LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN 4
CLASS AND METHOD

Disusun oleh:

Nama	: Aditya Lucky Zulkarnaen
NIM	: 24/537764/SV/24449
Kelas	: PLB1
Dosen Pengampu	: Margareta Hardiyanti, S.Kom., M.Eng.

PROGRAM STUDI D-IV TEKNOLOGI REKAYASA PERANGKAT LUNAK
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025

PERTEMUAN 4
CLASS AND METHOD

A. Soal

1. Buatlah sebuah program untuk membuat class pegawai, yang memiliki atribut nama, umur, dan gaji. Selain itu, terdapat juga sebuah atribut static yang digunakan untuk penanda seberapa banyak objek pegawai yang telah dibuat. Outputkan methode setiap objek sehingga output menjadi “Pegawai {nama} berumur {umur} memiliki gaji sebesar {gaji}”.
2. Buatlah sebuah scoreManager, dimana kita bisa menambahkan nilai dari setiap anak, kemudian buatlah juga method untuk menjumlahkan total semua nilai, mencari rata-rata nilai, nilai tertinggi dan terendah, serta perbedaan nilai tertinggi dan terendah.
3. Buatlah sebuah program menggunakan enum untuk setiap mata uang Indonesia kemudian simpan menggunakan map. Outputkan banyak uang pada setiap nominal kertas, kemudian outputkan juga jumlah total nominal uang.

B. Kode

1. Pegawai

```
public class Pegawai {
    private String nama;
    private int umur;
    private double gaji;

    private static int jumlahPegawai = 0;

    public Pegawai(String nama, int umur, double gaji) {
        this.nama = nama;
        this.umur = umur;
        this.gaji = gaji;
        jumlahPegawai++;
    }
```

```
    public void tampilkanInfo() {
        System.out.println("Pegawai " + nama + " berumur " + umur + " memiliki gaji sebesar " + gaji);
    }

    public static int getJumlahPegawai() {
        return jumlahPegawai;
    }

    public static void main(String[] args) {
        Pegawai pegawai1 = new Pegawai("Budi", 30, 5000000);
        Pegawai pegawai2 = new Pegawai("Ani", 28, 5500000);
        Pegawai pegawai3 = new Pegawai("Dodi", 35, 6000000);

        pegawai1.tampilkanInfo();
        pegawai2.tampilkanInfo();
        pegawai3.tampilkanInfo();

        System.out.println("Jumlah pegawai yang telah dibuat: " + Pegawai.getJumlahPegawai());
    }
}
```

2. scoreManager

```
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class ScoreManager {
    private Map<String, Integer> scores;
    private Scanner scanner;
```

```
public ScoreManager() {
    scores = new HashMap<>();
    scanner = new Scanner(System.in);
}

public void tambahNilaiDariInput() {
    System.out.print("Masukkan nama anak: ");
    String nama = scanner.nextLine();

    int nilai = 0;
    boolean inputValid = false;

    while (!inputValid) {
        try {
            System.out.print("Masukkan nilai " + nama + ": ");
            nilai = Integer.parseInt(scanner.nextLine());
            inputValid = true;
        } catch (NumberFormatException e) {
            System.out.println("Input tidak valid! Nilai harus berupa angka.");
        }
    }

    scores.put(nama, nilai);
    System.out.println("Nilai " + nama + " berhasil ditambahkan: " + nilai);
}

public void tambahNilai(String nama, int nilai) {
    scores.put(nama, nilai);
    System.out.println("Nilai " + nama + " berhasil ditambahkan: " + nilai);
}
```

```
public int getNilai(String nama) {
    if (scores.containsKey(nama)) {
        return scores.get(nama);
    } else {
        System.out.println("Nama tidak ditemukan!");
        return -1;
    }
}

public int hitungTotalNilai() {
    int total = 0;
    for (int nilai : scores.values()) {
        total += nilai;
    }
    return total;
}

public double hitungRataRata() {
    if (scores.isEmpty()) {
        return 0;
    }
    return (double) hitungTotalNilai() / scores.size();
}

public int nilaiTertinggi() {
    if (scores.isEmpty()) {
        return 0;
    }
    return Collections.max(scores.values());
}
```

```
public String anakDenganNilaiTertinggi() {
    if (scores.isEmpty()) {
        return "Tidak ada data";
    }

    int maxNilai = nilaiTertinggi();
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        if (entry.getValue() == maxNilai) {
            return entry.getKey();
        }
    }
    return "Tidak ditemukan";
}

public int nilaiTerendah() {
    if (scores.isEmpty()) {
        return 0;
    }
    return Collections.min(scores.values());
}

public String anakDenganNilaiTerendah() {
    if (scores.isEmpty()) {
        return "Tidak ada data";
    }

    int minNilai = nilaiTerendah();
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        if (entry.getValue() == minNilai) {
            return entry.getKey();
```

```
        }
    }
    return "Tidak ditemukan";
}

public int selisihNilai() {
    if (scores.isEmpty()) {
        return 0;
    }
    return nilaiTertinggi() - nilaiTerendah();
}

public void tampilkanSemuaNilai() {
    if (scores.isEmpty()) {
        System.out.println("Belum ada data nilai yang tersimpan.");
        return;
    }

    System.out.println("=== Data Nilai Anak ===");
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        System.out.println(entry.getKey() + ": " + entry.getValue());
    }
}

public void tampilkanStatistik() {
    if (scores.isEmpty()) {
        System.out.println("Belum ada data nilai yang tersimpan.");
        return;
    }

    System.out.println("=== Statistik Nilai ===");
```

```
        System.out.println("Jumlah anak: " + scores.size());
        System.out.println("Total nilai: " + hitungTotalNilai());
        System.out.println("Rata-rata nilai: " + String.format("%.2f", hitungRataRata()));
        System.out.println("Nilai tertinggi: " + nilaiTertinggi() + " (Dimiliki oleh: " + anakDenganNilaiTertinggi() + ")");
        System.out.println("Nilai terendah: " + nilaiTerendah() + " (Dimiliki oleh: " + anakDenganNilaiTerendah() + ")");
        System.out.println("Selisih nilai tertinggi dan terendah: " + selisihNilai());
    }

    public void jalankanProgram() {
        boolean lanjut = true;

        while (lanjut) {
            System.out.println("\n=== Menu ScoreManager ===");
            System.out.println("1. Tambah Nilai Anak");
            System.out.println("2. Lihat Semua Nilai");
            System.out.println("3. Lihat Statistik Nilai");
            System.out.println("4. Cari Nilai Anak");
            System.out.println("0. Keluar");
            System.out.print("Pilihan Anda: ");

            int pilihan = -1;
            try {
                pilihan = Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("Input tidak valid! Harap masukkan angka.");
                continue;
            }

            switch (pilihan) {
```

```
            case 1:
                tambahNilaiDariInput();
                break;
            case 2:
                tampilkanSemuaNilai();
                break;
            case 3:
                tampilkanStatistik();
                break;
            case 4:
                System.out.print("Masukkan nama anak: ");
                String nama = scanner.nextLine();
                int nilai = getNilai(nama);
                if (nilai != -1) {
                    System.out.println("Nilai " + nama + ": " + nilai);
                }
                break;
            case 0:
                lanjut = false;
                System.out.println("Terima kasih telah menggunakan ScoreManager!");
                break;
            default:
                System.out.println("Pilihan tidak valid!");
                break;
        }
    }

    scanner.close();
}

public static void main(String[] args) {
```

```
        System.out.println("Selamat datang di ScoreManager!");
        ScoreManager manager = new ScoreManager();
        manager.jalankanProgram();
    }
}
```

3. uangIndonesia

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class UangIndonesia {
    public enum MataUang {
        SERIBU(1000),
        DUARIBU(2000),
        LIMARIBU(5000),
        SEPULUHRIBU(10000),
        DUAPULUHRIBU(20000),
        LIMAPULUHRIBU(50000),
        SERATUSRIBU(100000);

        private final int nilai;

        MataUang(int nilai) {
            this.nilai = nilai;
        }

        public int getNilai() {
            return nilai;
        }

        @Override
```

```
    public String toString() {
        return "Rp " + nilai;
    }
}

public static class KasUang {
    private Map<MataUang, Integer> uangMap;
    private Scanner scanner;

    public KasUang() {
        uangMap = new HashMap<>();
        scanner = new Scanner(System.in);

        for (MataUang uang : MataUang.values()) {
            uangMap.put(uang, 0);
        }
    }

    public void tambahUang(MataUang uang, int jumlah) {
        if (jumlah < 0) {
            System.out.println("Jumlah tidak valid!");
            return;
        }

        int jumlahSekarang = uangMap.get(uang);
        uangMap.put(uang, jumlahSekarang + jumlah);
        System.out.println("Berhasil menambahkan " + jumlah + " lembar " + uang);
    }

    public void inputJumlahUang() {
        for (MataUang uang : MataUang.values()) {
```

```
        boolean inputValid = false;

        while (!inputValid) {
            try {
                System.out.print("Masukkan jumlah lembar " + uang + ": ");
                int jumlah = Integer.parseInt(scanner.nextLine());

                if (jumlah < 0) {
                    System.out.println("Jumlah tidak boleh negatif!");
                    continue;
                }

                tambahUang(uang, jumlah);
                inputValid = true;
            } catch (NumberFormatException e) {
                System.out.println("Input tidak valid! Harap masukkan angka.");
            }
        }
    }
}

public void tampilkanJumlahUang() {
    System.out.println("\n=== Jumlah Uang Per Nominal ===");

    for (MataUang uang : MataUang.values()) {
        int jumlah = uangMap.get(uang);
        System.out.println(uang + ": " + jumlah + " lembar");
    }
}

public long hitungTotalNilai() {
```

```
        long total = 0;

        for (MataUang uang : MataUang.values()) {
            int jumlah = uangMap.get(uang);
            total += (long) uang.getNilai() * jumlah;
        }

        return total;
    }

    public void tampilkanTotalNilai() {
        long total = hitungTotalNilai();
        System.out.println("\n=== Total Nilai Uang ===");
        System.out.println("Total: Rp " + total);
    }

    public void tampilkanDetailNilai() {
        System.out.println("\n=== Detail Nilai Per Nominal ===");

        for (MataUang uang : MataUang.values()) {
            int jumlah = uangMap.get(uang);
            long nilaiTotal = (long) uang.getNilai() * jumlah;
            System.out.println(uang + " x " + jumlah + " lembar = Rp " + nilaiTotal);
        }
    }

    public void tutup() {
        scanner.close();
    }
}
```

```
    public static void main(String[] args) {
        System.out.println("=== Program Perhitungan Uang Indonesia ===");

        KasUang kas = new KasUang();
        kas.inputJumlahUang();
        kas.tampilkanJumlahUang();
        kas.tampilkanDetailNilai();
        kas.tampilkanTotalNilai();
        kas.tutup();
    }
}
```

C. Penjelasan

1. Pegawai

   a. Data awal bawaan Pegawai

   Data pegawai disimpan dalam bentuk objek dari kelas Pegawai. Setiap pegawai memiliki atribut nama, umur, dan gaji yang diinisialisasi saat objek dibuat. Selain itu, terdapat atribut statik jumlahPegawai yang digunakan untuk menghitung jumlah pegawai yang telah dibuat.

```
private String nama;
private int umur;
private double gaji;
private static int jumlahPegawai = 0;
```

   - nama menyimpan nama pegawai.
   - umur menyimpan umur pegawai.
   - gaji menyimpan gaji pegawai.
   - jumlahPegawai adalah variabel statik yang menyimpan jumlah pegawai yang telah dibuat.

   b. Fungsi menambahkan data pegawai baru

   Fungsi ini digunakan untuk membuat objek pegawai baru dengan parameter nama, umur, dan gaji.

```
public Pegawai(String nama, int umur, double gaji) {
    this.nama = nama;
    this.umur = umur;
    this.gaji = gaji;
    jumlahPegawai++;
}
```

- this.nama = nama; → Menginisialisasi nama pegawai.
- this.umur = umur; → Menginisialisasi umur pegawai.
- this.gaji = gaji; → Menginisialisasi gaji pegawai.
- jumlahPegawai++; → Menambah jumlah pegawai setiap kali objek baru dibuat.

c. Fungsi menampilkan seluruh data pegawai

Fungsi ini menampilkan informasi pegawai yang telah dibuat.

```
public void tampilkanInfo() {
    System.out.println("Pegawai " + nama + " berumur " + umur + " memiliki gaji sebesar " + gaji);
}
```

- Fungsi ini mencetak nama, umur, dan gaji pegawai ke layar.

d. Fungsi untuk mendapatkan jumlah pegawai yang telah dibuat

Fungsi ini adalah fungsi statik yang mengembalikan jumlah pegawai yang telah dibuat.

```
public static int getJumlahPegawai() {
    return jumlahPegawai;
}
```

- Karena jumlahPegawai adalah variabel statik, fungsi ini juga harus statik agar bisa mengaksesnya.

e. Main menu - Eksekusi program

Di dalam method main, program membuat tiga objek Pegawai, menampilkan data mereka, dan mencetak jumlah pegawai yang telah dibuat.

```
public static void main(String[] args) {
    Pegawai pegawai1 = new Pegawai("Budi", 30, 5000000);
    Pegawai pegawai2 = new Pegawai("Ani", 28, 5500000);
```

```
Pegawai pegawai3 = new Pegawai("Dodi", 35, 6000000);
```

- Membuat tiga objek pegawai dengan data masing-masing.
- Memanggil fungsi tampilkanInfo() untuk setiap pegawai agar informasi mereka ditampilkan.
- Menampilkan jumlah total pegawai yang telah dibuat menggunakan method statik getJumlahPegawai().

f. Output Program

Ketika program dijalankan, output yang dihasilkan adalah:

```
Pegawai Budi berumur 30 memiliki gaji sebesar 5000000.0
Pegawai Ani berumur 28 memiliki gaji sebesar 5500000.0
Pegawai Dodi berumur 35 memiliki gaji sebesar 6000000.0
Jumlah pegawai yang telah dibuat: 3
```

- Informasi setiap pegawai ditampilkan.
- Jumlah total pegawai yang dibuat juga ditampilkan.

2. scoreManager

a. Data awal bawaan

Data nilai anak disimpan dalam HashMap dengan:

- Key → Nama anak (String)
- Value → Nilai anak (Integer)

```
private Map<String, Integer> scores;
private Scanner scanner;
```

- scores → Struktur data HashMap yang digunakan untuk menyimpan nama dan nilai.
- scanner → Digunakan untuk menerima input dari pengguna.

```
public ScoreManager() {
    scores = new HashMap<>();
    scanner = new Scanner(System.in);
}
```

b. Fungsi menambahkan nilai anak

Fungsi ini digunakan untuk menambahkan data nilai anak dengan validasi input.

```
public void tambahNilaiDariInput() {
```

```
    System.out.print("Masukkan nama anak: ");
    String nama = scanner.nextLine();

    int nilai = 0;
    boolean inputValid = false;

    while (!inputValid) {
        try {
            System.out.print("Masukkan nilai " + nama + ": ");
            nilai = Integer.parseInt(scanner.nextLine());
            inputValid = true;
        } catch (NumberFormatException e) {
            System.out.println("Input tidak valid! Nilai harus berupa angka.");
        }
    }

    scores.put(nama, nilai);
    System.out.println("Nilai " + nama + " berhasil ditambahkan: " + nilai);
}
```

- Meminta input nama anak.
- Melakukan validasi input nilai agar hanya menerima angka (Integer.parseInt()).
- Menyimpan nilai dalam HashMap dengan scores.put(nama, nilai).

c. Fungsi mencari nilai anak berdasarkan nama

```
public int getNilai(String nama) {
    if (scores.containsKey(nama)) {
        return scores.get(nama);
    } else {
        System.out.println("Nama tidak ditemukan!");
        return -1;
    }
}
```

- Mengecek apakah nama ada dalam scores.
- Jika ditemukan, mengembalikan nilai.
- Jika tidak, menampilkan pesan "Nama tidak ditemukan!" dan mengembalikan -1.

d. Fungsi menghitung total dan rata-rata nilai

```
public int hitungTotalNilai() {
    int total = 0;
    for (int nilai : scores.values()) {
        total += nilai;
    }
    return total;
}
```

- Menjumlahkan semua nilai anak dalam scores.

```
public int hitungTotalNilai() {
    int total = 0;
    for (int nilai : scores.values()) {
        total += nilai;
    }
    return total;
}
```

- Menghitung rata-rata nilai dengan membagi total nilai dengan jumlah anak.
- Jika tidak ada data, mengembalikan 0.

e. Fungsi mencari nilai tertinggi dan anak dengan nilai tertinggi

```
public int nilaiTertinggi() {
    if (scores.isEmpty()) {
        return 0;
    }
    return Collections.max(scores.values());
}
```

- Mencari nilai tertinggi dalam scores menggunakan Collections.max().

```
public String anakDenganNilaiTertinggi() {
    if (scores.isEmpty()) {
```

```
        return "Tidak ada data";
    }

    int maxNilai = nilaiTertinggi();
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        if (entry.getValue() == maxNilai) {
            return entry.getKey();
        }
    }
    return "Tidak ditemukan";
}
```

- Mencari anak dengan nilai tertinggi dengan membandingkan semua nilai dalam scores.

f. Fungsi mencari nilai terendah dan anak dengan nilai terendah

```
public int nilaiTerendah() {
    if (scores.isEmpty()) {
        return 0;
    }
    return Collections.min(scores.values());
}
```

- Menggunakan Collections.min() untuk mendapatkan nilai terendah dalam scores.

```
public String anakDenganNilaiTerendah() {
    if (scores.isEmpty()) {
        return "Tidak ada data";
    }

    int minNilai = nilaiTerendah();
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        if (entry.getValue() == minNilai) {
            return entry.getKey();
        }
```

```
    }
    return "Tidak ditemukan";
}
```

- Mencari anak dengan nilai terendah dengan membandingkan semua nilai dalam scores.

g. Fungsi menampilkan semua nilai

```
public void tampilkanSemuaNilai() {
    if (scores.isEmpty()) {
        System.out.println("Belum ada data nilai yang tersimpan.");
        return;
    }

    System.out.println("=== Data Nilai Anak ===");
    for (Map.Entry<String, Integer> entry : scores.entrySet()) {
        System.out.println(entry.getKey() + ": " + entry.getValue());
    }
}
```

- Menampilkan semua nama dan nilai anak dalam scores.

h. Fungsi untuk menampilkan statistik nilai

```
public void tampilkanStatistik() {
    if (scores.isEmpty()) {
        System.out.println("Belum ada data nilai yang tersimpan.");
        return;
    }

    System.out.println("=== Statistik Nilai ===");
    System.out.println("Jumlah anak: " + scores.size());
    System.out.println("Total nilai: " + hitungTotalNilai());
    System.out.println("Rata-rata nilai: " + String.format("%.2f", hitungRataRata()));
    System.out.println("Nilai tertinggi: " + nilaiTertinggi() + " (Dimiliki oleh: " +
anakDenganNilaiTertinggi() + ")");
```

```
    System.out.println("Nilai terendah: " + nilaiTerendah() + " (Dimiliki oleh: " +
anakDenganNilaiTerendah() + ")");
    System.out.println("Selisih nilai tertinggi dan terendah: " + selisihNilai());
}
```

- Menampilkan jumlah anak, total nilai, rata-rata, nilai tertinggi & terendah, serta selisih nilai.

i. Main menu - Interaksi dengan pengguna

```
public void jalankanProgram() {
    boolean lanjut = true;

    while (lanjut) {
        System.out.println("\n=== Menu ScoreManager ===");
        System.out.println("1. Tambah Nilai Anak");
        System.out.println("2. Lihat Semua Nilai");
        System.out.println("3. Lihat Statistik Nilai");
        System.out.println("4. Cari Nilai Anak");
        System.out.println("0. Keluar");
        System.out.print("Pilihan Anda: ");

        int pilihan = -1;
        try {
            pilihan = Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Input tidak valid! Harap masukkan angka.");
            continue;
        }

        switch (pilihan) {
            case 1:
                tambahNilaiDariInput();
                break;
```

```
            case 2:
                tampilkanSemuaNilai();
                break;
            case 3:
                tampilkanStatistik();
                break;
            case 4:
                System.out.print("Masukkan nama anak: ");
                String nama = scanner.nextLine();
                int nilai = getNilai(nama);
                if (nilai != -1) {
                    System.out.println("Nilai " + nama + ": " + nilai);
                }
                break;
            case 0:
                lanjut = false;
                System.out.println("Terima kasih telah menggunakan ScoreManager!");
                break;
            default:
                System.out.println("Pilihan tidak valid!");
                break;
        }
    }

    scanner.close();
}
```

- Loop utama program yang menerima input pengguna dan menjalankan fitur sesuai pilihan.

3. uangIndonesia
    a. Struktur Data Uang Indonesia

Program ini menggunakan Enum untuk mendefinisikan berbagai pecahan uang rupiah:

```
public enum MataUang {
    SERIBU(1000),
    DUARIBU(2000),
    LIMARIBU(5000),
    SEPULUHRIBU(10000),
    DUAPULUHRIBU(20000),
    LIMAPULUHRIBU(50000),
    SERATUSRIBU(100000);
```

- Setiap nilai uang direpresentasikan sebagai konstanta dengan nilai masing-masing.

```
private final int nilai;
```

- Setiap pecahan uang memiliki atribut nilai yang merepresentasikan nominalnya.

```
MataUang(int nilai) {
    this.nilai = nilai;
}
```

- Konstruktor untuk menginisialisasi nominal setiap pecahan uang.

```
public int getNilai() {
    return nilai;
}
```

- Fungsi getter untuk mengambil nilai nominal uang.

```
public String toString() {
    return "Rp " + nilai;
}
```

- Override method toString() agar setiap pecahan ditampilkan dalam format "Rp nominal".

b. Kelas KasUang untuk Mengelola Uang

Kelas KasUang digunakan untuk menyimpan dan menghitung jumlah lembar uang berdasarkan nominalnya.

```
private Map<MataUang, Integer> uangMap;
private Scanner scanner;
```

- uangMap → HashMap yang menyimpan jumlah lembar untuk setiap nominal uang.
- scanner → Scanner untuk menerima input dari pengguna.

```
public KasUang() {
    uangMap = new HashMap<>();
    scanner = new Scanner(System.in);

    for (MataUang uang : MataUang.values()) {
        uangMap.put(uang, 0);
    }
}
```

- Konstruktor menginisialisasi uangMap dengan semua pecahan uang dan mengatur jumlah awalnya 0.

c. Fungsi menambah uang

```
public void tambahUang(MataUang uang, int jumlah) {
    if (jumlah < 0) {
        System.out.println("Jumlah tidak valid!");
        return;
    }

    int jumlahSekarang = uangMap.get(uang);
    uangMap.put(uang, jumlahSekarang + jumlah);
    System.out.println("Berhasil menambahkan " + jumlah + " lembar " + uang);
}
```

- Mengecek apakah jumlah valid (tidak negatif).
- Menambahkan jumlah lembar uang ke dalam uangMap.

d. Fungsi input jumlah uang

```
public void inputJumlahUang() {
    for (MataUang uang : MataUang.values()) {
```

```
        boolean inputValid = false;

        while (!inputValid) {
            try {
                System.out.print("Masukkan jumlah lembar " + uang + ": ");
                int jumlah = Integer.parseInt(scanner.nextLine());

                if (jumlah < 0) {
                    System.out.println("Jumlah tidak boleh negatif!");
                    continue;
                }

                tambahUang(uang, jumlah);
                inputValid = true;
            } catch (NumberFormatException e) {
                System.out.println("Input tidak valid! Harap masukkan angka.");
            }
        }
    }
}
```

- Meminta input jumlah lembar untuk setiap nominal uang.
- Validasi input → Hanya menerima angka dan tidak boleh negatif.
- Memanggil tambahUang() untuk menyimpan jumlah uang ke dalam uangMap.

e. Fungsi menampilkan data uang

```
public void tampilkanJumlahUang() {
    System.out.println("\n=== Jumlah Uang Per Nominal ===");

    for (MataUang uang : MataUang.values()) {
        int jumlah = uangMap.get(uang);
        System.out.println(uang + ": " + jumlah + " lembar");
    }
```

```
}
```

- Menampilkan jumlah lembar uang untuk setiap nominal yang ada dalam uangMap.

f. Fungsi menghitung total nilai uang

```
public long hitungTotalNilai() {
    long total = 0;

    for (MataUang uang : MataUang.values()) {
        int jumlah = uangMap.get(uang);
        total += (long) uang.getNilai() * jumlah;
    }

    return total;
}
```

- Menghitung total nilai uang dengan mengalikan jumlah lembar dengan nominal uang masing-masing.

g. Fungsi Menampilkan Total Nilai Uang

```
public void tampilkanTotalNilai() {
    long total = hitungTotalNilai();
    System.out.println("\n=== Total Nilai Uang ===");
    System.out.println("Total: Rp " + total);
}
```

- Menampilkan total keseluruhan uang yang telah dimasukkan.

h. Fungsi Menampilkan Detail Nilai Per Nominal

```
public void tampilkanDetailNilai() {
    System.out.println("\n=== Detail Nilai Per Nominal ===");

    for (MataUang uang : MataUang.values()) {
        int jumlah = uangMap.get(uang);
        long nilaiTotal = (long) uang.getNilai() * jumlah;
        System.out.println(uang + " x " + jumlah + " lembar = Rp " + nilaiTotal);
```

```
    }
}
```

- Menampilkan rincian setiap nominal uang beserta total nilainya.

i. Eksekusi Program (Main Method)

```
public static void main(String[] args) {
    System.out.println("=== Program Perhitungan Uang Indonesia ===");

    KasUang kas = new KasUang();
    kas.inputJumlahUang();
    kas.tampilkanJumlahUang();
    kas.tampilkanDetailNilai();
    kas.tampilkanTotalNilai();
    kas.tutup();
}
```

- Membuat objek KasUang.
- Meminta input jumlah lembar uang dari pengguna.
- Menampilkan jumlah uang per nominal.
- Menampilkan rincian nilai per nominal.
- Menampilkan total keseluruhan uang.
- Menutup scanner setelah selesai.