Nama : Arditya Adjie Rosandi

NIM : 20230801074

Pemrogaman Berorinetasi Objek

Soal No 1

```java
class Dosen {
    private String nik;
    private String nama;

    public Dosen(String nik, String nama) {
        this.nik = nik;
        this.nama = nama;
    }

    public String getNik() {
        return nik;
    }

    public String getNama() {
        return nama;
    }

    public void view() {
        System.out.println("NIK : " + nik);
        System.out.println("NAMA : " + nama);
    }
}
```

```java
class Rektor extends Dosen {
    private int tahunMasuk;

    public Rektor(String nik, String nama, int tahunMasuk) {
        super(nik, nama);
        this.tahunMasuk = tahunMasuk;
    }

    public int getTahunMasuk() {
        return tahunMasuk;
    }

    public void viewRektor() {
        view();
        System.out.println("TAHUN MASUK : " + tahunMasuk);
    }
}
```

```java
class Dekan extends Dosen {
    private String fakultas;

    public Dekan(String nik, String nama, String fakultas) {
        super(nik, nama);
        this.fakultas = fakultas;
    }

    public String getFakultas() {
        return fakultas;
    }

    public void viewDekan() {
        view();
        System.out.println("FAKULTAS : " + fakultas);
    }
}
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.print("\033[H\033[2J");
        // Object Dosen
        Dosen dosen = new Dosen(nik:"101", nama:"ARDITYA ADJIE ROSANDI");
        System.out.println("==|| DATA DOSEN ||==");
        dosen.view();
        System.out.println();

        // Object Rektor
        Rektor rektor = new Rektor(nik:"102", nama:"DHCIKY HARYADI SUPRIYONO", tahunMasuk:2022);
        System.out.println("==|| DATA REKTOR ||==");
        rektor.viewRektor();
        System.out.println();

        // Object Dekan
        Dekan dekan = new Dekan(nik:"103", nama:"AVRIAN RIZKY MAULANA", fakultas:"ILMU KOMPUTER");
        System.out.println("==|| DATA DEKAN ||==");
        dekan.viewDekan();
    }
}
```

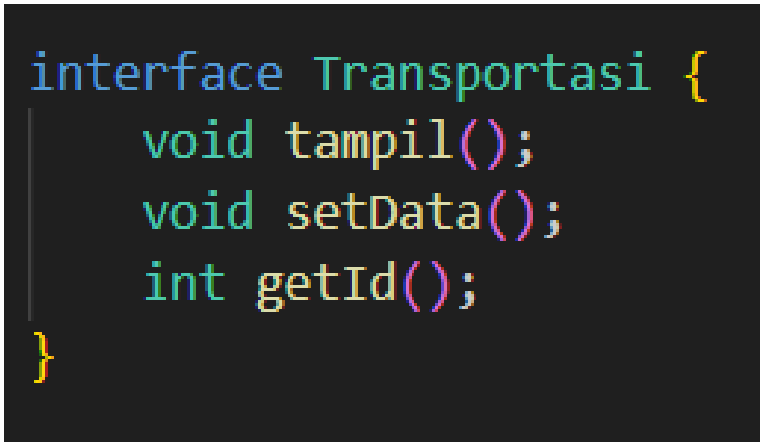
Nomor 1: Program Inheritance

Kode ini menggunakan konsep **inheritance** (pewarisan) dalam pemrograman berorientasi objek (OOP) untuk mewariskan properti dan metode dari satu kelas ke kelas lainnya:

1. **Class Dosen**:
    - Merupakan **superclass** yang memiliki atribut nik dan nama.
    - Constructor digunakan untuk menginisialisasi nik dan nama.
    - Metode getNik() dan getNama() mengembalikan nilai atribut.
    - Metode view() digunakan untuk menampilkan informasi Dosen.
2. **Class Rektor**:
    - Kelas ini mewarisi Dosen menggunakan **extends**.
    - Menambahkan atribut baru tahunMasuk.
    - Menambahkan metode getTahunMasuk() dan viewRektor() untuk menampilkan data tambahan.

3. **Class Dekan**:
    - Juga mewarisi Dosen dan menambahkan atribut fakultas.
    - Constructor menginisialisasi atribut tambahan fakultas.
    - Metode getFakultas() dan viewDekan() untuk menampilkan informasi Dekan.

Soal No 2



```
interface Transportasi {
    void tampil();
    void setData();
    int getId();
}
```

```
class Gojek implements Transportasi {
    protected int harga;
    protected int id;

    public Gojek() {
        this.harga = 0;
        this.id = 0;
    }

    @Override
    public void tampil() {
        System.out.println("GOJEK - HARGA : " + harga + ", ID: " + id);
    }

    @Override
    public void setData() {
        this.harga = 20000; // Contoh data
        this.id = 123;
    }

    @Override
    public int getId() {
        return id;
    }
}
```

```java
class Bayar extends Gojek {
    private int jarak;
    private int total;
    private String nama;

    public Bayar() {
        super();
        this.jarak = 10; // Contoh jarak
        this.nama = "PENGGUNA GOJEK";
        this.total = jarak * 2000; // Perhitungan total biaya
    }

    public Bayar(int id) {
        this();
        this.id = id;
        System.out.println("BAYAR DENGAN ID : " + this.id);
    }

    @Override
    public void tampil() {
        System.out.println("NAMA : " + nama);
        System.out.println("JARAK : " + jarak + " KM");
        System.out.println("TOTAL BAYAR : RP." + total);
        System.out.println("ID GOJEK : " + id);
    }

    @Override
    public void setData() {
        super.setData();
        this.total = jarak * 2000;
        System.out.println("DATA BAYAR SUDAH DISET.");
    }

    @Override
    public int getId() {
        return id;
    }
}
```

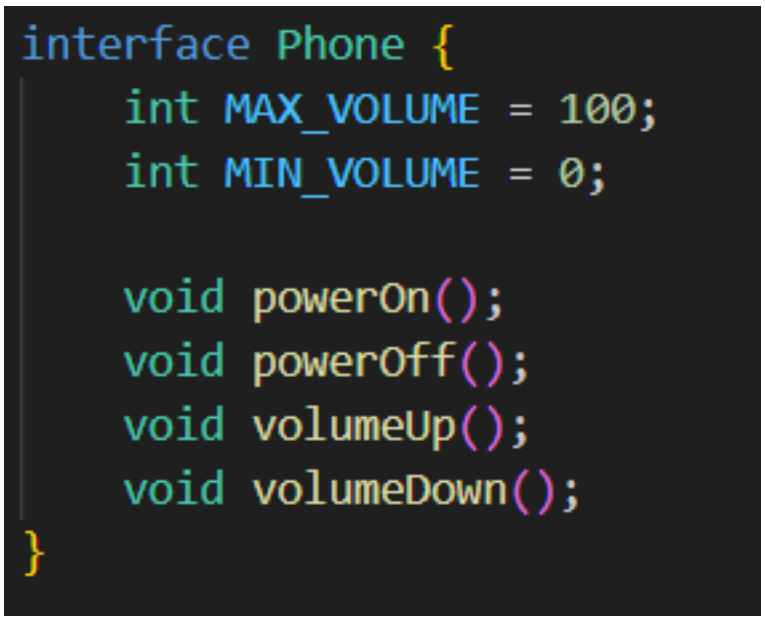
```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.print("\033[H\033[2J");

        System.out.println("==|| GOJEK ||==");
        Gojek gojek = new Gojek();
        gojek.setData();
        gojek.tampil();

        System.out.println("\n==|| BAYAR ||==");
        Bayar bayar = new Bayar(id:456);
        bayar.setData();
        bayar.tampil();
    }
}
```

Nomor 2: Program Inheritance dan Interface

Kode ini menggabungkan konsep **inheritance** dan **interface** untuk memperluas fungsionalitas:

1. **Interface Transportasi**:
    - Merupakan kontrak yang berisi metode abstrak seperti tampil(), setData(), dan getId().
    - Semua kelas yang mengimplementasi interface ini wajib mendefinisikan metode tersebut.
2. **Class Gojek**:
    - Merupakan **superclass** yang mengimplementasi Transportasi.
    - Menyediakan metode setData() untuk menginisialisasi atribut harga dan id.
    - Metode tampil() digunakan untuk menampilkan data harga dan id.
    - Metode getId() mengembalikan nilai id.
3. **Subclass Bayar**:
    - Mewarisi Gojek dan menambahkan atribut jarak, total, dan nama.
    - Constructor menghitung total biaya berdasarkan jarak.
    - Metode setData() dan tampil() diperbarui untuk menampilkan informasi pembayaran.

Soal No 3



```java
class Xiaomi implements Phone {
    private int volume;
    private boolean isPowerOn;

    public Xiaomi() {
        this.volume = 50; // Default volume
        this.isPowerOn = false;
    }

    @Override
    public void powerOn() {
        isPowerOn = true;
        System.out.println("XIAOMI IS ON.");
    }

    @Override
    public void powerOff() {
        isPowerOn = false;
        System.out.println("XIAOMI IS OFF.");
    }

    @Override
    public void volumeUp() {
        if (isPowerOn) {
            if (volume < MAX_VOLUME) {
                volume++;
                System.out.println("XIAOMI VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MAX.");
            }
        }
    }

    @Override
    public void volumeDown() {
        if (isPowerOn) {
            if (volume > MIN_VOLUME) {
                volume--;
                System.out.println("XIAOMI VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MIN.");
            }
        }
    }
}
```

```java
class iPhone implements Phone {
    private int volume;
    private boolean isPowerOn;

    public iPhone() {
        this.volume = 50; // Default volume
        this.isPowerOn = false;
    }

    @Override
    public void powerOn() {
        isPowerOn = true;
        System.out.println("IPHONE IS ON.");
    }

    @Override
    public void powerOff() {
        isPowerOn = false;
        System.out.println("IPHONE IS OFF.");
    }

    @Override
    public void volumeUp() {
        if (isPowerOn) {
            if (volume < MAX_VOLUME) {
                volume++;
                System.out.println("IPHONE VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MAX.");
            }
        }
    }

    @Override
    public void volumeDown() {
        if (isPowerOn) {
            if (volume > MIN_VOLUME) {
                volume--;
                System.out.println("IPHONE VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MIN.");
            }
        }
    }
}
```

```java
class Samsung implements Phone {
    private int volume;
    private boolean isPowerOn;

    public Samsung() {
        this.volume = 50; // Default volume
        this.isPowerOn = false;
    }

    @Override
    public void powerOn() {
        isPowerOn = true;
        System.out.println("SAMSUNG IS ON.");
    }

    @Override
    public void powerOff() {
        isPowerOn = false;
        System.out.println("SAMSUNG IS OFF.");
    }

    @Override
    public void volumeUp() {
        if (isPowerOn) {
            if (volume < MAX_VOLUME) {
                volume++;
                System.out.println("SAMSUNG VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MAX.");
            }
        }
    }

    @Override
    public void volumeDown() {
        if (isPowerOn) {
            if (volume > MIN_VOLUME) {
                volume--;
                System.out.println("SAMSUNG VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT.");
            }
        }
    }
}
```

```java
class Oppo implements Phone {
    private int volume;
    private boolean isPowerOn;

    public Oppo() {
        this.volume = 50; // Default volume
        this.isPowerOn = false;
    }

    @Override
    public void powerOn() {
        isPowerOn = true;
        System.out.println("OPPO IS ON.");
    }

    @Override
    public void powerOff() {
        isPowerOn = false;
        System.out.println("OPPO IS OFF.");
    }

    @Override
    public void volumeUp() {
        if (isPowerOn) {
            if (volume < MAX_VOLUME) {
                volume++;
                System.out.println("OPPO VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MAX.");
            }
        }
    }

    @Override
    public void volumeDown() {
        if (isPowerOn) {
            if (volume > MIN_VOLUME) {
                volume--;
                System.out.println("OPPO VOLUME : " + volume);
            } else {
                System.out.println("VOLUME IS AT MIN.");
            }
        }
    }
}
```

```java
class PhoneUser {
    private Phone phone;

    public PhoneUser(Phone phone) {
        this.phone = phone;
    }

    public void turnOnThePhone() {
        phone.powerOn();
    }

    public void turnOffThePhone() {
        phone.powerOff();
    }

    public void makePhoneLouder() {
        phone.volumeUp();
    }

    public void makePhoneSilent() {
        phone.volumeDown();
    }
}
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.print("\033[H\033[2J");

        Phone[] phones = {new Xiaomi(), new iPhone(), new Samsung(), new Oppo()};
        String[] brands = {"XIAOMI", "IPHONE", "SAMSUNG", "OPPO"};

        for (int i = 0; i < phones.length; i++) {
            System.out.println("\n==|| " + brands[i] + " ||==");
            PhoneUser user = new PhoneUser(phones[i]);

            user.turnOnThePhone();
            user.makePhoneLouder();
            user.makePhoneLouder();
            user.makePhoneSilent();
            user.turnOffThePhone();
        }
    }
}
```

Nomer 3: Program Polimorfisme dan Overriding

Program ini menerapkan **konsep polimorfisme** dan **method overriding** dalam pemrograman berorientasi objek (OOP) menggunakan Java. Berikut penjelasannya:

Konsep Polimorfisme

- Polimorfisme memungkinkan sebuah objek untuk memiliki **banyak bentuk**.
- Polimorfisme terjadi ketika sebuah metode dalam **kelas induk** di-**override** oleh **kelas anak**.
- Pemanggilan metode yang sama akan menghasilkan output yang berbeda bergantung pada objek yang digunakan.

1. **Superclass**:
    - Sebuah **kelas induk** mendefinisikan metode yang nantinya akan di-override oleh kelas turunannya.
2. **Subclass**:
    - Beberapa **kelas anak** yang mewarisi kelas induk dan mengimplementasikan ulang metode (overriding).
3. **Polimorfisme**:
    - Kelas utama (main) menggunakan **referensi kelas induk** untuk memanggil metode di kelas anak.
    - Pemanggilan metode akan mengeksekusi versi metode yang **di-override** oleh kelas anak.

Soal No 4

```
interface Movement {
    void move();
}

interface Flying {
    void fly();
}
```

```java
abstract class Animal {
    private String nama;
    private String sifat;
    private int ukuran;

    public Animal() {}

    public Animal(String nama, int ukuran) {
        this.nama = nama;
        this.ukuran = ukuran;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNama() {
        return nama;
    }

    public void setUkuran(int ukuran) {
        this.ukuran = ukuran;
    }

    public int getUkuran() {
        return ukuran;
    }

    public void setSifat(String sifat) {
        this.sifat = sifat;
    }

    public String getSifat() {
        return sifat;
    }
}
```

```java
class Mamalia extends Animal implements Movement {
    private String jalan;
    private String jenisMamalia;
    private boolean bisaJalan;
    private int jumlahKaki;

    public Mamalia() {}

    public Mamalia(String nama) {
        super.setNama(nama);
    }

    public void setJalan(String jalan) {
        this.jalan = jalan;
    }

    public String getJalan() {
        return jalan;
    }

    public void setBisaJalan(boolean bisaJalan) {
        this.bisaJalan = bisaJalan;
    }

    public boolean getBisaJalan() {
        return bisaJalan;
    }

    public void setJumlahKaki(int jumlahKaki) {
        this.jumlahKaki = jumlahKaki;
    }

    public int getJumlahKaki() {
        return jumlahKaki;
    }

    public void setJenisMamalia(String jenisMamalia) {
        this.jenisMamalia = jenisMamalia;
    }

    public String getJenisMamalia() {
        return jenisMamalia;
    }

    @Override
    public void move() {
        System.out.println("MAMALIA BERJALAN DENGAN " + jumlahKaki + " KAKI DI " + jalan + ".");
    }
}
```

```java
class Aves extends Animal implements Flying {
    private String jenisAves;
    private boolean bisaTerbang;

    public Aves() {}

    public Aves(String nama, int ukuran) {
        super(nama, ukuran);
    }

    public void setBisaTerbang(boolean bisaTerbang) {
        this.bisaTerbang = bisaTerbang;
    }

    public boolean getBisaTerbang() {
        return bisaTerbang;
    }

    public void setJenisAves(String jenisAves) {
        this.jenisAves = jenisAves;
    }

    public String getJenisAves() {
        return jenisAves;
    }

    @Override
    public void fly() {
        if (bisaTerbang) {
            System.out.println("AVES INI BISA TERBANG.");
        } else {
            System.out.println("AVES INI TIDAK BISA TERBANG.");
        }
    }
}
```

```java
class Ayam extends Aves {
    private String jenisAyam;
    private boolean bisaDiadu;

    public Ayam() {}

    public Ayam(String nama, int ukuran) {
        super(nama, ukuran);
    }

    public void setJenisAyam(String jenisAyam) {
        this.jenisAyam = jenisAyam;
    }

    public String getJenisAyam() {
        return jenisAyam;
    }

    public void setBisaDiadu(boolean bisaDiadu) {
        this.bisaDiadu = bisaDiadu;
    }

    public boolean getBisaDiadu() {
        return bisaDiadu;
    }

    @Override
    public final void fly() {
        System.out.println("AYAM BIASANYA TIDAK BISA TERBANG JAUH.");
    }
}
```

```java
class Merpati extends Aves {
    public Merpati() {}

    public Merpati(String nama, int ukuran) {
        super(nama, ukuran);
    }

    @Override
    public void fly() {
        System.out.println("MERPATI TERBANG TINGGI DAN JAUH.");
    }
}
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.print("\033[H\033[2J");

        // Mamalia Object
        Mamalia mamalia = new Mamalia(nama:"SINGA");
        mamalia.setJumlahKaki(jumlahKaki:4);
        mamalia.setBisaJalan(bisaJalan:true);
        mamalia.setJalan(jalan:"PADANG RUMPUT");
        System.out.println("NAMA MAMALIA : " + mamalia.getNama());
        mamalia.move();

        // Ayam Object
        Ayam ayam = new Ayam(nama:"AYAM JAGO", ukuran:3);
        ayam.setJenisAyam(jenisAyam:"AYAM ADUAN");
        ayam.setBisaDiadu(bisaDiadu:true);
        System.out.println("\nNAMA AVES : " + ayam.getNama());
        System.out.println("JENIS AVES : " + ayam.getJenisAyam());
        ayam.fly();

        // Merpati Object
        Merpati merpati = new Merpati(nama:"MERPATI POS", ukuran:2);
        System.out.println("\nNAMA AVES : " + merpati.getNama());
        merpati.fly();
    }
}
```

Nomer 4: Program Encapsulation dengan Konstruktor

Program ini menerapkan **encapsulation** menggunakan **getter dan setter**, serta menambahkan penggunaan **konstruktor**.

Konsep Encapsulation:

- Encapsulation adalah proses **menyembunyikan data** (atribut) di dalam kelas agar tidak dapat diakses langsung dari luar kelas.
- Data diakses dan dimodifikasi melalui **getter** dan **setter**.

1. **Kelas Utama**:
    - Kelas ini berisi atribut-atribut yang dibuat dengan **akses modifier private**.
    - Atribut hanya bisa diakses melalui **getter** dan **setter**.
2. **Konstruktor**:
    - Konstruktor digunakan untuk menginisialisasi nilai atribut pada saat objek dibuat.
    - Konstruktor dapat berupa **parameterized constructor** (dengan parameter) atau **default constructor**.
3. **Keuntungan Encapsulation**:
    - **Keamanan**: Data tidak dapat diakses atau dimodifikasi secara langsung.

- **Fleksibilitas**: Mengontrol bagaimana data diakses atau diubah melalui metode getter dan setter.
- **Kemudahan Pemeliharaan**: Jika terjadi perubahan atribut, hanya metode setter dan getter yang perlu diperbarui.