JOBSHEET 6

INHERITANCE (PEWARISAN)

# 1. KOMPETENSI

1. Memahami konsep dasar inheritance atau pewarisan.
2. Mampu membuat suatu subclass dari suatu superclass tertentu.

# 2. PENDAHULUAN

Inheritance atau pewarisan sifat merupakan suatu cara untuk menurunkan suatu class yang lebih umum menjadi suatu class yang lebih spesifik. Inheritance adalah salah satu ciri utama suatu bahasa program yang berorientasi pada objek. Inti dari pewarisan adalah sifat reusable dari konsep object oriented. Setiap subclass akan “mewarisi” sifat dari superclass selama bersifat protected ataupun public.

Dalam inheritance terdapat dua istilah yang sering digunakan. Kelas yang menurunkan disebut kelas dasar (base class/super class), sedangkan kelas yang diturunkan disebut kelas turunan (derived class/sub class/child class) . Di dalam Java untuk mendeklarasikan suatu class sebagai subclass dilakukan dengan cara menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parent class---nya. Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan perluasan class. Berikut adalah contoh deklarasi inheritance.

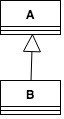
|  |
| --- |
| public class B extends A {  ...  } |

Contoh diatas memberitahukan kompiler Java bahwa kita ingin meng---extend class A ke class B. Dengan kata lain, class B adalah subclass (class turunan) dari class A, sedangkan class A adalah parent class dari class B.

Karakteristik pada super class akan dimiliki juga oleh subclassnya. Terdapat 3 bentuk pewarisan: single inheritance, multilevel inheritance, dan multiple inheritance. Namun yang akan dibahas pada jobsheet ini adalah single inheritance dan multilevel inheritance.

1. Single Inheritance

Single inheritance adalah Suatu class yang hanya mempunyai satu parent class. Contoh:



Gambar 1. Contoh Single Inheritance

Berdasarkan Gambar 1 dapat diketahui bahwa class B merupakan subclass yang mempunyai satu parent yaitu class A sehingga disebut single inheritance.

1. Multilevel Inheritance

Multilevel inheritance adalah Suatu subclass bisa menjadi superclass bagi class yang lain.

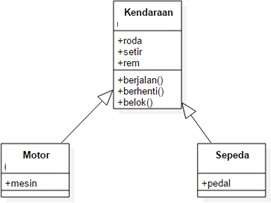
Contoh:



Gambar 2. Contoh Multilevel Inheritance

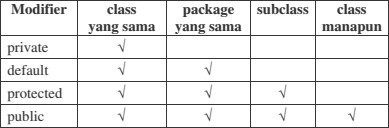
Berdasarkan Gambar 2 diatas dapat dilihat bahwa class B merupakan subclass dari class A, sehingga dalam hal ini class A adalah superclass dan class B adalah subclass. Kemudian class B yang awalmya merupakan subclass mempunyai subclass lagi yaitu class C sehingga class B menjadi superclass dari class C, begitu juga seterunya jika class C memilki subclass lagi.

Pada class diagram, pewarisan digambarkan dengan sebuah garis tegas, dengan segitiga di ujungnya. Class yang dekat pada segitiga merupakan superclass, sedangkan class yang jauh dari segitiga merupakan subclass. Untuk membentuk sebuah subclass, keyword “extends” digunakan (lihat contoh pada sesi “Implementasi Pewarisan”). Berikut ini adalah contoh class diagram dari pewarisan:



Gambar 3 Contoh class diagram dalam inheritance

Suatu parent class dapat tidak mewariskan sebagian member---nya kepada subclass---nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam tabel berikut ini:



Kata kunci super dipakai untuk merujuk pada member dari parent class. Sebagaimana kata kunci this yang dipakai untuk merujuk pada member dari class itu sendiri. Format penulisannya adalah sebagai berikut:

* super.namaAtribut

Merujuk/mengakses atribut dari parent class /superclass

* super.namaMethod()

Merujuk/memanggil method dari parent class /superclass

* super()

Merujuk / memanggil konstruktor parent class /superclass Hanya bisa digunakan dibaris pertama dalam kontruktor.

* super(parameter1, parameter2,dst)

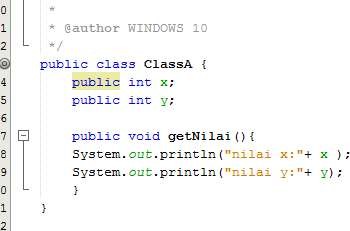
Merujuk / memanggil konstruktor berparamter dari superklas

Ketika mmebuat objek dari subclass, pada saat itu juga objek pada superclass juga akan terbentuk. Dengan katalain, ketika kontruktor subclass dijalankan untuk membuat objek, saat itu juga kontruktor superclass akan berjalan. Jadi di setiap konstruktor subclass, pada baris pertama konstruktor subclass tersebut akan dipanggil konstruktor superclass. Sebelum subclass menjalankan kontruktornya sendiri, subclass akan menjalankan kontruktor superclass terlebih dahulu.

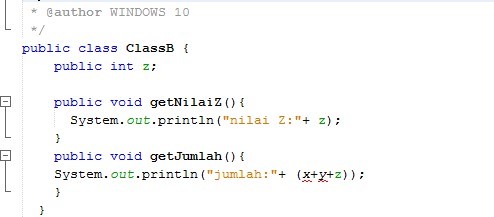
# 3. PERCOBAAN 1 (extends)

## A. TAHAPAN PERCOBAAN

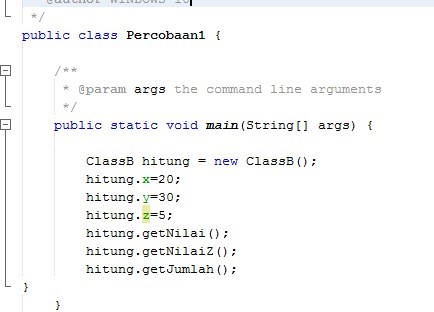
1. Buatlah sebuah class parent/superclass dengan nama ClassA.java



1. Buatlah sebuah class anak/subclass dengan nama ClassB.java



1. Buatlah class Percobaan1.java untuk menjalankan program diatas!



1. Jalankan program diatas, kemudian amati apa yang terjadi!

## B. PERTANYAAN

1. Pada percobaan 1 diatas program yang dijalankan terjadi error, kemudian perbaiki sehingga program tersebut bisa dijalankan dan tidak error!

* Dengan menambahkan extends pada ClassB.

package percobaan1;

public class ClassA {

    public int **x**, **y**;

    public void getNilai() {

        System.**out**.println("Nilai x = " + **x**);

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

    }

}

package percobaan1;

public class ClassB extends ClassA {

    public int **z**;

    public void getNilaiZ() {

        System.**out**.println("Nilai z = " + **z**);

    }

    public void getJumlah() {

        System.**out**.println("Jumlah = " + (**x** + **y** + **z**));

    }

}

package percobaan1;

public class Main {

    public static void main(String[] args) {

        ClassB hitung = new ClassB();

        hitung.**x** = 20;

        hitung.**y** = 30;

        hitung.**z** = 5;

        hitung.getNilai();

        hitung.getNilaiZ();

        hitung.getJumlah();

    }

}

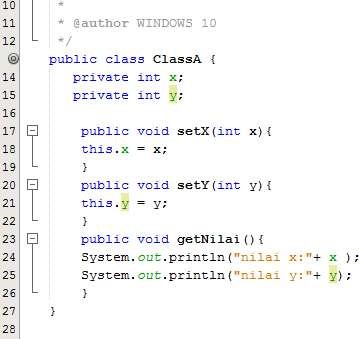
1. Jelaskan apa penyebab program pada percobaan 1 ketika dijalankan terdapat error!

* Karena pada ClassB belum meng-extends ClassA yang menjadikan error. Sehingga pada pengambilan fungsi ClassB di main terjadi error karena hal tersbut.

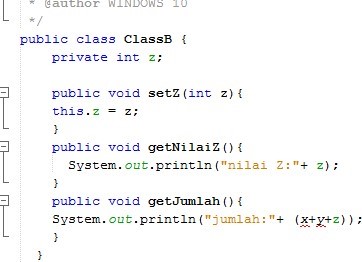
# 4. PERCOBAAN 2 (Hak Akses)

## A. TAHAPAN PERCOBAAN

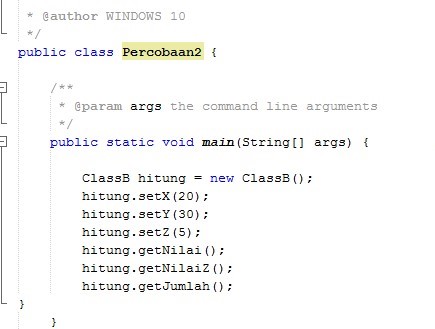
1. Buatlah sebuah class parent/superclass dengan nama ClassA.java



1. Buatlah sebuah class anak/subclass dengan nama ClassB.java



1. Buatlah class Percobaan2.java untuk menjalankan program diatas!



1. Jalankan program diatas, kemudian amati apa yang terjadi!

## B. PERTANYAAN

1. Pada percobaan 2 diatas program yang dijalankan terjadi error, kemudian perbaiki sehingga program tersebut bisa dijalankan dan tidak error!

* Dengan menambahkan extends pada ClassB. Kemudian memberi fungsi getter di ClassA agar bisa mengambil nilai atribut ClassA di ClassB. Dan yang terakhir menyesuaikan program di class Main.

package percobaan2;

public class ClassA {

    private int **x**, **y**;

    public void getNilai() {

        System.**out**.println("Nilai x = " + **x**);

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

    }

    public void setX(int x) {

        this.**x** = x;

    }

    public void setY(int y) {

        this.**y** = y;

    }

    public int getX() {

        return **x**;

    }

    public int getY() {

        return **y**;

    }

}

package percobaan2;

public class ClassA {

    private int **x**, **y**;

    public void getNilai() {

        System.**out**.println("Nilai x = " + **x**);

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

    }

    public void setX(int x) {

        this.**x** = x;

    }

    public void setY(int y) {

        this.**y** = y;

    }

    public int getX() {

        return **x**;

    }

    public int getY() {

        return **y**;

    }

}

package percobaan2;

public class Main {

    public static void main(String[] args) {

        ClassB hitung = new ClassB();

        hitung.setX(20);

        hitung.setY(30);

        hitung.setNilaiZ(5);

        hitung.getNilai();

        hitung.getNilaiZ();

        hitung.getJumlah();

    }

}

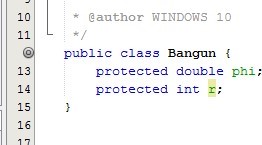
1. Jelaskan apa penyebab program pada percobaan 1 ketika dijalankan terdapat error!

* Karena pada ClassB belum meng-extends ClassA yang menjadikan error. Sehingga pada pengambilan fungsi ClassB di main terjadi error karena hal tersbut.

# 5. PERCOBAAN 3 (Super)

## A. TAHAPAN PERCOBAAN

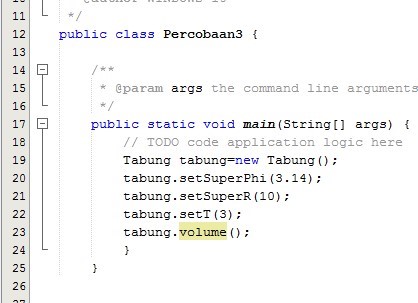
1. Buatlah sebuah class parent/superclass dengan nama Bangun.java



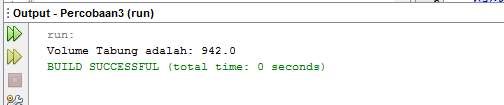
1. Buatlah sebuah class anak/subclass dengan nama Tabung.java



1. Buatlah class Percobaan3.java untuk menjalankan program diatas!

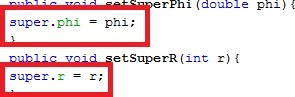


1. Jalankan program diatas!



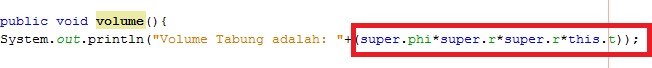
## B. PERTANYAAN

1. Jelaskan fungsi “super” pada potongan program berikut di class Tabung!



* Untuk menggunakan atribut dari class yang digunakan sebagai superclass.

1. Jelaskan fungsi “super” dan “this” pada potongan program berikut di class Tabung!



* “super” digunakan untuk mengambil nilai dari class yang digunakan sebagai cuperclass. Sedangkna “this” digunakan untuk mengambil nilai dari atribut yang ada di class di tempat “this” digunakan.

1. Jelaskan mengapa pada class Tabung tidak dideklarasikan atribut “phi” dan “r” tetapi class tersebut dapat mengakses atribut tersebut!

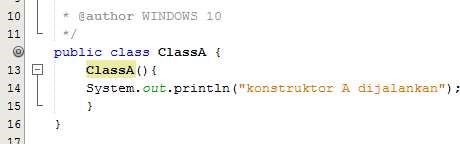
* Karena telah meng-extends dari class Bangun yang mana dalam class tersebut terdapat atribut dengan nama phi dan r.

# 6. PERCOBAAN 4 (super contsructor)

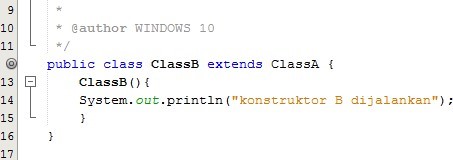
## A. TAHAPAN PERCOBAAN

1. Buatlah tiga file dengan nama ClassA.java , ClassB.java , dan ClassC.java, seperti pada kode program dibawah ini!

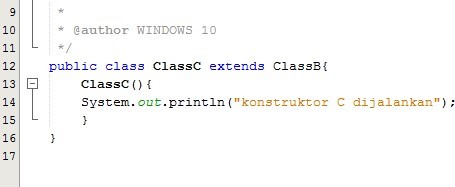
ClassA.java



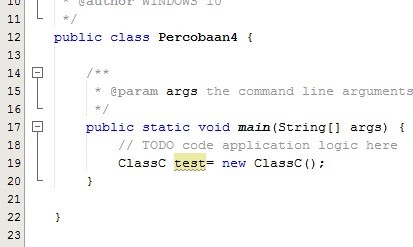
ClassB.java



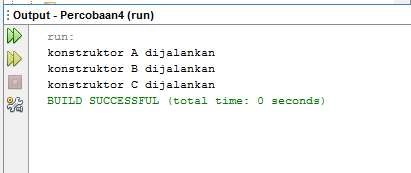
ClassC.java



1. Buatlah class Percobaan4.java untuk menjalankan program diatas!



1. Jalankan program kemudian amati apa yang terjadi!

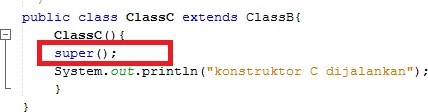


## B. PERTANYAAN

1. Pada percobaan 4 sebutkan mana class yang termasuk superclass dan subclass, kemudian jelaskan alasannya!

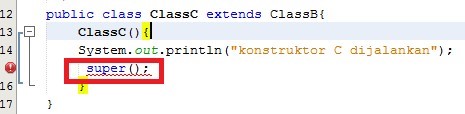
* Class yang menjadi superclass sekaligus sub-class adalaa ClassB.

1. Ubahlah isi konstruktor default ClassC seperti berikut:



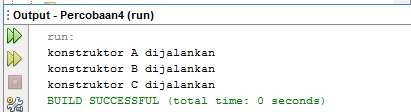
Tambahkan kata super() di baris Pertaman dalam konstruktor defaultnya. Coba jalankan kembali class Percobaan4 dan terlihat tidak ada perbedaan dari hasil outputnya!

1. Ublah isi konstruktor default ClassC seperti berikut:



Ketika mengubah posisi super() dibaris kedua dalam kontruktor defaultnya dan terlihat ada error. Kemudian kembalikan super() kebaris pertama seperti sebelumnya, maka errornya akan hilang.

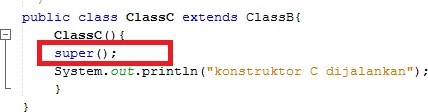
Perhatikan hasil keluaran ketika class Percobaan4 dijalankan. Kenapa bisa tampil output seperti berikut pada saat instansiasi objek test dari class ClassC



Jelaskan bagaimana urutan proses jalannya konstruktor saat objek test dibuat!

* Dimulai dengan menjalankan ClassC, kemudian menjalankan ClassB, dan yang terakhir menjalankan ClassA.

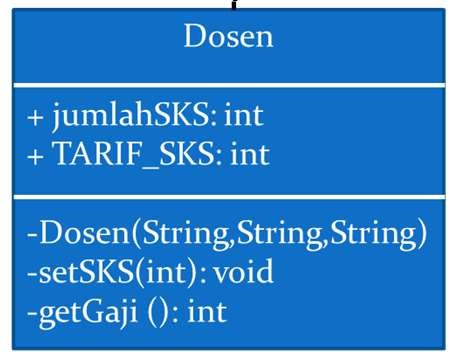
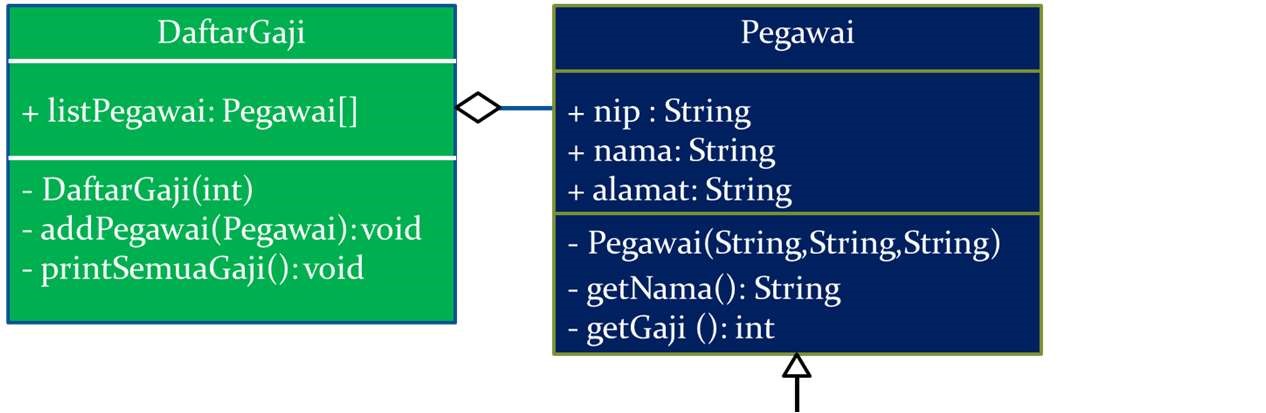
1. Apakah fungsi super() pada potongan program dibawah ini di ClassC!



* Untuk mengambil konstruktor dari superclassnya tanpa menulis ulang yang mana fungsi nya adalah untuk mengambil pendeklarasian dari atribut ClassC agar sama dengan ClassB jika ada.

# 9. TUGAS

1. Buatlah sebuah program dengan konsep pewarisan seperti pada class diagram berikut ini. Kemudian buatlah instansiasi objek untuk menampilkan data nama pegawai dan gaji yang didapatkannya.



public class DaftarGaji {

    private Pegawai **pegawai**[];

    public DaftarGaji(int jumlahList) {

        this.**pegawai** = new Pegawai[jumlahList];

    }

    public void addPegawai(Pegawai pegawai) {

        for (int i = 0; i < this.**pegawai**.**length**; i++) {

            if (this.**pegawai**[i] == null) {

                this.**pegawai**[i] = pegawai;

                break;

            }

        }

    }

    public void printGaji() {

        for (int i = 0; i < this.**pegawai**.**length**; i++) {

            if (this.**pegawai**[i] != null) {

                System.**out**.println(this.**pegawai**[i].getNama() + "\t:" + this.**pegawai**[i].getGaji());

            }

        }

    }

}

public class Pegawai {

    private String **nip**, **nama**, **alamat**;

    private int **gaji**;

    public Pegawai(String nip, String nama, String alamat) {

        this.**nip** = nip;

        this.**nama** = nama;

        this.**alamat** = alamat;

    }

    public String getNip() {

        return **nip**;

    }

    public String getNama() {

        return **nama**;

    }

    public String getAlamat() {

        return **alamat**;

    }

    public int getGaji() {

        return **gaji**;

    }

    public void setGaji(int gaji) {

        this.**gaji** = gaji;

    }

    public void print() {

        System.**out**.println("NIP\t: " + getNip());

        System.**out**.println("Nama\t: " + getNama());

        System.**out**.println("Alamat\t: " + getAlamat());

        System.**out**.println("Gaji\t: " + getGaji());

        System.**out**.println();

    }

}

public class Dosen extends Pegawai {

    private int **jumlahSKS**, **tarifSKS**;

    public Dosen(String nip, String nama, String alamat) {

        super(nip, nama, alamat);

    }

    public int getTarif() {

        return **tarifSKS**;

    }

    public void setTarif(int tarifSKS) {

        this.**tarifSKS** = tarifSKS;

    }

    public int getSKS() {

        return **jumlahSKS**;

    }

    public void setSKS(int sks) {

        this.**tarifSKS** = sks;

    }

public void print() {

        System.**out**.println("SKS\t: " + getSKS());

        System.**out**.println("Tarif\t: " + getTarif());

        super.print();

        System.**out**.println();

    }

}

public class Main {

    public static void main(String[] args) {

        Dosen bambang = new Dosen("12345", "Bambang", "Bandung");

        Dosen wahyu = new Dosen("23456", "Wahyu", "Wonogiri");

        Dosen supratman = new Dosen("34567", "Supratman", "Surabaya");

        DaftarGaji daftargaji = new DaftarGaji(5);

        daftargaji.addPegawai(bambang);

        daftargaji.addPegawai(wahyu);

        daftargaji.addPegawai(supratman);

        bambang.setGaji(2000000);

        wahyu.setGaji(3000000);

        supratman.setGaji(4000000);

bambang.setSKS(2);

        wahyu.setSKS(3);

        supratman.setSKS(4);

        bambang.print();

        wahyu.print();

        supratman.print();

        daftargaji.printGaji();

    }

}

