        int number = 153;
        int[] digits = getDigits(number);
        System.out.println("Number: " + number);
        System.out.println("Is Duck Number? " + isDuckNumber(digits));
        System.out.println("Is Armstrong Number? " + isArmstrongNumber(number));
        int[] largest = findTwoLargest(digits);
              System.out.println("Largest:  "  +  largest[0]  +  ",  Second  Largest:  "  +
largest[1]);
        int[] smallest = findTwoSmallest(digits);
             System.out.println("Smallest:  "  +  smallest[0]  +  ",  Second  Smallest:  "  +
smallest[1]);
    }
}
```

3. Number Checker Utility (Harshad, Squares, Frequency)

```
public class NumberCheckerUtility2 {
    public static int[] getDigits(int number) {
        int count = String.valueOf(number).length();
        int[] digits = new int[count];
        for (int i = count - 1; i >= 0; i--) {
            digits[i] = number % 10;
            number /= 10;
        }
        return digits;
    }

    public static int sumOfDigits(int[] digits) {
        int sum = 0;
        for (int d : digits) sum += d;
        return sum;
    }

    public static int sumOfSquares(int[] digits) {
        int sum = 0;
        for (int d : digits) sum += Math.pow(d, 2);
        return sum;
    }

    public static boolean isHarshadNumber(int number) {
        int[] digits = getDigits(number);
        int sum = sumOfDigits(digits);
        return number % sum == 0;
    }

    public static int[][] digitFrequency(int[] digits) {
        int[] freq = new int[10];
        for (int d : digits) freq[d]++;
        int[][] result = new int[10][2];
        for (int i = 0; i < 10; i++) {
            result[i][0] = i;
            result[i][1] = freq[i];
        }
        return result;
    }

    public static void main(String[] args) {
        int number = 21;
        int[] digits = getDigits(number);
        System.out.println("Sum of digits: " + sumOfDigits(digits));
        System.out.println("Sum of squares: " + sumOfSquares(digits));
        System.out.println("Is Harshad Number? " + isHarshadNumber(number));
        int[][] frequency = digitFrequency(digits);
        System.out.println("Digit Frequencies:");
```

```
        for (int[] row : frequency) {
            if (row[1] > 0)
                System.out.println("Digit: " + row[0] + " Frequency: " + row[1]);
        }
    }
}
```

4. Number Checker Utility (Palindrome, Duck)

```
public class NumberCheckerUtility3 {
    public static int[] getDigits(int number) {
        String str = String.valueOf(number);
        int[] digits = new int[str.length()];
        for (int i = 0; i < str.length(); i++) {
            digits[i] = str.charAt(i) - '0';
        }
        return digits;
    }

    public static int[] reverseDigits(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 areEqual(int[] a, int[] b) {
        if (a.length != b.length) return false;
        for (int i = 0; i < a.length; i++) {
            if (a[i] != b[i]) return false;
        }
        return true;
    }

    public static boolean isPalindrome(int number) {
        int[] digits = getDigits(number);
        int[] reversed = reverseDigits(digits);
        return areEqual(digits, reversed);
    }

    public static boolean isDuckNumber(int[] digits) {
        for (int i = 1; i < digits.length; i++) {
            if (digits[i] == 0) return true;
        }
        return false;
    }

    public static void main(String[] args) {
        int number = 121;
        int[] digits = getDigits(number);
        System.out.println("Is Palindrome? " + isPalindrome(number));
        System.out.println("Is Duck Number? " + isDuckNumber(digits));
    }
}
```

5. Number Checker Utility (Prime, Neon, Spy, Automorphic, Buzz)

```
public class NumberCheckerUtility4 {
    public static boolean isPrime(int number) {
        if (number <= 1) return false;
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) return false;
        }
        return true;
    }

    public static boolean isNeon(int number) {
        int square = number * number;
        int sum = 0;
        while (square > 0) {
            sum += square % 10;
            square /= 10;
        }
        return sum == number;
    }

    public static boolean isSpy(int number) {
        int sum = 0, prod = 1;
        while (number > 0) {
            int digit = number % 10;
            sum += digit;
            prod *= digit;
            number /= 10;
        }
        return sum == prod;
    }

    public static boolean isAutomorphic(int number) {
        int square = number * number;
        return String.valueOf(square).endsWith(String.valueOf(number));
    }

    public static boolean isBuzz(int number) {
        return number % 7 == 0 || number % 10 == 7;
    }

    public static void main(String[] args) {
        int number = 7;
        System.out.println("Is Prime? " + isPrime(number));
        System.out.println("Is Neon? " + isNeon(number));
        System.out.println("Is Spy? " + isSpy(number));
        System.out.println("Is Automorphic? " + isAutomorphic(number));
        System.out.println("Is Buzz? " + isBuzz(number));
    }
}
```

6. Number Checker Utility (Factors, Perfect, Abundant, Deficient, Strong)

```
public class NumberCheckerUtility5 {
    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 idx = 0;
        for (int i = 1; i <= number; i++) {
            if (number % i == 0) factors[idx++] = i;
        }
        return factors;
    }

    public static int greatestFactor(int[] factors) {
        int max = 0;
        for (int f : factors) if (f > max) max = f;
        return max;
    }

    public static int sumFactors(int[] factors) {
        int sum = 0;
        for (int f : factors) sum += f;
        return sum;
    }

    public static int productFactors(int[] factors) {
        int product = 1;
        for (int f : factors) product *= f;
        return product;
    }

    public static double cubeProduct(int[] factors) {
        double product = 1;
        for (int f : factors) product *= Math.pow(f, 3);
        return product;
    }

    public static boolean isPerfect(int number) {
        int sum = 0;
        for (int i = 1; i < number; i++) {
            if (number % i == 0) sum += i;
        }
        return sum == number;
    }

    public static boolean isAbundant(int number) {
        int sum = 0;
```

```
        for (int i = 1; i < number; i++) {
            if (number % i == 0) sum += i;
        }
        return sum > number;
    }

    public static boolean isDeficient(int number) {
        int sum = 0;
        for (int i = 1; i < number; i++) {
            if (number % i == 0) sum += i;
        }
        return sum < number;
    }

    public static boolean isStrong(int number) {
        int temp = number, sum = 0;
        while (temp > 0) {
            int digit = temp % 10;
            sum += factorial(digit);
            temp /= 10;
        }
        return sum == number;
    }

    public static int factorial(int n) {
        int result = 1;
        for (int i = 2; i <= n; i++) result *= i;
        return result;
    }

    public static void main(String[] args) {
        int number = 28;
        int[] factors = findFactors(number);
        System.out.println("Factors: ");
        for (int f : factors) System.out.print(f + " ");
        System.out.println("\nGreatest: " + greatestFactor(factors));
        System.out.println("Sum: " + sumFactors(factors));
        System.out.println("Product: " + productFactors(factors));
        System.out.println("Cube Product: " + cubeProduct(factors));
        System.out.println("Is Perfect: " + isPerfect(number));
        System.out.println("Is Abundant: " + isAbundant(number));
        System.out.println("Is Deficient: " + isDeficient(number));
        System.out.println("Is Strong: " + isStrong(number));
    }
}
```

7. OTP Validator

```
public class OtpValidator {
    public static int generateOtp() {
        return (int)(Math.random() * 900000) + 100000;
    }

    public static boolean areUnique(int[] otps) {
        for (int i = 0; i < otps.length; i++) {
            for (int j = i + 1; j < otps.length; j++) {
                if (otps[i] == otps[j]) return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        int[] otps = new int[10];
        for (int i = 0; i < otps.length; i++) {
            otps[i] = generateOtp();
            System.out.println("OTP " + (i+1) + ": " + otps[i]);
        }
        System.out.println("All OTPs Unique? " + areUnique(otps));
    }
}
```

8. Calendar Display

```
import java.util.Scanner;

public class CalendarDisplay {
     static String[] months = { "January", "February", "March", "April", "May", "June",
"July",
            "August", "September", "October", "November", "December" };
    static int[] daysInMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

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

    public static int getFirstDayOfMonth(int year, int month) {
        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;
        return (d + x + (31*m0)/12) % 7;
    }

    public static void displayCalendar(int month, int year) {
        int days = daysInMonth[month - 1];
        if (month == 2 && isLeapYear(year)) days = 29;

        System.out.println("\n  " + months[month - 1] + " " + year);
        System.out.println("Su Mo Tu We Th Fr Sa");

        int startDay = getFirstDayOfMonth(year, month);
        for (int i = 0; i < startDay; i++) System.out.print("   ");
        for (int i = 1; i <= days; i++) {
            System.out.printf("%3d", i);
            if ((i + startDay) % 7 == 0) System.out.println();
        }
        System.out.println();
    }

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

9. Euclidean Distance and Line Equation

```
public class EuclideanAndLineEquation {
    public static double euclideanDistance(int x1, int y1, int x2, int y2) {
        return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
    }

    public static double[] lineEquation(int x1, int y1, int x2, int y2) {
        double m = (double)(y2 - y1) / (x2 - x1);
        double b = y1 - m * x1;
        return new double[]{m, b};
    }

    public static void main(String[] args) {
        int x1 = 2, y1 = 3, x2 = 5, y2 = 7;
        System.out.println("Euclidean Distance: " + euclideanDistance(x1, y1, x2, y2));
        double[] eq = lineEquation(x1, y1, x2, y2);
        System.out.println("Line Equation: y = " + eq[0] + "x + " + eq[1]);
    }
}
```

10. Collinearity Check

```
public class CollinearityCheck {
    public static boolean checkSlope(int x1, int y1, int x2, int y2, int x3, int y3) {
        return (y2 - y1) * (x3 - x2) == (y3 - y2) * (x2 - x1);
    }

    public static boolean checkArea(int x1, int y1, int x2, int y2, int x3, int y3) {
        double area = 0.5 * (x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2));
        return area == 0;
    }

    public static void main(String[] args) {
        int x1 = 2, y1 = 4, x2 = 4, y2 = 6, x3 = 6, y3 = 8;
        System.out.println("Collinear by Slope? " + checkSlope(x1, y1, x2, y2, x3, y3));
        System.out.println("Collinear by Area? " + checkArea(x1, y1, x2, y2, x3, y3));
    }
}
```

Java Level 3 Methods - Programs

11. Bonus Calculator

```
public class BonusCalculator {
    public static int[][] generateSalariesAndService(int employees) {
        int[][] data = new int[employees][2];
        for (int i = 0; i < employees; i++) {
            data[i][0] = (int)(Math.random() * 90000) + 10000; // salary
            data[i][1] = (int)(Math.random() * 10); // years of service
        }
        return data;
    }

    public static double[][] calculateBonus(int[][] data) {
        double[][] updated = new double[data.length][2]; // new salary and bonus
        for (int i = 0; i < data.length; i++) {
            double bonus = data[i][1] > 5 ? 0.05 * data[i][0] : 0.02 * data[i][0];
            updated[i][0] = data[i][0] + bonus;
            updated[i][1] = bonus;
        }
        return updated;
    }

    public static void displaySummary(int[][] original, double[][] updated) {
        double totalOld = 0, totalNew = 0, totalBonus = 0;
        System.out.println("Emp\tOld Salary\tYears\tNew Salary\tBonus");
        for (int i = 0; i < original.length; i++) {
                    System.out.printf("%d\t%d\t\t%d\t%.2f\t%.2f\n",  i+1,  original[i][0],
original[i][1], updated[i][0], updated[i][1]);
            totalOld += original[i][0];
            totalNew += updated[i][0];
            totalBonus += updated[i][1];
        }
           System.out.printf("\nTotal  Old  Salary:  %.2f,  Total  New  Salary:  %.2f,  Total
Bonus: %.2f\n", totalOld, totalNew, totalBonus);
    }

    public static void main(String[] args) {
        int[][] data = generateSalariesAndService(10);
        double[][] updated = calculateBonus(data);
        displaySummary(data, updated);
    }
}
```

Java Level 3 Methods - Programs

12. Student Scorecard

```
public class StudentScorecard {
    public static int[][] generateScores(int students) {
        int[][] scores = new int[students][3]; // PCM
        for (int i = 0; i < students; i++) {
            for (int j = 0; j < 3; j++) {
                scores[i][j] = (int)(Math.random() * 50) + 50; // 2-digit scores
            }
        }
        return scores;
    }

    public static double[][] calculateResults(int[][] scores) {
        double[][] results = new double[scores.length][3]; // total, average, percentage
        for (int i = 0; i < scores.length; i++) {
            int total = scores[i][0] + scores[i][1] + scores[i][2];
            double avg = total / 3.0;
            double percent = (total / 300.0) * 100;
            results[i][0] = total;
            results[i][1] = Math.round(avg * 100) / 100.0;
            results[i][2] = Math.round(percent * 100) / 100.0;
        }
        return results;
    }

    public static void displayScorecard(int[][] scores, double[][] results) {
        System.out.println("Stu\tPhysics\tChemistry\tMath\tTotal\tAvg\tPercent");
        for (int i = 0; i < scores.length; i++) {
                        System.out.printf("%d\t%d\t\t%d\t\t%d\t%.0f\t%.2f\t%.2f\n",  i+1,
scores[i][0], scores[i][1], scores[i][2], results[i][0], results[i][1], results[i][2]);
        }
    }

    public static void main(String[] args) {
        int[][] scores = generateScores(5);
        double[][] results = calculateResults(scores);
        displayScorecard(scores, results);
    }
}
```

13. Matrix Operations

```
public class MatrixOperations {
    public static int[][] createMatrix(int rows, int cols) {
        int[][] matrix = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = (int)(Math.random() * 10);
            }
        }
        return matrix;
    }

    public static int[][] add(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[][] subtract(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[][] multiply(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[][] transpose(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 void printMatrix(int[][] matrix) {
```

```
        for (int[] row : matrix) {
            for (int val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] A = createMatrix(2, 2);
        int[][] B = createMatrix(2, 2);

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

        System.out.println("Addition:");
        printMatrix(add(A, B));
        System.out.println("Subtraction:");
        printMatrix(subtract(A, B));
        System.out.println("Multiplication:");
        printMatrix(multiply(A, B));
        System.out.println("Transpose A:");
        printMatrix(transpose(A));
    }
}
```