**LAPORAN RESMI**

**MODUL IV**

**ABSTRACT CLASS & ABSTRACT METHOD**

**PEMROGRAMAN BERBASIS OBJEK**

**Disetujui : Juni 2024**

**Asisten**

**MUHAMMAD ROSYID MAULANA 20.04.411.00019**

**NAMA : ABAS PERMADANI**

**N.R.P : 230441100107**

**DOSEN : ACHMAD ZAIN NUR S.Kom M.T**

**ASISTEN : MUHAMMAD ROSYID MAULANA**

**TGL PRAKTIKUM : 18 MEI 2024**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# BAB I PENDAHULUAN

## **Latar Belakang**

Prakitkum merupakan suatu kegiatan belajar yang berbentuk suatu pengamatan terhadap suatu percobaan atau pengujian yang dilaksanakan di labolatorium dan diikuti dengan analisis serta penyimpulan terhadap pengamatan yang dilakukan tersebut. Pada kesempatan kali ini praktikan akan membahas tentang *Abstract class* dan *abstract method .*yang ada di dalam Bahasa pemrograman *Java*.

Dalam pemrograman Java, abstract class adalah kelas yang tidak dapat diinstansiasi secara langsung dan digunakan untuk menyediakan kerangka kerja dasar bagi kelas-kelas turunannya. Abstract class dideklarasikan dengan kata kunci abstract dan dapat berisi metode-metode lengkap serta abstract method. Abstract method adalah metode yang dideklarasikan tanpa implementasi, memaksa subclass untuk memberikan definisi konkret. Misalnya, jika kita memiliki abstract class Hewan dengan abstract method bersuara, setiap subclass seperti Anjing dan Kucing harus mengimplementasikan metode ini dengan cara yang sesuai, misalnya, Anjing mengimplementasikan bersuara dengan mencetak "Anjing menggonggong" dan Kucing dengan mencetak "Kucing mengeong".

Abstract class memungkinkan pengembang untuk mendefinisikan perilaku umum dan memastikan konsistensi perilaku di antara kelas-kelas yang berbeda, sehingga meningkatkan modularitas, menjaga konsistensi, dan mengurangi duplikasi kode. Dengan menggunakan abstract class, pengembang dapat mengelompokkan kelas-kelas yang memiliki kesamaan dalam sebuah hierarki yang logis, sehingga mempermudah pemeliharaan dan pengembangan sistem.

## **Tujuan**

* Mahasiswa mampu memahami konsep Abstract Class dan Abstract Method dalam pemrograman berorientasi objek serta mampu mengimplementasikannya.

# BAB II DASAR TEORI

## **Konsep Abstract Class**

Abstract class atau kelas abstrak adalah kelas yang terletak di posisi tertinggi dalam hierarki class. Class ini tidak dapat diinstansiasi karena masih bersifat abstrak. Class ini hanya berisi variabel umum dan deklarasi method tanpa detail penggunaannya (abstract method). Selanjutnya class yang menjadi turunan dari abstract class ini yang akan menentukan detail penggunaan method tersebut.



## **Deklarasi Abstract Class**

* Abstract class dideklarasikan dengan cara sebagai berikut :

https://lh7-us.googleusercontent.com/RyqjaQuOPkwtegIndFt46DrW1iKEE9DjSa1NbIJnVRA4191ng-onCmtt_h12qI-k5o6Ob8W5MWqAP0RJbKF59u5sAOlyYM_nllYAQAuF_L9e08kGno5bnI4HJwwxYYNbrb530GcEOHe9MfA99ANsxKQ

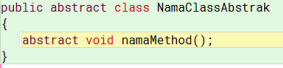
* Abstract Class digunakan dengan cara sebagai berikut :

https://lh7-us.googleusercontent.com/HTphULKRrBP_A9CXlWg36yf_6jyqgtMfVVZYj-SQiihRER-UwJ9JQh_6HRDqnSo5LnyJ7OT0cAmJSUCBTpNBVwzuPW3i0-6iEbv42irW4iixlLrkCWiD-oWNbHg70qi0VWZomIcwaXiNdq7BCEk3u3A

## **Abstract Method**

Abstract method adalah method yang dideklarasikan tanpa body method dan digunakan menggunakan kata kunci *abstract.*

Contoh :

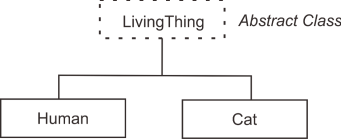


Java memiliki aturan-aturan dalam penggunaan method abstrak dan class abstrak sebagai berikut :

* Class yang didalamnya terdapat abstract method harus dideklarasikan sebagai abstract class.
* Abstract class tidak dapat diintansi, tetepai harus diturunkan.
* Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak memiliki abstract method.
* Abstract method tidak boleh mempunyai body method dan demikian juga sebaliknya bahwa method yang tidak ditulis body methodnya maka harus dideklarasikan sebagai abstract method.

## **Contoh Program**

Kita akan membuat superclass bernama LivingThing. Class ini mempunyai method tertentu seperti breath, eat, sleep, dan walk. Akan tetapi, ada beberapa method di dalam superclass yang sifatnya tidak dapat digeneralisasi. Kita ambil contoh, method walk. Tidak semua kehidupan berjalan (walk) dalam cara yang sama. Ambil manusia sebagai misal, kita manusia berjalan dengan dua kaki, dimana kehidupan lainnya seperti kucing berjalan dengan empat kaki. Akan tetapi, beberapa ciri umum dalam kehidupan sudah biasa, itulah kenapa kita inginkan membuat superclass umum dalam hal ini.



Dalam program ini ada,

1. Abstract Class yaitu :

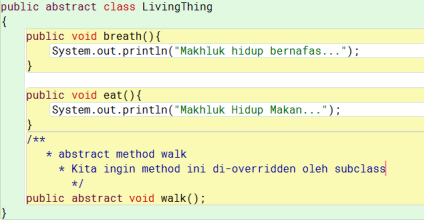
* Abstract Class Living Thing

1. Subclass yaitu :

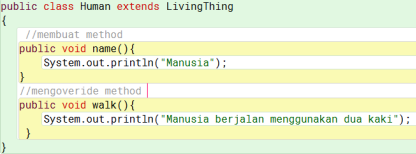
* Class Human
* Class Cat

1. Class yang digunakan untuk menjalankan program yaitu :

* Class Run

Membuat abstract class LivingThing :

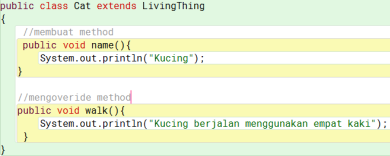
Ketika class meng-extend class abstract LivingThing, dibutuhkan untuk override method abstract walk(),seperti gambar dibawah ini



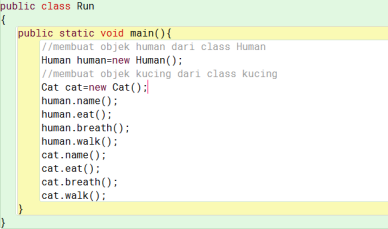
Jika class Human tidak dapat overridde method walk, kita akan menemui pesan error berikut ini,

https://lh7-us.googleusercontent.com/A_bd7vJ0f4ZtRRPD34WZTjlYaM4DlywspmmepvQr_Ec19fXO3gaKJPiyi3xHSOPTnQA7H9JmSe643YHvF3pfypBc6JZ14jwtxCa6fdSq-hmQ1N_z5sKuFgg30taufmvInn9vxTyZD3ZWGPfqCnlipRw

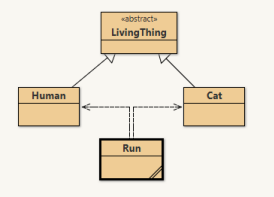
Sama seperti class Human , class Cat juga ketika meng-extend class abstract LivingThing maka harus mengoveride method walk jika tidak akan terjadi error.



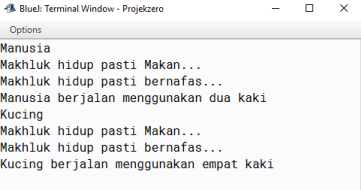
Membuat class bernama Run yang digunakan untuk menjalankan program dengan membuat objek dari masing-masing class kemudian memanggil semua methodnya seperti pada gambar dibawah ini.



Setelah selesai membuat semua classnya maka akan seperti gambar dibawah ini.



## **Running Program**

****

# BAB IV IMPLEMENTASI

## **Soal**

Buatlah sebuah program Java yang dapat mengelola data diri karyawan dalam sebuah perusahaan. Program ini harus menggunakan konsep "abstract class" dan "abstract method" untuk mendefinisikan atribut dan method umum yang dimiliki oleh semua jenis karyawan. Gunakan juga konsep polymorphism untuk memproses data karyawan dari berbagai jenis (misalnya, Karyawan Tetap dan Karyawan Kontrak) secara dinamis berdasarkan input dari pengguna. Simpan data karyawan dalam array.

* Spesifikasi

1. Abstract Class Karyawan :

* Atribut: nama, usia, dan posisi.
* Method abstrak: tampilkanData().

1. Class KaryawanTetap dan KaryawanKontrak yang Mengextends Abstract Class Karyawan :

* Implementasi method tampilkanData() untuk menampilkan data diri karyawan dengan format yang sesuai.
* Class KaryawanTetap memiliki tambahan atribut gajiBulanan.
* Class KaryawanKontrak memiliki tambahan atribut upahPerJam dan jumlahJamKerja.

1. Main Class :

* Meminta pengguna untuk memasukkan data diri karyawan (nama, usia, posisi, jenis karyawan, dan data tambahan sesuai jenis karyawan).
* Menyimpan data karyawan dalam sebuah array.
* Menampilkan semua data karyawan yang telah dimasukkan.

## **Source-code**

**Package Karyawan**

package com.mycompany.modul45;

abstract class Karyawan {

String nama;

int usia;

String posisi;

public Karyawan(String nama, int usia, String posisi){

this.nama = nama;

this.usia = usia;

this.posisi = posisi;

}

abstract void tampilkanData();

}

**Package KaryawanKontrak**

package com.mycompany.modul45;

public class KaryawanKontrak extends Karyawan {

double upahPerJam;

int jumlahJamKerja;

public KaryawanKontrak(String nama, int usia, String posisi, double upahPerJam, int jumlahJamKerja) {

super(nama, usia, posisi);

this.upahPerJam = upahPerJam;

this.jumlahJamKerja = jumlahJamKerja;

}

@Override

void tampilkanData() {

System.out.println("===== Karyawan Kontrak =====");

System.out.println("Nama : " + nama);

System.out.println("Usia : " + usia);

System.out.println("Posisi : " + posisi);

System.out.println("Upah Per Jam : " + upahPerJam);

System.out.println("Jumlah Jam Kerja : " + jumlahJamKerja);

}

}

**Package KaryawanTetap**

package com.mycompany.modul45;

public class KaryawanTetap extends Karyawan {

double gajibulanan;

public KaryawanTetap(String nama, int usia, String posisi, double gajibulanan){

super(nama, usia, posisi);

this.gajibulanan = gajibulanan;

}

void tampilkanData(){

System.out.println("===== Karyawan Tetap =====");

System.out.println("Nama : " + nama);

System.out.println("Usia : " + usia);

System.out.println("Posisi : " + posisi);

System.out.println("Gaji : " + gajibulanan);

}

}

**Package modul45**

package com.mycompany.modul45;

import java.util.ArrayList;

import java.util.Scanner;

public class Modul45 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ArrayList<Karyawan> karyawanList = new ArrayList<>();

while (true) {

System.out.println("===== Masukkan Data Karyawan =====");

System.out.print("Nama Karyawan : ");

String nama = scanner.nextLine();

System.out.print("Usia Karyawan : ");

int usia = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Posisi Karyawan : ");

String posisi = scanner.nextLine();

System.out.print("Jenis Karyawan (1. Tetap, 2. Kontrak): ");

int jenisKaryawan = scanner.nextInt();

if (jenisKaryawan == 1) {

System.out.print("Gaji Bulanan : ");

double gajiBulanan = scanner.nextDouble();

scanner.nextLine(); // Consume newline

Karyawan karyawan = new KaryawanTetap(nama, usia, posisi, gajiBulanan);

karyawanList.add(karyawan);

} else if (jenisKaryawan == 2) {

System.out.print("Upah per Jam : ");

double upahPerJam = scanner.nextDouble();

System.out.print("Jumlah Jam Kerja : ");

int jumlahJamKerja = scanner.nextInt();

scanner.nextLine(); // Consume newline

Karyawan karyawan = new KaryawanKontrak(nama, usia, posisi, upahPerJam, jumlahJamKerja);

karyawanList.add(karyawan);

} else {

System.out.println("Jenis karyawan tidak valid!");

}

System.out.print("Apakah Anda ingin menambahkan karyawan lagi? (y/n): ");

String lagi = scanner.nextLine();

if (!lagi.equalsIgnoreCase("y")) {

break;

}

}

System.out.println("\nData Karyawan :");

for (Karyawan karyawan : karyawanList) {

karyawan.tampilkanData();

System.out.println();

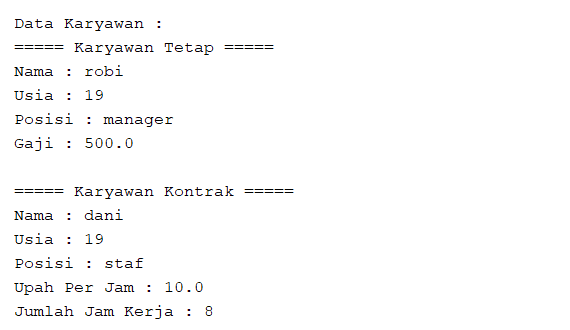
}

scanner.close();

}

}

## **Hasil**



## **Penjelasan**

Program di atas merupakan suatu program Java untuk menginputkan data karyawan dari suatu perusahaan. Pada program tersebut terdapat class Karyawan yang merupakan abstract class dan bertugas untuk mewariskan atribut serta method abstrak yang ia punya terhadap subclass-nya atau child class-nya.

Kemudian ada juga class KaryawanTetap dan KaryawanKontrak, kelas-kelas ini merepresentasikan child class. Kelas KaryawanTetap dan KaryawanKontrak mewarisi atribut dari kelas Karyawan dan mengimplementasikan method abstrak tampilkanData(). Selain itu, juga ada class Main yang merupakan tempat eksekusi utama dari program ini. Pada class ini juga terdapat ArrayList untuk menyimpan hasil inputan data karyawan dari user.

Ketika program berjalan, user akan diminta untuk mengisi data karyawan seperti nama, usia, posisi, dan jenis karyawan (tetap atau kontrak). Jika karyawan adalah karyawan tetap, user juga akan diminta mengisi gaji bulanan. Jika karyawan adalah karyawan kontrak, user akan diminta mengisi upah per jam dan jumlah jam kerja. Data karyawan tersebut kemudian disimpan dalam ArrayList. Setelah semua data karyawan dimasukkan, program akan menampilkan semua data karyawan yang telah diinputkan oleh user.

# BAB V PENUTUP

## **Analisa**

Dari hasil praktikum, Abstract class dan abstract method adalah elemen penting dalam paradigma pemrograman berorientasi objek (OOP) yang memberikan struktur dan fleksibilitas dalam pengembangan perangkat lunak. Abstract class adalah kelas yang tidak dapat diinstansiasi dan sering kali berperan sebagai kerangka dasar untuk kelas-kelas turunannya. Dalam sebuah proyek pemrograman, penggunaan abstract class memungkinkan pengembang untuk mengorganisir kode dengan lebih baik dengan mengelompokkan kelas-kelas yang memiliki sifat atau perilaku yang sama. Misalnya, dalam pengembangan permainan, abstract class dapat digunakan untuk mengelompokkan karakter-karakter yang memiliki sifat atau perilaku umum.

Secara keseluruhan, penggunaan abstract class dan abstract method memungkinkan pengembang untuk membuat kode yang lebih terstruktur, fleksibel, dan mudah dipelihara dalam proyek pemrograman berorientasi objek. Dengan mengelompokkan kode yang memiliki sifat atau perilaku yang sama menggunakan abstract class, dan dengan mendefinisikan antarmuka yang harus diikuti oleh kelas-kelas turunan menggunakan abstract method, pengembang dapat meningkatkan produktivitas dan kualitas perangkat lunak yang mereka kembangkan.

## **Kesimpulan**

* + 1. Abstract class adalah class yang tidak dapat diinstansiasi secara langsung.
    2. Abstract method adalah metode tanpa implementasi yang harus diimplementasikan oleh subclass.
    3. Abstract class menyediakan kerangka dasar untuk kelas turunan.
    4. Abstract method digunakan untuk mendefinisikan perilaku umum yang harus diimplementasikan oleh subclass.
    5. Abstract class dan abstract method membantu dalam membuat struktur kode yang konsisten, modular, dan mudah dikelola.