

Praktikum PBO

Inheritance Java

Nama : Billy Achmad Bachrudin

NIM : 20210040100

Kelas : TI21F

2. Berikan analisa setiap percobaan dalam bentuk File teks pdf dan upload juga ke github praktikum-inheritance

a. Percobaan 1:

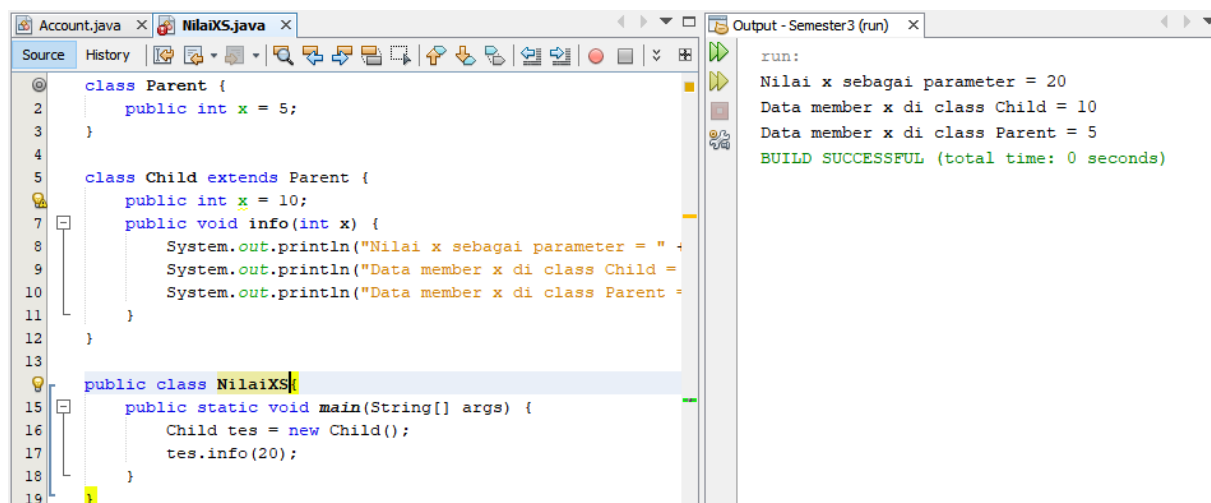
Percobaan berikut ini menunjukkan penggunaan kata kunci “super”.

```
class Parent {
    public int x = 5;
}

class Child extends Parent {
    public int x = 10;
    public void Info(int x) {
        System.out.println("Nilai x sebagai parameter = " + x);
        System.out.println("Data member x di class Child = " + this.x);
        System.out.println("Data member x di class Parent = " +
            super.x);
    }
}

public class NilaiX {
    public static void main(String args[]) {
        Child tes = new Child();
        tes.Info(20);
    }
}
```

Pada percobaan 1 objek Child tes memanggil fungsi info, yang kemudian akan menampilkan output dari 3 fungsi x yaitu 20, 10, dan 5. Nilai x yang ditampilkan berbeda walaupun fungsi yang dipanggil sama-sama x, x yang pertama merupakan nilai dari parameter yang bernilai 20, this.x merupakan variabel x kedua yang menempel pada objek maka bernilai 10, super.x bernilai 5 yang mengambil nilai dari parent class.



The screenshot shows an IDE with two tabs: 'Account.java' and 'NilaiXS.java'. The 'NilaiXS.java' tab is active, displaying the same Java code as shown in the previous block. The code defines a 'Parent' class with a variable 'x' set to 5, a 'Child' class that extends 'Parent' with its own 'x' set to 10 and an 'Info' method that prints the parameter 'x', 'this.x', and 'super.x', and a 'NilaiXS' class with a 'main' method that creates a 'Child' object and calls its 'Info' method with the argument 20. To the right of the code editor is an 'Output - Semester3 (run)' window. It shows the execution output: 'run:', 'Nilai x sebagai parameter = 20', 'Data member x di class Child = 10', 'Data member x di class Parent = 5', and 'BUILD SUCCESSFUL (total time: 0 seconds)'.

b. Percobaan 2 :

Percobaan berikut ini menunjukkan penggunaan kontrol akses terhadap atribut parent class. Mengapa terjadi error, dan bagaimana solusinya?

```
public class Pegawai {  
    private String nama;  
    public double gaji;  
}  
  
public class Manajer extends Pegawai {  
    public String departemen;  
  
    public void IsiData(String n, String d) {  
        nama=n;  
        departemen=d;  
    }  
}
```

Masalah :

- Error 1 : Terjadi karena class Pegawai dinyatakan sebagai public, public seharusnya digunakan ketika kelas pegawai diletakan pada filenya itu sendiri.
- Error 2: Terjadi karena String nama di jadikan private, maka saat dipanggil akan terjadi error karena tidak bisa memanggil fungsi tersebut.

Solusi :

- Mengganti modifier atribut nama pada class Pegawai menjadi public agar bisa diakses oleh class Manajer, kemudian menghapus public pada class Pegawai menjadi hanya class Pegawai saja.
- Menghapus private pada string nama.

c. Percobaan 3

Percobaan berikut ini menunjukkan penggunaan konstruktor yang tidak diwariskan. Mengapa terjadi error, dan bagaimana solusinya?

```
public class Parent {  
    // kosong  
}  
  
public class Child extends Parent {  
    int x;  
    public Child() {  
        x = 5;  
    }  
}
```

Tidak terjadi error karena konstruktor tidak di wariskan, tetapi terjadi error saat menjalankan kode karena class parent ataupun child tidak mempunyai main method.

d. Percobaan 4

Percobaan berikut ini menunjukkan penggunaan kelas Employee dan subkelas Manager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji kelas Manager.

```
class Employee {
    private static final double BASE_SALARY = 15000.00;
    private String Name = "";
    private double Salary = 0.0;
    private Date birthDate;

    public Employee() {}
    public Employee(String name, double salary, Date DoB) {
        this.Name=name;
        this.Salary=salary;
        this.birthDate=DoB;
    }
    public Employee(String name,double salary) {
        this(name,salary,null);
    }
    public Employee(String name, Date DoB) {
        this(name,BASE_SALARY,DoB);
    }
    public Employee(String name){
        this(name,BASE_SALARY);
    }

    public String GetName(){ return Name;}
    public double GetSalary(){ return Salary; }
}

class Manager extends Employee {

    //tambahan attribrute untuk kelas manager
    private String department;

    public Manager(String name,double salary,String dept) {
        super(name,salary);
        department=dept;
    }
    public Manager(String n,String dept){
        super(n);
        department=dept;
    }
    public Manager(String dept){
        super();
        department=dept;
    }
    public String GetDept(){
        return department;
    }
}

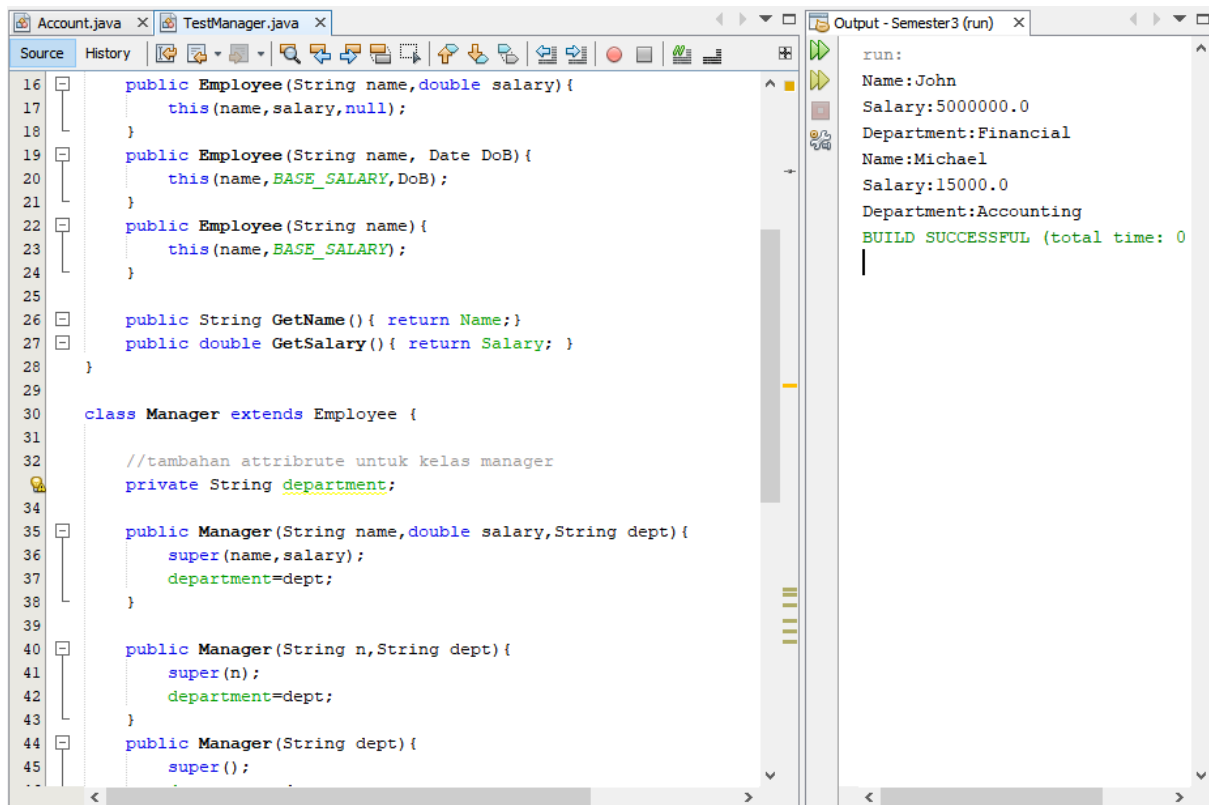
public class TestManager {

    public static void main(String[] args) {
        Manager Utama = new Manager("John",5000000,"Financial");
        System.out.println("Name:"+ Utama.GetName());
        System.out.println("Salary:"+ Utama.GetSalary());

        System.out.println("Department:"+ Utama.GetDept());

        Utama = new Manager("Michael","Accounting");
        System.out.println("Name:"+ Utama.GetName());
        System.out.println("Salary:"+ Utama.GetSalary());
        System.out.println("Department:"+ Utama.GetDept());
    }
}
```

Output :



The screenshot shows an IDE with two tabs: 'Account.java' and 'TestManager.java'. The 'TestManager.java' tab is active, displaying the following Java code:

```
16 public Employee(String name, double salary) {
17     this(name, salary, null);
18 }
19 public Employee(String name, Date DoB) {
20     this(name, BASE_SALARY, DoB);
21 }
22 public Employee(String name) {
23     this(name, BASE_SALARY);
24 }
25
26 public String GetName() { return Name; }
27 public double GetSalary() { return Salary; }
28 }
29
30 class Manager extends Employee {
31
32     //tambahan attribrute untuk kelas manager
33     private String department;
34
35     public Manager(String name, double salary, String dept) {
36         super(name, salary);
37         department = dept;
38     }
39
40     public Manager(String n, String dept) {
41         super(n);
42         department = dept;
43     }
44     public Manager(String dept) {
45         super();
46     }
47 }
```

The output window on the right, titled 'Output - Semester3 (run)', shows the following text:

```
run:
Name:John
Salary:5000000.0
Department:Financial
Name:Michael
Salary:15000.0
Department:Accounting
BUILD SUCCESSFUL (total time: 0
```

Analisis : Tidak ditemukan error pada kode yang menggunakan 3 parameter(nama, salary, Dept) dalam konstruktor, maupun 2 parameter(nama, Dept) dalam konstruktor

e. Percobaan 5

Percobaan berikut ini menunjukkan penggunaan kelas MoodyObject dengan subkelas HappyObject dan SadObject. Kelas MoodyTest digunakan untuk menguji kelas dan subkelas.

- SadObject berisi :

- sad, method untuk menampilkan pesan, tipe public

- HappyObject berisi :

- laugh, method untuk menampilkan pesan, tipe public

- MoodyObject berisi :

- getMood, memberi nilai mood sekarang, tipe public, return type string
- speak, menampilkan mood, tipe public.

```

public class MoodyObject {

    protected String getMood(){
        return "moody";
    }
    public void speak(){
        System.out.println("I am"+getMood());
    }
    void laugh() {}
    void cry() {}
}

public class SadObject extends MoodyObject{
    protected String getMood(){
        return "sad";
    }
    public void cry(){
        System.out.println("Hoo hoo");
    }
}

public class HappyObject extends MoodyObject{
    protected String getMood(){
        return "happy";
    }
    public void laugh(){
        System.out.println("Hahaha");
    }
}

public class MoodyTest {
    public static void main(String[] args) {

        MoodyObject m = new MoodyObject();

        //test perent class
        m.speak();

        //test inheritance class
        m = new HappyObject();
        m.speak();
        m.laugh();

        //test inheritance class
        m=new SadObject();
        m.speak();
        m.cry();
    }
}

```

Analisis: Tidak ada masalah dalam program ini, program dapat dijalankan dengan lancar. Program tersebut akan menjalankan kelas yang dibuat untuk menjalankan fungsi fungsinya.

f. Percobaan 6 :

Percobaan berikut ini menunjukkan penggunaan kelas A dan dengan subkelas B. Simpan kedua kelas ini dalam 2 file yang berbeda (A.java dan B.java) dan dalam satu package. Perhatikan proses pemanggilan konstruktor dan pemanggilan variabel

```

class A {
    String var_a = "Variabel A";
    String var_b = "Variabel B";
    String var_c = "Variabel C";
    String var_d = "Variabel D";

    A() {
        System.out.println("Konstruktor A dijalankan");
    }
}

class B extends A{
    B() {
        System.out.println("Konstruktor B dijalankan ");
        var_a = "Var_a dari class B";
        var_b = "Var_a dari class B";
    }

    public static void main(String args[]) {

        System.out.println("Objek A dibuat");
        A aa= new A();
        System.out.println("menampilkan nama variabel obyek aa");
        System.out.println(aa.var_a);
        System.out.println(aa.var_b);
        System.out.println(aa.var_c);
        System.out.println(aa.var_d);
        System.out.println("");

        System.out.println("Objek B dibuat");
        B bb= new B();
        System.out.println("menampilkan nama variabel obyek bb");
        System.out.println(bb.var_a);
        System.out.println(bb.var_b);
        System.out.println(bb.var_c);
        System.out.println(bb.var_d);
    }
}

```

- Seharusnya pada constructor B dalam attribute var_b="Var b dari class B" diganti menjadi var_b="Var_a dari class B". Dalam percobaan ini Class B dapat mengakses kelas A walaupun keduanya dalam dua file yang berbeda, berada di package yang sama dan menggunakan modifier default menyebabkan class B dapat mengakses class A .
- Saat objek B dijalankan akan menampilkan output "konstruktor A dijalankan" hal ini terjadi karena class B mewariskan class A dimana pada constructor class A menampilkan hal tersebut di dalam filenya.

g. Percobaan 7

Percobaan berikut ini menunjukkan penggunaan Inheritance dan Overriding method pada kelas Bapak dan subkelas Anak. Terjadi override pada method show_variabel. Perhatikan perubahan nilai pada variabel a, b, dan c.

Kemudian lakukan modifikasi pada method show_variabel() pada class Anak. Gunakan super untuk menampilkan nilai a dan b (memanfaatkan method yang sudah ada pada superclass)

```

class Bapak {
    int a;
    int b;

    void show_variabel(){
        System.out.println("Nilai a="+ a);
        System.out.println("Nilai b="+ b);
    }
}

class Anak extends Bapak{
    int c;
    void show_variabel(){
        System.out.println("Nilai a="+ a);
        System.out.println("Nilai b="+ b);
        System.out.println("Nilai c="+ c);
    }
}

public class InheritExample {

    public static void main(String[] args) {

        Bapak objectBapak = new Bapak();
        Anak objectAnak = new Anak();

        objectBapak.a=1;
        objectBapak.b=1;
        System.out.println("Object Bapak (Superclass):");

        objectBapak.show_variabel();
        objectAnak.c=5;
        System.out.println("Object Anak (Superclass dari Bapak):");
        objectAnak.show_variabel();
    }
}

```

Output Tanpa menggunakan super :

```

class Bapak {
    int a;
    int b;

    void show_variabel() {
        System.out.println("Nilai a="+a);
        System.out.println("Nilai b="+b);
    }
}

class Anak extends Bapak{
    int c;
    void show_variabel(){
        System.out.println("Nilai a="+a);
        System.out.println("Nilai b="+b);
        System.out.println("Nilai c="+c);
    }
}

public class InheritExample {
    public static void main(String[] args) {

        Bapak objectBapak=new Bapak();
        Anak objectAnak=new Anak();

        objectBapak.a=1;
        objectBapak.b=1;
        System.out.println("Object Bapak (Superclass):");

        objectBapak.show_variabel();
        objectAnak.c=5;
        System.out.println("Object Anak (Superclass dari Bapak):");
        objectAnak.show_variabel();
    }
}

```

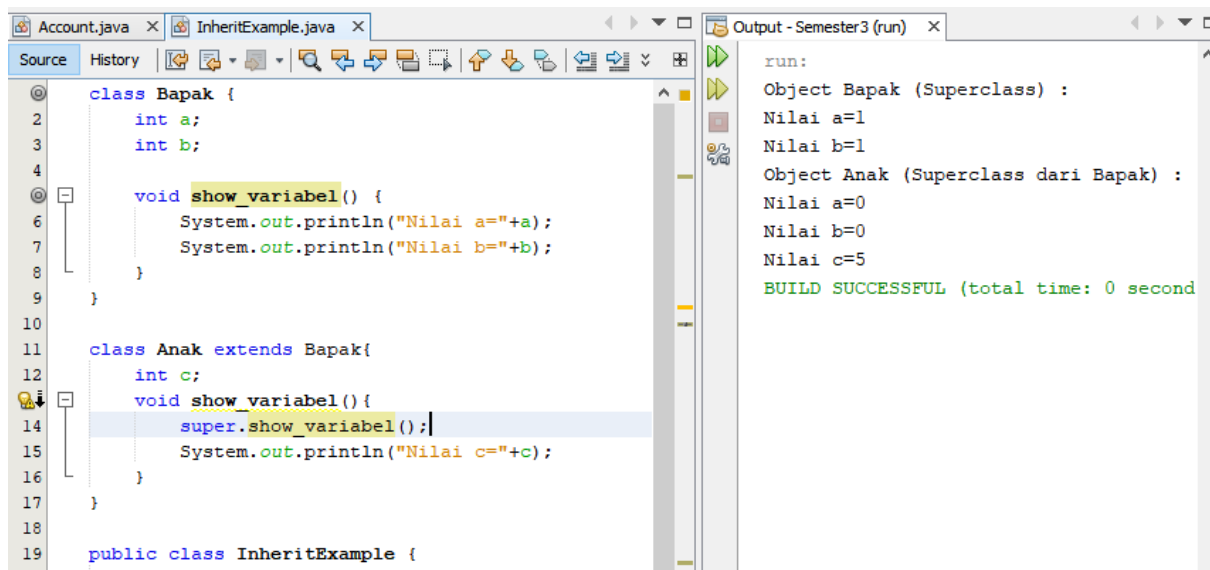
run:

```

Object Bapak (Superclass) :
Nilai a=1
Nilai b=1
Object Anak (Superclass dari Bapak) :
Nilai a=0
Nilai b=0
Nilai c=5
BUILD SUCCESSFUL (total time: 0 second

```

Dengan menggunakan super :



The screenshot shows an IDE with two windows. The left window, titled 'InheritExample.java', contains the following Java code:

```
class Bapak {
    int a;
    int b;

    void show_variabel() {
        System.out.println("Nilai a="+a);
        System.out.println("Nilai b="+b);
    }
}

class Anak extends Bapak{
    int c;
    void show_variabel(){
        super.show_variabel();
        System.out.println("Nilai c="+c);
    }
}

public class InheritExample {
```

The right window, titled 'Output - Semester3 (run)', shows the output of the program:

```
run:
Object Bapak (Superclass) :
Nilai a=1
Nilai b=1
Object Anak (Superclass dari Bapak) :
Nilai a=0
Nilai b=0
Nilai c=5
BUILD SUCCESSFUL (total time: 0 second)
```

Print nilai a dan b dalam fungsi show_variabel di class Anak diganti, jadi menggunakan super. Sehingga terjadi override fungsi show_variabel.

h. Percobaan 8

Percobaan berikut ini menunjukkan penggunaan overriding method pada kelas Parent dan subkelas Baby, saat dilakukan pemanggilan konstruktor superclass dengan menggunakan super.

```
public class Parent {
    String parentName;
    Parent() {}

    Parent(String parentName) {
        this.parentName = parentName;
        System.out.println("Konstruktor parent");
    }
}

class Baby extends Parent {
    String babyName;

    Baby(String babyName) {
        super();
        this.babyName = babyName;
        System.out.println("Konstruktor Baby");
        System.out.println(babyName);
    }

    public void Cry() {
        System.out.println("Owek owek");
    }
}
```

Analisis:

Pada kelas Baby mewariskan Parent. terdapat super() pada fungsi konstruktor yang akan menjadikan override kelas parentnya. Atribut babyName diset pada konstruktor. Ketika class Baby dibuat akan menampilkan print dari constructor kelas Parent juga.

