**TUGAS 6**

**MATA KULIAH ALGORITMA DAN DASAR PEMROGRAMAN SEMESTER GENAP TAHUN AKADEMIK 2023-2024**



**Oleh**

**IFAN ALI MURTADHO**

**(4123014)**

**JURUSAN SISTEM INFORMASI**

**UNIVERSITAS PESANTREN TINGGI DARUL 'ULUM JOMBANG**

**2024**

**TUGAS PENDAHULUAN**

1. Apa yang dimaksud dengan inheritance?
2. Buatlah contoh kasus yang menerapkan konsep inheritance !
3. Adakah perbedaan cara mengakses member class milik parent dan member class milik sendiri? Jelaskan melalui contoh ! (Silahkan memanfaatkan jawaban soal nomor 2.)
4. Apa yang dimaksud dengan konsep single inheritance ?
5. Apa yang dimaksud dengan konsep multi level inheritance ?

**Jawaban**

1. inhertance adalah mekanisme dimana salah satu kelas dapat mengambil properti dan method dari kelas lain. Kelas yang mengambil properti dan method disebut sebagai kelas anak atau subclass, sedangkan kelas yang dipilih untuk diwarisi disebut sebagai kelas induk atau base class.
2. hubungan inheritance antara kelas Vehicle sebagai superclass dan kelas Car sebagai subclass. Vehicle akan memiliki atribut umum seperti brand dan metode drive(), sementara Car akan memiliki atribut tambahan seperti numberOfDoors dan metode tambahan honkHorn().
3. Dalam contoh di atas, kelas Vehicle merupakan superclass yang memiliki atribut brand dan metode drive(). Kelas Car merupakan subclass dari Vehicle. Selain mewarisi atribut dan metode dari Vehicle, Car juga memiliki atribut tambahan numberOfDoors dan metode tambahan honkHorn(). Dalam method main(), kita membuat objek myCar dari kelas Car. Untuk mengakses atribut dan metode yang dimiliki oleh parent class (Vehicle), kita menggunakan sintaks myCar.brand dan myCar.drive(). Untuk mengakses atribut dan metode yang dimiliki oleh subclass sendiri (Car), kita menggunakan sintaks myCar.numberOfDoors dan myCar.honkHorn(). Jadi, perbedaan utama dalam cara mengakses member class milik parent dan milik sendiri adalah hanya pada penggunaan objek. Untuk mengakses member class milik parent, kita menggunakan objek dari subclass itu sendiri, sedangkan untuk mengakses member class milik subclass, kita juga menggunakan objek dari subclass tersebut.
4. single inheritance adalah prinsip dalam pemrograman berorientasi objek di mana sebuah kelas hanya dapat mewarisi sifat-sifat dari satu kelas lain, yaitu kelas induk atau superclass. Dengan kata lain, dalam single inheritance, sebuah kelas hanya dapat memiliki satu kelas induk.
5. multi level inheritance adalah prinsip dalam pemrograman berorientasi objek di mana sebuah kelas dapat mewarisi sifat-sifat dari kelas lain, yang pada gilirannya juga dapat menjadi kelas induk bagi kelas lainnya. Dengan kata lain, dalam multi level inheritance, suatu kelas dapat memiliki kelas induk dan dapat menjadi kelas induk bagi kelas lainnya dalam hierarki warisan.

Ini berarti kelas yang berada pada tingkat lebih rendah dalam hierarki dapat mengakses sifat-sifat dan perilaku yang diwarisi dari kelas-kelas yang berada pada tingkat lebih tinggi dalam hierarki warisan.

**LATIHAN**

**Latihan 1.**

Tempatkan class Base dan class Class1 di direktori yang sama. Apa

yang terjadi ketika Class1.java dikompile dan dijalankan jika sebelumnya

Base.java belum dikompile? Jelaskan !

1. Eror karena tidak bisa mengakases class base

**Latihan 2.**

Aturan overriding

1. Berdasarkan kode di bawah ini, akses modifier (public, protected atau private) apa yang diijinkan di tambahkan sebelum myMethod() baris 3?
2. Jika baris 3 seperti kode di bawah (apa adanya tanpa perubahan) keywords apa yang diijinkan ditambahkan sebelum myMethod baris 8?
   1. Protected
   2. protected void MyMethod(){

System.out.println("Mymethod");

}

**Latihan 3.**

1. Apa yang terjadi bila kedua kode dibawah ini dikompile dan dijalankan dalam satu direktori? Jelaskan !
2. Bagaimana solusi supaya tidak terjadi error?
3. Eror karena tidak ada atribut akses modifier seperti (public,protected,privat)
4. Menambahkan atribut akses modifier seperti protected untuk mengakses class sebelumnya contoh : public class P2 extends P1{

public static void main(String argv[]){

P2 p2 = new P2();

p2.aFancyMethod();

}

}

**Latihan 4.**

Mengimplementasikan UML class diagram dalam program untuk

soure code:

/\*\*

\*

\* @author ifanAliMurtadho

\*/

public class Tabungan {

protected double tingkatBunga;

protected int saldo;

public Tabungan(int saldo, double tingkatBunga) {

this.saldo = saldo;

this.tingkatBunga = tingkatBunga;

}

public double cekUang() {

return saldo + (saldo \* tingkatBunga);

}

}

Listing untuk mencoba code :

import Perbankan.Tabungan;

public class TesLatihan {

public static void main(String args[]) {

Tabungan tabungan = new Tabungan(5000, 8.5 / 100);

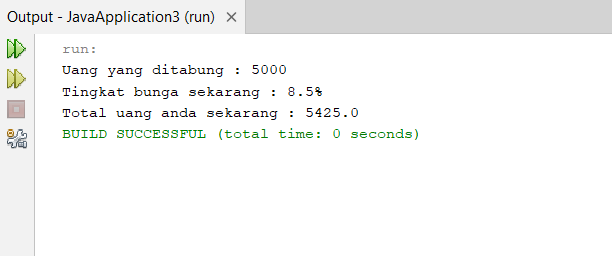
System.out.println("Uang yang ditabung : 5000");

System.out.println("Tingkat bunga sekarang : 8.5%");

System.out.println("Total uang anda sekarang : " + tabungan.cekUang());

}

}



**TUGAS**

Mengimplementasikan UML class diagram dalam program untuk package perbankan

soure code:

package Perbankan;

/\*\*

\*

\* @author ifanAliMurtadho

\*/

public class PengambilanUang {

private int saldo;

private int proteksi;

private int tingkatBunga;

public PengambilanUang(int saldo, int proteksi, int tingkatBunga) {

this.saldo = saldo;

this.proteksi = proteksi;

this.tingkatBunga = tingkatBunga;

}

public int getSaldo() {

return saldo;

}

public boolean ambilUang(int jumlah) {

if (saldo - jumlah >= proteksi) {

saldo -= jumlah;

return true;

} else {

return false;

}

}

}

Listing code :

import Perbankan.PengambilanUang;

public class TesTugas {

public static void main(String args[]) {

PengambilanUang tabungan = new PengambilanUang(5000, 1000, 2);

System.out.println("Uang yang ditabung : 5000");

System.out.println("Uang yang diproteksi : 1000");

System.out.println(" ");

System.out.println("Uang yang akan diambil : 4500 " + tabungan.ambilUang(4500));

System.out.println("Saldo sekarang : " + tabungan.getSaldo());

System.out.println(" ");

System.out.println("Uang yang akan diambil : 2500 " + tabungan.ambilUang(2500));

System.out.println("Saldo sekarang : " + tabungan.getSaldo());

}

}

Kode di atas merupakan implementasi sederhana dari sebuah Kelas PengambilanUang

1. Kelas PengambilanUang :

* Kelas ini mewakili sebuah rekening bank.
* Memiliki tiga variabel instance private:
* saldo: Mewakili saldo dari rekening.
* proteksi: Mewakili tingkat proteksi, yaitu saldo minimum yang diizinkan setelah penarikan.
* tingkatBunga: Mewakili tingkat bunga (walaupun tidak digunakan dalam kode ini).
* Kelas memiliki konstruktor PengambilanUang(int saldo, int proteksi, int tingkatBunga) untuk menginisialisasi variabel-variabel tersebut.
* Juga memiliki dua metode:
* getSaldo(): Mengembalikan saldo terkini dari rekening.
* ambilUang(int jumlah): Mencoba menarik jumlah tertentu (jumlah) dari rekening. Pertama, melakukan pengecekan apakah penarikan akan meninggalkan saldo yang cukup (saldo - jumlah >= proteksi). Jika iya, maka jumlah tersebut akan dikurangkan dari saldo dan mengembalikan true, menandakan penarikan berhasil. Jika tidak, akan mengembalikan false.

1. Kelas TesTugas:

* Kelas ini berisi metode main, berfungsi sebagai titik masuk program.
* Di dalam main, sebuah instance dari kelas PengambilanUang dengan nama tabungan dibuat dengan saldo awal 5000, tingkat proteksi 1000, dan tingkat bunga 2.
* Dilakukan dua operasi penarikan:
* Pertama, mencoba menarik 4500. Karena saldo rekening (5000) dikurangi jumlah penarikan (4500) lebih besar dari atau sama dengan tingkat proteksi (1000), penarikan berhasil. Saldo terkini setelah penarikan dicetak.
* Kedua, mencoba menarik 2500. Sekali lagi, karena saldo rekening (5000) dikurangi jumlah penarikan (2500) lebih besar dari atau sama dengan tingkat proteksi (1000), penarikan berhasil. Saldo terkini setelah penarikan ini dicetak.

Output program menampilkan setup awal rekening dan hasil dari percobaan penarikan, termasuk apakah berhasil (true untuk berhasil, false untuk gagal) dan saldo terkini setelah setiap penarikan.

**Output :**

