

Nama : SETIANA ANDIKA PUTRA
NIM : 20210040077 (TI21F)
Matkul : Pemrograman berorientasi objek

Tugas Praktikum

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);  
    }  
}
```

Analisa : disaat suatu obyek (tes) dibuat dan objek itu memanggil fungsi info maka output yang akan dihasilkan yaitu 20, 10, 5. Walaupun variabel yang sama yaitu “x” tetapi yang membedakan adalah pemanggil variabelnya.

Kalua misalkan cuman ‘x’ saja yang dipanggil maka x itu hanya nilai dari parameter (jika itu dalam sebuah fungsi) karena itu bernilai 20. Jika “this.x” maka nilai yang diambil adalah nilai x yang menempel pada objek itu karena itu bernilai 10. Sedang “super.x” dia aka mengambil nilai pada parent class karena itu

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;  
    }  
}
```

Analisa : jika terjadi error itu dikarenakan pada kelas Manajer dan dalam fungsi Isi Data memanggil variabel nama sedangkan dalam class Manajer tidak ada variabel nama.

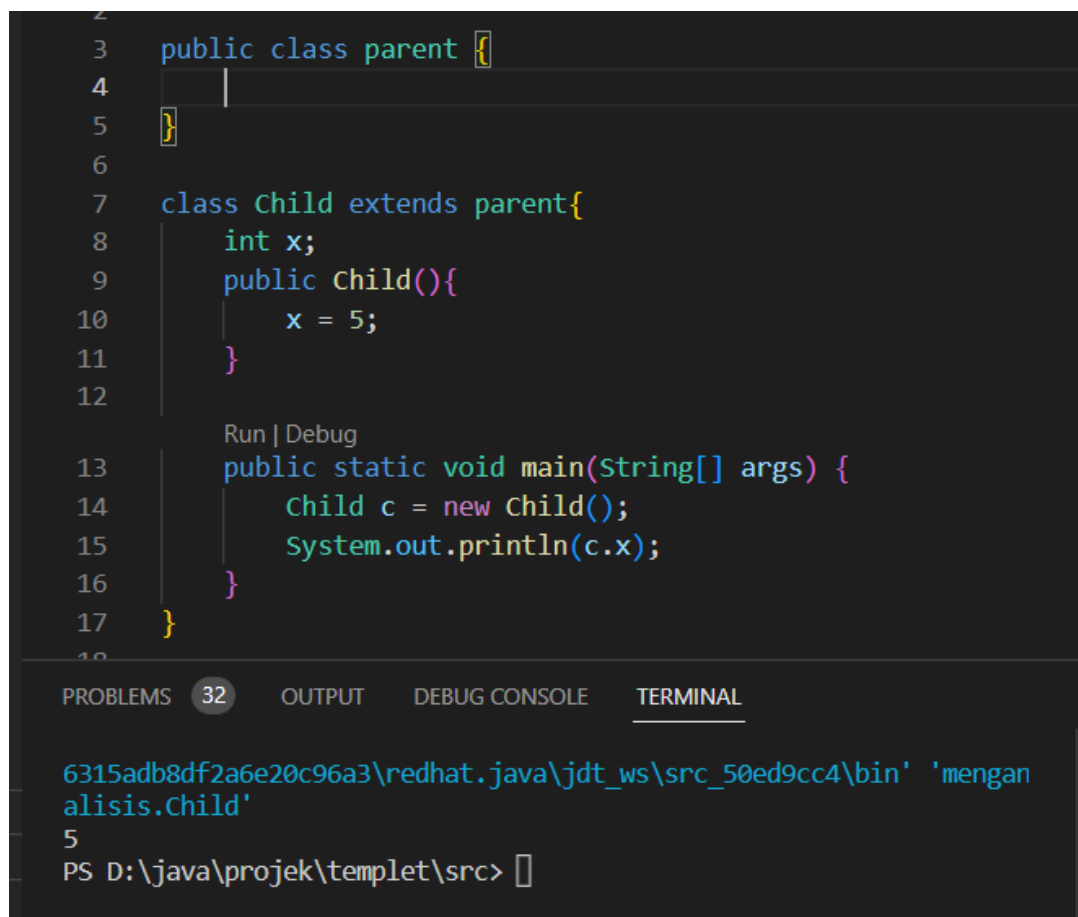
Solusinya adalah, jika atribut nama pada kelas pegawai access modifier diganti dari private menjadi public. Dan juga pemanggilan nama pada fungsi Isi Data diganti menjadi super.nama = n

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;  
    }  
}
```

Analisa : tidak terjadi error walaupun parent class tidak mempunyai konstruktor



The screenshot shows an IDE with a dark theme. The code editor displays the following Java code:

```
2  
3 public class parent {  
4  
5 }  
6  
7 class Child extends parent{  
8     int x;  
9     public Child(){  
10         x = 5;  
11     }  
12  
13     Run | Debug  
14     public static void main(String[] args) {  
15         Child c = new Child();  
16         System.out.println(c.x);  
17     }  
18 }
```

Below the code editor, there is a panel with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the output of the program:

```
6315adb8df2a6e20c96a3\redhat.java\jdt_ws\src_50ed9cc4\bin' 'mengan  
alisis.Child'  
5  
PS D:\java\projek\templet\src>
```

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());
    }
}

```

Analisa : Pemanggilan objek pertama menggunakan konstruktor dengan 3 parameter yaitu nama, salary dan Dept sedangkan objek kedua menggunakan konstruktor dengan 2 parameter yaitu nama dan Dept.

Percobaan 5:

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

- SadObject berisi :
 - o sad, method untuk menampilkan pesan, tipe public
- HappyObject berisi :
 - o laugh, method untuk menampilkan pesan, tipe public
- MoodyObject berisi :
 - o getMood, memberi nilai mood sekarang, tipe public, return type string
 - o 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();
}
}

```

Analisa : Tidak terdeteksi adanya masalah dalam program ini, program ini akan menjalankan kelas yang dibuat menjalankan fungsi fungsinya.

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);
    }
}

```

Analisa : karena terdapat 2 kelas yaitu kelas A sebagai parent class dan kelas B sebagai subclass dari A, maka sub class A akan mengganti nilai var_a dan var_b dari parent kelas nya.

Ketika objek B dibuat, constuktor A akan tetap dijalankan

Percobaan 7:

```
8
9     void show_variabel(){
10         System.out.println("Nilai a="+ a);
11         System.out.println("Nilai b="+ b);
12     }
13 }
14 class Anak extends Bapak{
15     int c;
16     int a;
17     int b;
18     void show_variabel(){
19         System.out.println("Nilai a="+ super.a);
20         System.out.println("Nilai b="+ super.b);
21         System.out.println("Nilai c="+ c);
22     }
23 }
24
25 public class InheritExamlle {
26
27     Run | Debug
28     public static void main(String[] args) {
29
30         Bapak objectBapak = new Bapak();
31         Anak objectAnak = new Anak();
32     }
33 }
34
35 PROBLEMS 33 OUTPUT DEBUG CONSOLE TERMINAL
36
37 PS D:\java\projek\templet\src> d::; cd 'd:\java\projek\templet\src
38 sers\LENOVO\AppData\Roaming\Code\User\workspaceStorage\ab12d5f1b32
39 6315adb8df2a6e20c96a3\redhat.java\jdt_ws\src_50ed9cc4\bin' 'mengan
40 alisis.InheritExamlle'
41 Object Bapak (Superclass):
42 Nilai a=1
43 Nilai b=3
44 Object Anak (superclass dari Bapak) :
45 Nilai a=0
46 Nilai b=0
47 Nilai c=80
48 PS D:\java\projek\templet\src>
```

Analisa : Walaupun sudah menggunakan super pada kelas anak untuk mengakses nilai dari parent kelas, nilai a dan b dari kelas anak akan tetap 0 karena pada dasarnya blueprint nya bernilai 0. Jadi objek Anak tidak akan melakukan "override" pada objek Bapak, selama dalam bentuk Objek.

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");
    }
}

```

Analisa : di kelas ini (Baby) menurunkan Parent. terdapat super() pada fungsi konstruktor yang akan mengoveride kelas parentnya. this.babyName = babyName untuk passing nilai babyName pada objek dengan parameter konstruktor babyName.