# Soal Praktikum

Abstraction adalah fitur yang dapat mengabaikan fungsional keseluruhan secara detail tanpa menghilangkan fungsionalnya secara umum

Interface adalah kontrak yang mendefinisikan metode yang harus diimplementasikan oleh kelas yang menerapkannya.

# Jawaban

Ketik jawaban disini …

# Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

Tulis Penjelasan disini …

**Output**

Masukan screenshot output disini

# Soal Praktikum

Jelaskan apa yang dimaksud Exception Handling!

Exception Handling adalah masalah yang terjadi Ketika kita mengeksekusi program. Pada saat masalah ini terjadi alur jalanya program yang terganggu Bisa terjadi karena user salah memasukan Nilai, file tidak dapat di temukan atau terputusnya koneksi jaringan di saat melakukan Komunikasi

# Jawaban

Ketik jawaban disini …

# Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

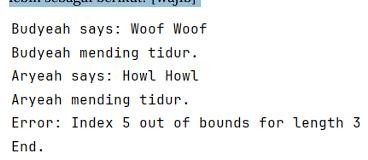
Tulis Penjelasan disini …

**Output**

Masukan screenshot output disini

# Soal Praktikum

Budi Arye membuat sedang belajar tentang abstraction dan exception handling Ia membuat program sebagai berikut :

Bantu Budi menyelesaikan kodingannya agar mendapatkan output kurang lebih sebagai berikut: [wajib]

# Jawaban

Ketik jawaban disini …

# Source Code

abstract class Animal {

String nama;

// Konstruktor untuk menginisialisasi nama

Animal(String nama) { this.nama = nama;

}

abstract void bersuara();

void sleep() {

System.out.println(nama + " mending tidur.");

}

}

class Wolf extends Animal {

Wolf(String name) { super(name);

}

@Override

void bersuara() {

System.out.println(nama + " says: Howl Howl");

}

}

class Dog extends Wolf {

Dog(String name) { super(name);

}

@Override

void bersuara() {

System.out.println(nama + " says: Woof Woof");

}

}

public class Main {

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

Wolf seringila = new Wolf("Budyeah"); seringila.bersuara(); seringila.sleep();

Dog anjing = new Dog("Aryeah"); anjing.bersuara(); anjing.sleep();

int[] numbers = {1, 2, 3, 4, 5}; System.out.println(numbers[5]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: " + e.getMessage());

} finally {

System.out.println("end.");

}

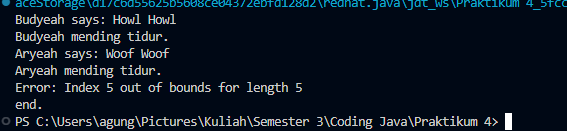
}

}

**Penjelasan**

Kelas abstrak Animal mendefinisikan metode bersuara() yang harus diimplementasikan oleh kelas turunan, yaitu Wolf dan Dog. Metode sleep() pada kelas Animal memberikan pesan bahwa hewan tersebut sedang tidur. Di dalam main, objek Wolf dan Dog dibuat untuk menunjukkan implementasi metode bersuara() dan sleep(). Exception handling digunakan untuk menangani kesalahan ketika mencoba mengakses elemen array di luar batas, dengan menangkap ArrayIndexOutOfBoundsException. Program ini juga menampilkan pesan kesalahan dan memastikan bahwa blok finally selalu dieksekusi setelah pengecualian.

**Output**



# Soal Praktikum

Dari soal praktikum sebelumnya kalian telah membuat class Paket yang di extend atau diturunkan ke kelas PaketDomestik dan PaketInternasional.Buatlah interface Pembayaran dengan metode prosesPembayaran(double jumlah). Kemudian buatlah class PembayaranTunai dan PembayaranKartu yang mengimplementasikan interface Pembayaran,dengan syarat sebagai berikut:

* Pada class PembayaranKartu, tambahkan atribut nomorkartu dan namapemilik.
* Tambahkan metode bayar pada class PaketDomestik yang menerima objek Pembayaran sebagai parameter dan memanggil metode prosesPembayaran.
* Buatlah objek PembayaranTunai dan PembayaranKartu, lalu gunakan metode bayar pada objek PaketDomestik untuk melakukan

pembayaran.

# Jawaban

Ketik jawaban disini …

# Source Code

interface Pembayaran {

void prosesPembayaran(double jumlah);

}

class PembayaranTunai implements Pembayaran {

@Override

public void prosesPembayaran(double jumlah) {

System.out.println("Pembayaran Tunai sebesar: " +

jumlah);

}

}

class PembayaranKartu implements Pembayaran { String nomorkartu;

String namapemilik;

PembayaranKartu(String nomorkartu, String namapemilik) { this.nomorkartu = nomorkartu;

this.namapemilik = namapemilik;

}

@Override

public void prosesPembayaran(double jumlah) { System.out.println("Pembayaran Kartu sebesar: " +

jumlah);

}

System.out.println("Nomor Kartu: " + nomorkartu); System.out.println("Nama Pemilik: " + namapemilik);

}

abstract class Paket { String namaPaket;

Paket(String namaPaket) { this.namaPaket = namaPaket;

}

abstract void bayar(Pembayaran pembayaran, double jumlah);

}

class PaketDomestik extends Paket { PaketDomestik(String namaPaket) {

super(namaPaket);

}

@Override

void bayar(Pembayaran pembayaran, double jumlah) { System.out.println("Pembayaran untuk Paket Domestik: " +

namaPaket);

pembayaran.prosesPembayaran(jumlah);

}

}

class PaketInternasional extends Paket { PaketInternasional(String namaPaket) {

super(namaPaket);

}

@Override

void bayar(Pembayaran pembayaran, double jumlah) { System.out.println("Pembayaran untuk Paket

Internasional: " + namaPaket);

pembayaran.prosesPembayaran(jumlah);

}

}

public class Paket\_07651 {

public static void main(String[] args) {

Pembayaran pembayaranTunai = new PembayaranTunai(); Pembayaran pembayaranKartu = new PembayaranKartu("1234-

5678-9012-3456", "Budi Arye");

Bali");

Paket paketDomestik = new PaketDomestik("Paket Liburan

Paket paketInternasional = new PaketInternasional("Paket

Tour Eropa");

paketDomestik.bayar(pembayaranTunai, 150000);

paketInternasional.bayar(pembayaranKartu, 3000000);

}

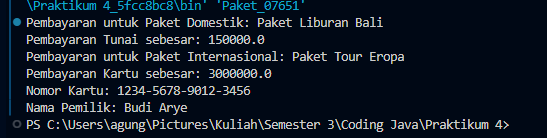
}

**Penjelasan**

Kode di atas mendemonstrasikan penggunaan interface di Java untuk mendefinisikan metode umum prosesPembayaran(double jumlah), yang kemudian

diimplementasikan oleh dua kelas pembayaran: PembayaranTunai dan PembayaranKartu. Kelas PaketDomestik dan PaketInternasional mewarisi kelas abstrak Paket dan mengimplementasikan metode bayar, yang menerima objek Pembayaran untuk memproses pembayaran paket. PembayaranTunai hanya mencetak jumlah pembayaran tunai, sedangkan PembayaranKartu juga mencetak informasi kartu yang digunakan. Kode ini menekankan prinsip polymorphism dengan menggunakan objek Pembayaran yang dapat berupa jenis pembayaran apa pun, baik tunai maupun kartu.

**Output**



# Soal Praktikum

Ketik soal disini …

# Jawaban

Ketik jawaban disini …

# Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

Tulis Penjelasan disini …

**Output**

Masukan screenshot output disini

# Soal Praktikum

Ketik soal disini …

# Jawaban

Ketik jawaban disini …

# Source Code

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class

**Penjelasan**

Tulis Penjelasan disini …

**Output**

Masukan screenshot output disini