

PEMROGRAMAN BERORIENTASI OBYEK
PRE TEST



Shera Zahra Alya Nasywa Hardian
(3122522026)
3 D3 PSDKU Sumenep

**PRODI D3 TEKNIK INFORMATIKA
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
PENS PSDKU SUMENEP**

1. Jelaskan secara detail apa yang kalian pahami tentang abstract Class

Abstract class adalah konsep dalam pemrograman berorientasi objek (OOP) yang memberikan struktur dasar untuk kelas-kelas turunannya. Kelas abstrak sendiri tidak dapat diinstansiasi, yang berarti tidak dapat membuat objek langsung dari kelas abstrak. Sebaliknya, kelas abstrak digunakan sebagai kerangka dasar untuk kelas-kelas turunannya.

Berikut beberapa karakteristik utama dari abstract class:

1. Tidak Dapat Diinstansiasi:

- tidak dapat membuat objek langsung dari kelas abstrak. Tujuannya adalah untuk menyediakan struktur umum yang akan diimplementasikan oleh kelas-kelas turunannya.

2. Mengandung Setidaknya Satu Metode Abstrak:

- Abstract class seringkali memiliki setidaknya satu metode abstrak. Metode abstrak adalah metode yang dideklarasikan di kelas abstrak tetapi tidak diimplementasikan di dalamnya. Implementasinya akan dilakukan oleh kelas turunan.

3. Boleh Memiliki Metode Non-Abstrak (Konkret):

- Abstract class juga dapat memiliki metode yang diimplementasikan (konkret). Ini dapat berupa metode yang tidak perlu diubah oleh kelas turunan atau metode yang memiliki implementasi dasar yang dapat dioverride.

4. Digunakan Sebagai Blueprints:

- Abstract class berfungsi sebagai "blueprints" atau cetakan untuk kelas-kelas turunannya. Mereka menyediakan struktur dan aturan dasar yang harus diikuti oleh kelas-kelas turunan.

5. Subclass Wajib Mengimplementasikan Metode Abstrak:

- Setiap kelas turunan dari abstract class wajib mengimplementasikan semua metode abstrak yang dideklarasikan di dalam abstract class tersebut. Ini memastikan bahwa kelas turunan memberikan implementasi konkret untuk setiap fungsi abstrak.

6. Dapat Mengandung Properti dan Metode Non-Abstrak:

- Selain metode abstrak, abstract class dapat memiliki properti (variabel) dan metode non-abstrak yang memiliki implementasi konkret.

2. Jelaskan secara detail mengenai fungsi dari Interface

Interface adalah sebuah konsep dalam pemrograman berorientasi objek (OOP) yang menyediakan cara untuk mendefinisikan metode tanpa memberikan implementasi konkret. Interface memberikan kontrak atau spesifikasi tentang apa yang harus dilakukan oleh kelas-kelas yang mengimplementasikannya. Interface membantu dalam mencapai abstraksi lebih tinggi dan memisahkan antara apa yang harus dilakukan oleh suatu kelas dan bagaimana itu dilakukan.

Berikut adalah beberapa poin kunci tentang fungsi dan karakteristik dari interface:

1. Tidak Memiliki Implementasi Konkret:

- Interface tidak memiliki implementasi sendiri. Itu hanya berisi deklarasi metode atau properti tanpa memberikan logika implementasinya. Implementasi dilakukan oleh kelas-kelas yang mengimplementasikan interface tersebut.

2. Mendefinisikan Kontrak:

- Interface mendefinisikan kontrak atau spesifikasi yang harus dipatuhi oleh kelas-kelas yang menggunakannya. Oleh karena itu, kelas yang mengimplementasikan interface diharapkan untuk menyediakan implementasi konkret untuk setiap metode yang dideklarasikan oleh interface.

3. Mendukung Polimorfisme:

- Dengan menggunakan interface, dapat mencapai polimorfisme di dalam kode. Polimorfisme memungkinkan objek dari kelas yang berbeda untuk digunakan secara seragam melalui antarmuka yang sama.

4. Memungkinkan Multiple Inheritance:

- Beberapa bahasa pemrograman yang mendukung interface, seperti Java dan C#, memungkinkan kelas untuk mengimplementasikan lebih dari satu interface. Ini memfasilitasi multiple inheritance dan memungkinkan kelas untuk menyediakan implementasi untuk berbagai kontrak sekaligus.

5. Mengurangi Ketergantungan pada Implementasi:

- Penggunaan interface membantu mengurangi ketergantungan pada implementasi kelas-kelas. Kode yang bergantung pada antarmuka tidak perlu tahu detail implementasi kelas tersebut, hanya perlu tahu bahwa kelas tersebut mematuhi kontrak yang dijelaskan oleh interface.

6. Memisahkan Antara Antarmuka dan Implementasi:

- Interface membantu dalam pemisahan antara antarmuka (interface) dan implementasi kelas. Ini memungkinkan pengembang untuk fokus pada apa yang harus dilakukan oleh kelas tanpa terlalu khawatir tentang bagaimana hal itu dilakukan.

3. Buatlah Program sederhana menggunakan konsep OOP yang menerapkan Inheritance, Abstract dan Interface

Class person

```
1
2 package pretest;
3
4 public abstract class Person {
5     protected String name;
6     protected int age;
7
8     public Person(String name, int age) {
9         this.name = name;
10        this.age = age;
11    }
12
13    public abstract void displayInfo();
14 }
```

Class medicalstaff

```
1
2 package pretest;
3
4 public interface MedicalStaff {
5     void performDuties();
6 }
```

Class nurse

```
1
2 package pretest;
3
4 public class Nurse extends Person implements MedicalStaff {
5     private int experienceYears;
6
7     public Nurse(String name, int age, int experienceYears) {
8         super(name, age);
9         this.experienceYears = experienceYears;
10    }
11
12    @Override
13    public void displayInfo() {
14        System.out.println("Nurse: " + name + ", Age: " + age + ", Experience Years: " + experienceYears);
15    }
16
17    @Override
18    public void performDuties() {
19        System.out.println(name + " membantu dokter dan merawat pasien.");
20    }
21 }
```

Class doctor





```
1
2 package pretest;
3
4 public class Doctor extends Person implements MedicalStaff {
5     private String specialization;
6
7     public Doctor(String name, int age, String specialization) {
8         super(name, age);
9         this.specialization = specialization;
10    }
11
12    @Override
13    public void displayInfo() {
14        System.out.println("Doctor: " + name + ", Age: " + age + ", Specialization: " + specialization);
15    }
16
17    @Override
18    public void performDuties() {
19        System.out.println(name + " sedang memeriksa pasien dan memberikan perawatan medis.");
20    }
21 }
```

Class hospitalapp (main)

```
1
2 package pretest;
3
4 public class HospitalApp {
5     public static void main(String[] args) {
6         Doctor doctor = new Doctor(name: "Dr. Malik Abraham", age:35, specialization: "Cardiologist");
7         Nurse nurse = new Nurse(name: "Nurse Sheila Dara", age:28, experienceYears: 5);
8
9         doctor.displayInfo();
10        doctor.performDuties();
11
12        System.out.println();
13
14        nurse.displayInfo();
15        nurse.performDuties();
16    }
17 }
```

Hasil

Output - pretest (run)



run:
Doctor: Dr. Malik Abraham, Age: 35, Specialization: Cardiologist
Dr. Malik Abraham sedang memeriksa pasien dan memberikan perawatan medis.

Nurse: Nurse Sheila Dara, Age: 28, Experience Years: 5
Nurse Sheila Dara membantu dokter dan merawat pasien.
BUILD SUCCESSFUL (total time: 0 seconds)

Penjelasan

1. Person (Kelas Abstrak):
- `Person` adalah kelas abstrak yang memiliki atribut `name` dan `age`. Metode `displayInfo` dideklarasikan sebagai metode abstrak yang harus diimplementasikan oleh kelas turunannya.
2. MedicalStaff (Interface):
- `MedicalStaff` adalah interface yang menyatakan bahwa kelas yang mengimplementasikannya harus memiliki metode `performDuties`. Interface digunakan untuk menetapkan kontrak yang harus dipatuhi oleh kelas yang berbeda.
3. Doctor (Kelas Turunan dari Person dan Mengimplementasikan MedicalStaff):
- `Doctor` adalah kelas yang mengimplementasikan `MedicalStaff` dan merupakan turunan dari kelas abstrak `Person`. Ini memiliki atribut tambahan `specialization` yang merupakan spesialisasi dokter. Metode `displayInfo` diimplementasikan dari kelas abstrak `Person`, dan metode `performDuties` diimplementasikan dari interface `MedicalStaff`.

4. Nurse (Kelas Turunan dari Person dan Mengimplementasikan MedicalStaff):

- `Nurse` adalah kelas yang juga mengimplementasikan `MedicalStaff` dan merupakan turunan dari kelas abstrak `Person`. Ini memiliki atribut tambahan `experienceYears` yang menunjukkan tahun pengalaman perawat. Metode `displayInfo` diimplementasikan dari kelas abstrak `Person`, dan metode `performDuties` diimplementasikan dari interface `MedicalStaff`.

5. HospitalApp (Kelas Utama):

- `HospitalApp` adalah kelas utama yang memiliki metode `main`. Di dalamnya, objek `Doctor` dan `Nurse` dibuat, informasi tentang mereka ditampilkan menggunakan metode `displayInfo`, dan tugas medis yang dilakukan oleh mereka disimulasikan menggunakan metode `performDuties`.

6. Relasi Antara Kelas-Kelas:

- `Doctor` dan `Nurse` adalah turunan dari kelas abstrak `Person`, sehingga mereka mewarisi atribut dan metode dari `Person`. Selain itu, keduanya mengimplementasikan interface `MedicalStaff`, yang memaksa mereka untuk memberikan implementasi untuk metode `performDuties`.

Dengan struktur ini, program menciptakan objek yang merepresentasikan dokter dan perawat, memanfaatkan sifat polimorfisme melalui interface `MedicalStaff`. Hal ini memungkinkan penggunaan umum dari metode `performDuties` tanpa peduli tentang tipe spesifik objeknya. Konsep ini membantu dalam pemisahan tanggung jawab dan meningkatkan fleksibilitas program.