LAPORAN RESMI

PRAKTIKUM 10

PRAKTIKUM PEMROGRAMAN BERBASIS OBYEK



Nama : Muhammad Zaid

Kelas : 2 D4 Teknik Informatika A

NRP : 2110191013

PRAKTIKUM 10 – INHERITANCE

**PENDAHULUAN**

1. Jelaskan mengenai konsep inheritance!

Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan.

1. Deklarasikan dalam program java, sebuah class B, yang merupakan sub class dari class A?

class A{

public double y = 5;

}

public class B extends A{

public double y = 10;

System.out.println("Nilai y = " + this.y);

System.out.println("Nilai y = " + this.y);

}

**PERCOBAAN**

Percobaan 1 – Single Inheritance

1. Listing Program

class Animal{

void eat(){

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

}

}

class Dog extends Animal{

void bark(){

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

}

}

class TestInheritance{

public static void main(String args[]){

Dog d=new Dog();

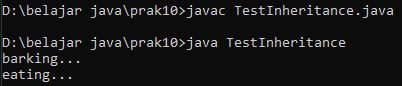
d.bark();

d.eat();

}

}

1. Output



1. Analisis

Pada percobaan pertama ini membuat single inheritance. Pertama yang dibuat adalah program parent

class-nya yaitu Animal.java dengan method void eat() yang melakukan print "eating...".

Kedua yaitu membuat child class dari class Animal yaitu Dog.java, kita mewariskan konstruktor

yang dimiliki parent class terlebih dahulu yaitu dengan cara mendeklarasikan class parent didalam

file yang sama kemudian baru mendeklarasikan child class dengan tambahan kata kunci

"extends". Setelah class parent dan child selesai, membuat class dengan main method bernama

TestInheritance.java yang melakukan deklarasi variabel Dog baru bernama d dan melakukan

d.bark() serta d.eat().

Percobaan 2 – Multilevel Inheritance

1. Listing Program

class Animal{

void eat(){

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

}

}

class Dog extends Animal{

void bark(){

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

}

}

class BabyDog extends Dog{

void weep(){

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

}

}

class TestInheritance2{

public static void main(String args[]){

BabyDog d=new BabyDog();

d.weep();

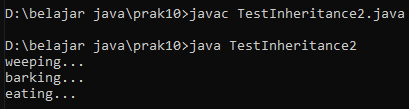
d.bark();

d.eat();

}

}

1. Output



1. Analisis

Pada percobaan 2, kita mencoba Multilevel Inheritance yang dimana sub-class dari super-

class memiliki sub-class lagi. Jadi di dalam program tersebut terdapat 2 child-class yaitu Dog dan

BabyDog. Program ini hampir sama dengan sebelumnya, dan di program ini kita hanya

menambahkan class baru yaitu BabyDog yang dimana extendsnay berasal dari class Dog dan

isinya hanya method weep() yang berisi “weeping...”. Di fungsi class test kita tinggal memanggil

method dari masing-masing tersebut.

Percobaan 3 – Hierarchical Inheritance

1. Listing Program

class Animal{

void eat(){

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

}

}

class Dog extends Animal{

void bark(){

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

}

}

class Cat extends Animal{

void meow(){

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

}

}

class TestInheritance3{

public static void main(String args[]){

Cat c=new Cat();

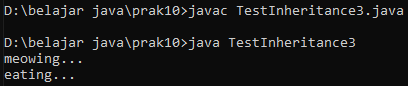
c.meow();

c.eat();

}

}

1. Output



1. Analisis

Pada percobaan ini ditunjukkan bagaimana jika dilakukan hierarchical inheritance di java.

Disini kelas yang menjadi parent dari kelas turunan-turunannya adalah kelas Animal yang sama seperti percobaan sebelumnya yaitu memiliki metode eat(). Kemudian terdapat kelas Dog yang menurunkan dari kelas Animal yang menambahkan metode bark(). Lalu juga terdapat kelas Cat yang juga menurunkan dari kelas Animal dengan menambahkan metode meow(). Kelas-kelas tersebut kemudian diuji coba pada kelas TestInheritance3 yang memiliki metode main sehingga program bisa dijalankan.

Percobaan 4 – Konstruktor di Inheritance

1. Listing Program

class Bicycle{

public int gear;

public int speed;

public Bicycle(int gear, int speed){

this.gear = gear;

this.speed = speed;

}

public void applyBrake(int decrement){

speed -= decrement;

}

public void speedUp(int increment){

speed += increment;

}

public String toString(){

return("No of gears are "+gear+"\n"+ "speed of bicycle is "+speed);

}

}

class MountainBike extends Bicycle{

public int seatHeight;

public MountainBike(int gear,int speed,int startHeight){

super(gear, speed);

seatHeight = startHeight;

}

public void setHeight(int newValue){

seatHeight = newValue;

}

public String toString(){

return (super.toString()+"\nseat height is "+seatHeight);

}

}

public class Test{

public static void main(String args[]){

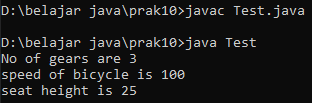
MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

}

1. Output



1. Analisis

Pada percobaan ini, kita akan membuat program dengan tujuan

pemanfaatan dari konstruktor inheritance. Inheritance atau pewarisan disini

berguna untuk menurunkan method atau atribut. Nantinya subclass akan

dapat menggunakan atribut dan method dari class parent nya. Pada program

diatas, class yang memiliki konstruktor, konstruktor nya tidak akan

diturunkan kepada subclassnya. Sehingga yang diturunkan hanya method dan

atribut yang memiliki akses protected dan public saja. Untuk program

selanjutnya, terdapat class dari Bicycle yang merupakan parent class dari

class Mountainbike. Penggunaannya sama saja dengan program-program

sebelumnya, dimana atribut dan method yang ada pada class Bicycle akan

bisa di akses oleh subclass nya yaitu class Mountainbike.

**LATIHAN SOAL**

Latihan 1 - Mengimplementasikan UML class diagram dalam program untuk package perbankan

1. Listing Program

package perbankan;

class Tabungan{

private int saldo;

public Tabungan(){

}

public int getSaldo(){

return this.saldo;

}

public void setSaldo(int uang){

this.saldo=uang;

}

}

public class PengambilanUang extends Tabungan{

private int proteksi;

public PengambilanUang(int saldo){

}

public PengambilanUang(int saldo, int tingkatBunga){

super.setSaldo(saldo - tingkatBunga);

this.proteksi=tingkatBunga;

}

public int getSaldo(){

return super.getSaldo();

}

public boolean ambilUang(int jumlah){

if(jumlah <= super.getSaldo()){

super.setSaldo(super.getSaldo() - jumlah);

return true;

}

else{

return false;

}

}

}

import perbankan.\*;

public class TesTugas{

public static void main(String[] args){

PengambilanUang tabungan = new

PengambilanUang(5000,1000);

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("-----------------");

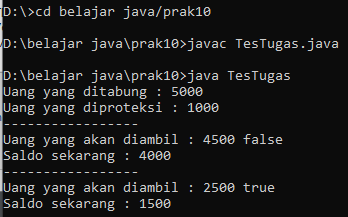
System.out.println("Uang yang akan diambil : 2500 "+tabungan.ambilUang(2500));

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

}

}

1. Output



1. Analisis

Dalam latihan ini terdapat Class Tabungan dan Class PengambilanUang. Class

Tabungan merupakan Parent Class dari Class PengambilanUang. Dalam class

PengambilanUang terdapat variabel Proteksi yang berguna untuk menyimpan nilai

Proteksi dari tabungan. Jadi ketika Kostruktor PengambilanUang(saldo,tingkatbunga)

dijalankan ada menggunakan 2 parameter yaitu saldo dan tngkatbunga yang nantinya

saldo yang ada di superclass akan diset dengan saldo-tingkatbunga. Dan tingkatbungan

akan disimpan sebagai proteksi. Dan dalam method ambilUang(jumlah) terdapat

percabangan ketika jumlah yang akan diambil kurang dari saldo maka akan bernilai Tru

dan uang bisa diambil dan ketika tidak akan mengembalikan nilai False dan Transaksi

Gagal.Dan kemudia saya coba di Class tesLatihan untuk melihat hasilnya.

**TUGAS**

Tugas 1

1. Listing Program

package rumus;

import java.lang.Math;

class Lingkaran{

private double r;

public Lingkaran(){

}

public Lingkaran(double nilai){

this.r=nilai;

}

public void setR(double nilai){

this.r=nilai;

}

public double getR(){

return this.r;

}

public double getDiameter(){

return (2 \* this.r);

}

public double getKeliling(){

return (2 \* 3.14 \* this.r);

}

public double getLuas(){

return (3.14 \* Math.pow(this.r,2));

}

public String toString(){

return("Jari - jari : " + this.r);

}

}

public class Silinder extends Lingkaran{

private double tinggi;

public Silinder(){

}

public Silinder(double r, double t){

super(r);

this.tinggi=t;

}

public void setTinggi(double tinggi){

this.tinggi=tinggi;

}

public double getTinggi(){

return this.tinggi;

}

public double getVolume(){

return (3.14 \* Math.pow(super.getR(),2) \* this.tinggi);

}

public String toString(){

return (super.toString() + "\nTinggi : " + this.tinggi);

}

public double getLuas(){

return (2 \* 3.14 \* super.getR() \* this.tinggi);

}

}

package rumus;

import java.lang.Math;

class Lingkaran{

private double r;

public Lingkaran(){

}

public Lingkaran(double nilai){

this.r=nilai;

}

public void setR(double nilai){

this.r=nilai;

}

public double getR(){

return this.r;

}

public double getDiameter(){

return (2 \* this.r);

}

public double getKeliling(){

return (2 \* 3.14 \* this.r);

}

public double getLuas(){

return (3.14 \* Math.pow(this.r,2));

}

public String toString(){

return("Jari - jari : " + this.r);

}

}

public class Kerucut extends Lingkaran{

private double tinggi;

private double s;

public Kerucut(){

}

public Kerucut(double r, double tinggi){

super(r);

this.tinggi=tinggi;

}

public void setTinggi(double tinggi){

this.tinggi=tinggi;

}

public double getTinggi(){

return this.tinggi;

}

public double getVolume(){

return ((3.14 \* Math.pow(super.getR(),2) \*this.tinggi)/3);

}

public String toString(){

return (super.toString() + "\nTinggi : " + this.tinggi);

}

public double getLuas(){

this.s = Math.sqrt(Math.pow(super.getR(),2) \*Math.pow(this.tinggi,2));

return (3.14 \* super.getR() \* this.s);

}

}

package rumus;

import java.lang.Math;

public class Lingkaran{

private double r;

public Lingkaran(){

}

public Lingkaran(double nilai){

this.r=nilai;

}

public void setR(double nilai){

this.r=nilai;

}

public double getR(){

return this.r;

}

public double getDiameter(){

return (2 \* this.r);

}

public double getKeliling(){

return (2 \* 3.14 \* this.r);

}

public double getLuas(){

return (3.14 \* Math.pow(this.r,2));

}

public String toString(){

return("Jari - jari : " + this.r);

}

}

import rumus.\*;

public class TesLingkaran{

public static void main(String[] args){

Lingkaran li = new Lingkaran(14.0);

System.out.println("Informasi Lingkaran : \n" +li.toString());

System.out.println("Diameter : " + li.getDiameter());

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

System.out.println("Keliling : " + li.getKeliling());

System.out.println("Luas : " + li.getLuas());

}

}

import rumus.\*;

public class TesKerucut{

public static void main(String[] args){

Kerucut kc = new Kerucut(14.0, 9.0);

System.out.println("Informasi Kerucut : " +kc.toString());

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

System.out.println("Volume : " + kc.getVolume());

System.out.println("Luas Selimut : " + kc.getLuas());

}

}

import rumus.\*;

public class TesSilinder{

public static void main(String[] args){

Silinder si = new Silinder(14.0, 9.0);

System.out.println("Informasi Silinder : \n" +si.toString());

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

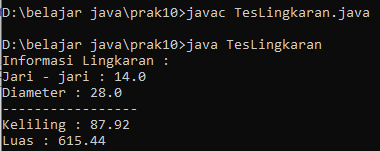
System.out.println("Volume : " + si.getVolume());

System.out.println("Luas Selimut : " + si.getLuas());

}

}

1. Output



1. Analisis

Terdapat class Kerucut dan Silinder yang mana keduanya menginherit class Lingkaran1. Di

Kerucut dan Silinder ini terdapat penambahan variabel yaitu tinggi dan method volume(). Juga terdapat

rebuild method dari Lingkaran yaitu toString() dan getLuas(). Luas permukaan Kerucut didapat dari

phi\*r\*(r+s) dengan s didapat dari akar dari penjumlahan r kuadrat dan t kuadrat. Volume Kerucut

didapat dari rumus phi\*r\*r\*t/3. Luas permukaan Silinder didapat dari 2\*phi\*r\*(r+t). Volume Silinder

didapat dari rumus phi\*r\*r\*t.