TUGAS & EVALUASI

Soal Tugas & Evaluasi

Jelaskan secara singkat tentang Abstraction dan Interfaces, serta perbedaannya!

Jawaban

Abstraction adalah konsep dalam OOP yang digunakan untuk menyembunyikan detail implementasi dan hanya menampilkan fungsional penting kepada pengguna.

Interface adalah kontrak yang mendefinisikan sekumpulan metode yang harus diimplementasikan oleh kelas yang menggunakan

Perbedaan Abstract dan interface:

- Abstract menggunakan keyword abstract, sedangkan interface menggunakan keyword interface
- Abstract class dapat memiliki modifier private, protected dan public sedangkan semua method di interface bersifat public secara default
- Abstract tidak mendukung multiple inheritance langsung, tetapi interface mendukung multiple inheritance dangan implementasi

Source Code

```
Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class
```

Penjelasan

Tulis Penjelasan disini …

Output

Masukan screenshot output disini

TUGAS & EVALUASI

Soal Tugas & Evaluasi

Buatlah class abstract dengan nama Mahasiswa, kemudian tambahkan method nama(), npm(), isMhsAktif(). Kemudian buatlah method non-abstract untuk menampilkan data mahasiswa tersebut. Pada method non-abstract gunakan operator ternary. Setelahnya, buatlah 2 sub-class dari Mahasiswa tersebut dengan nama Pagi dan Malam!

Jawaban

Ketik jawaban disini …

Source Code

```
abstract class Mahasiswa {
    abstract String nama();
    abstract String NPM();
    abstract boolean isMhsAktif();

    void tampilkanMahasiswa() {
        System.out.println("Nama: " + nama());
        System.out.println("NPM: " + NPM());
        System.out.println("Status: " + (isMhsAktif() ? "Aktif"
: "Tidak Aktif"));
    }
}

class Pagi extends Mahasiswa {
    private String nama;
    private String NPM;
    private boolean aktif;


    Pagi(String nama, String NPM, boolean aktif){
        this.nama = nama;
```

```
        this.NPM = NPM;
        this.aktif = aktif;
    }

    @Override
    String nama() {
        return nama;
    }

    @Override
    String NPM() {
        return NPM;
    }

    @Override
    boolean isMhsAktif() {
        return aktif;
    }
}

class Malam extends Mahasiswa {
    private String nama;
    private String NPM;
    private boolean aktif;


    Malam(String nama, String NPM, boolean aktif){
        this.nama = nama;
        this.NPM = NPM;
        this.aktif = aktif;
    }

    @Override
    String nama() {
        return nama;
    }

    @Override
    String NPM() {
```

```
        return NPM;
    }

    @Override
    boolean isMhsAktif() {
        return aktif;
    }
}


public class Mahasiswa_07651{
    public static void main(String[] args) {
        Mahasiswa mhsPagi = new Pagi("Budi", "12345", true);
        Mahasiswa mhsMalam = new Malam("Anggi", "12345", false);



        System.out.println("Mahasiswa pagi");
        mhsPagi.tampilkanMahasiswa();

        System.out.println("\n");

        System.out.println("Mahasiswa malam");
        mhsMalam.tampilkanMahasiswa();
    }
}
```

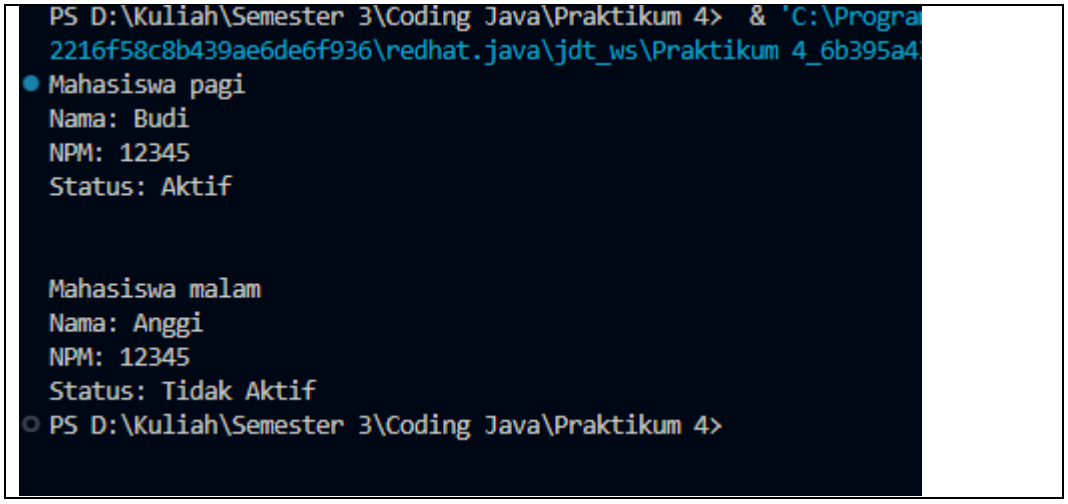
Penjelasan

Kode ini menggunakan *class abstrak* untuk mendefinisikan template data mahasiswa, termasuk nama, NPM, dan status aktif mahasiswa. Class abstrak **Mahasiswa** memiliki metode abstrak seperti nama(), NPM(), dan isMhsAktif() yang harus diimplementasikan oleh subkelas seperti **Pagi** dan **Malam**. Kedua subkelas ini mengimplementasikan perilaku spesifik sesuai data mahasiswa pagi dan malam. Metode tampilkanMahasiswa() di kelas abstrak digunakan untuk mencetak informasi mahasiswa dengan format yang seragam. Kode ini

TUGAS & EVALUASI

memanfaatkan *inheritance dan polymorphism* untuk mempermudah pengelolaan berbagai tipe mahasiswa.

Output



TUGAS & EVALUASI

Soal Tugas & Evaluasi

Buatlah class abstract dengan nama Transportasi, yang di dalamnya terdapat atribut nama dan kapasitas serta method abstract untuk menampilkan cara memesan dan juga untuk menghitung tarif transportasinya. Setelahnya, buat sub-class Taksi, Bus, dan KeretaApi yang meng-implementasikan method serta atribut dari superclass. Note: Masing-masing Sub-Class harus memiliki perilaku pemesanan dan penghitungan tarif yang berbeda sesuai jenisnya.

Jawaban

Ketik jawaban disini …

Source Code

```
abstract class Transportasi {
    String nama;
    int kapasitas;

    abstract String caraMemesan();

    abstract String hitungTarif(int jarak);

    Transportasi(String nama, int kapasitas) {
        this.nama = nama;
        this.kapasitas = kapasitas;
    }
}

class Taksi extends Transportasi {
```

```
    double tarifPerKm;

    Taksi(String nama, int kapasitas, double tarifPerKm) {
        super(nama, kapasitas);
        this.tarifPerKm = tarifPerKm;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui aplikasi";
    }

    @Override
    String hitungTarif(int jarak) {
        return tarifPerKm * jarak + " Rupiah";
    }
}

class Bus extends Transportasi {
    double tarifPerPenumpang;

    Bus(String nama, int kapasitas, double tarifPerPenumpang) {
        super(nama, kapasitas);
        this.tarifPerPenumpang = tarifPerPenumpang;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui terminal atau halte";
    }

    @Override
    String hitungTarif(int jarak) {
        return tarifPerPenumpang * kapasitas + " Rupiah";
    }
}

class KeretaApi extends Transportasi {
    double tarifDasar;
```

```
    KeretaApi(String nama, int kapasitas, double tarifDasar) {
        super(nama, kapasitas);
        this.tarifDasar = tarifDasar;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui aplikasi atau datang langsung ke
stasiun";
    }

    @Override
    String hitungTarif(int jarak) {
        return tarifDasar + (jarak * 500) + " Rupiah";
    }
}

public class Transport_07651 {
    public static void main(String[] args) {
        Transportasi taksi = new Taksi("Taksi", 4, 5000);
        Transportasi bus = new Bus("Bus", 40, 2000);
        Transportasi keretaApi = new KeretaApi("Kereta Api",
200, 5000);

        System.out.println("Cara memesan Taksi: " +
taksi.caraMemesan());
        System.out.println("Tarif Taksi untuk 10 km: " +
taksi.hitungTarif(10) + "\n");

        System.out.println("Cara memesan Bus: " +
bus.caraMemesan());
        System.out.println("Tarif Bus untuk 10 km: " +
bus.hitungTarif(10) + "\n");

        System.out.println("Cara memesan Kereta Api: " +
keretaApi.caraMemesan());
        System.out.println("Tarif Kereta Api untuk 10 km: " +
keretaApi.hitungTarif(10) + "\n");
    }
```

```
}
```

Penjelasan

Kode di atas merupakan penerapan abstraksi dalam Java, di mana class abstract Transportasi digunakan sebagai kerangka kerja untuk berbagai jenis transportasi seperti Taksi, Bus, dan KeretaApi. Kelas abstrak ini mendefinisikan metode caraMemesan() dan hitungTarif(int jarak) yang wajib diimplementasikan oleh subkelasnya. Setiap jenis transportasi memiliki logika perhitungan tarif dan cara pemesanan yang berbeda sesuai karakteristiknya. Kode ini juga memanfaatkan polymorphism, di mana semua subkelas diperlakukan sebagai objek Transportasi di dalam metode main() untuk mengakses metode yang diimplementasikan masing-masing kelas.

Output

```
PS D:\Kuliah\Semester 3\Coding Java\Praktikum 4>  & 'C:\Program Files\Java\jdk-23\bi
2216f58c8b439ae6de6f936\redhat.java\jdt_ws\Praktikum 4_6b395a43\bin' 'Transport_0765
Cara memesan Taksi: Pesan melalui aplikasi
Tarif Taksi untuk 10 km: 50000.0

Cara memesan Bus: Pesan melalui terminal atau halte
Tarif Bus untuk 10 km: 80000.0

Cara memesan Kereta Api: Pesan melalui aplikasi atau datang langsung ke stasiun
Tarif Kereta Api untuk 10 km: 1000000.0

PS D:\Kuliah\Semester 3\Coding Java\Praktikum 4>
```

TUGAS & EVALUASI

Soal Tugas & Evaluasi

Buatlah class interface Pembayaran dan Transportasi berdasarkan studi kasus dari soal no.3, kemudian implementasikan kedua class tersebut pada class Taksi, Bus, dan KeretaApi yang baru!

PETUNJUK:

- Interface Transportasi berisi method void bergerak() dan double hitungTarif().

- Interface Pembayaran berisi method void bayar().

Jawaban

Ketik jawaban disini …

Source Code

```
abstract class Transportasi {
    String nama;
    int kapasitas;

    abstract String  caraMemesan();

    abstract double hitungTarif(int jarak);

    abstract void bergerak();

    Transportasi(String nama, int kapasitas){
        this.nama = nama;
        this.kapasitas = kapasitas;

    }
}

interface Pembayaran{
    void bayar(double jumlah);
}
```

```
class Taksi extends Transportasi implements Pembayaran
{
    double tarifPerKm;


    Taksi(String nama, int kapasitas, double tarifPerKm){
        super(nama, kapasitas);
        this.tarifPerKm = tarifPerKm;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui aplikasi";
    }

    @Override
    double hitungTarif(int jarak) {
        return tarifPerKm * jarak;
    }

    @Override
    public void bayar(double jumlah) {
        System.out.println("Bayar dengan uang tunai atau kartu "
+ jumlah + " Rupiah");
    }

    @Override
    void bergerak() {
        System.out.println(nama + " Taksi bergerak");
    }
}

class Bus extends Transportasi implements Pembayaran{
    double tarifPerPenumpang;


    Bus(String nama, int kapasitas, double tarifPerPenumpang){
        super(nama, kapasitas);
```

```
        this.tarifPerPenumpang = 2000;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui terminal atau halte";
    }

    @Override
    double hitungTarif(int jarak) {
        return tarifPerPenumpang * kapasitas;
    }

    @Override
    public void bayar(double jumlah) {
        System.out.println("Bayar dengan uang tunai " + jumlah +
" Rupiah");
    }

    @Override
    void bergerak() {
        System.out.println(nama + " Bus bergerak");
    }
}

class KeretaApi extends Transportasi{
    double tarifDasar;

    KeretaApi(String nama, int kapasitas, double tarifDasar){
        super(nama, kapasitas);
        this.tarifDasar = 5000;
    }

    @Override
    String caraMemesan() {
        return "Pesan melalui aplikasi atau datang langsung ke
stasiun";
    }
```

```
    @Override
    double hitungTarif(int jarak) {
        return tarifDasar * kapasitas;
    }

    @Override
    void bergerak() {
        System.out.println("Kereta Api bergerak");
    }
}


public class Tranport2_07651 {
    public static void main(String[] args) {
        Taksi taksi = new Taksi("Blue Bird", 4, 5000);

        taksi.bergerak();
        double tarifTaksi = taksi.hitungTarif(10);
        taksi.bayar(tarifTaksi);

    }
}
```

Penjelasan

Kode di atas menunjukkan penerapan **abstract class** dan **interface** dalam Java untuk mendefinisikan kerangka dasar transportasi. Kelas abstrak **Transportasi** digunakan untuk memberikan atribut dan metode umum seperti caraMemesan, hitungTarif, dan bergerak, sementara **interface Pembayaran** menyediakan kontrak untuk metode pembayaran. Kelas **Taksi** dan **Bus** mewarisi Transportasi dan mengimplementasikan Pembayaran, sehingga mereka memiliki perilaku khusus seperti cara memesan, tarif, dan metode pembayaran yang spesifik. **KeretaApi**, meskipun tidak mengimplementasikan Pembayaran, tetap mengimplementasikan metode Transportasi.

TUGAS & EVALUASI

Output