

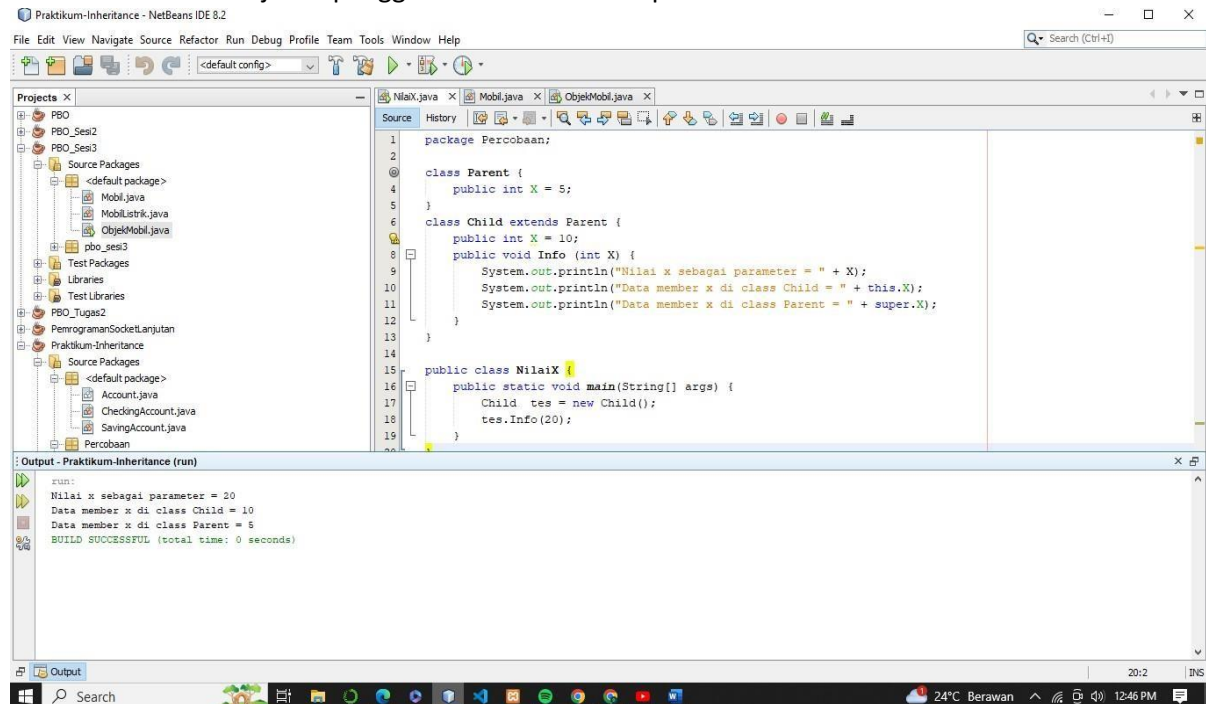
Nama: DWI ADI PERMANA

Kelas: TI22J

Matkul: Pemrograman Berorientasi Objek

### Percobaan 1 :

Percobaan ini menunjukkan penggunaan kata kunci "super".



The screenshot shows the NetBeans IDE interface. The main editor displays the following Java code:

```
1 package Percobaan;
2
3 class Parent {
4     public int X = 5;
5 }
6 class Child extends Parent {
7     public int X = 10;
8     public void Info (int X) {
9         System.out.println("Nilai x sebagai parameter = " + X);
10        System.out.println("Data member x di class Child = " + this.X);
11        System.out.println("Data member x di class Parent = " + super.X);
12    }
13 }
14
15 public class NilaiX {
16     public static void main(String[] args) {
17         Child tes = new Child();
18         tes.Info(20);
19     }
20 }
```

The output window at the bottom shows the following text:

```
run:
Nilai x sebagai parameter = 20
Data member x di class Child = 10
Data member x di class Parent = 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Hasil Analisa :

Pada percobaan 1 class `Child` mewariskan class Parent. Fungsi tes.info(20) akan menset nilai x menjadi 20 yang terdapat pada class `Child` dengan nama objek test. Hasil output akan menunjukkan nilai x sebagai parameter = 20. Data member x di class `Child` = 10 karena di class `Child` sudah didefinisikan x menjadi 10. Lalu Data member x di class `Parent` = 5. **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;
    }
}

```

#### Hasil Analisa :

Error yang pertama terjadi karena class **'Manajer'** dinyatakan sebagai public seharusnya public digunakan ketika kelas **'Manajer'** diletakkan pada filenya sendiri. Error kedua terjadi karena fungsi IsiData di class **'Manajer'** memanggil variabel nama dari class **'Pegawai'**, akan tetapi variabel nama diclass **'Pegawai'** di private sehingga terjadi error.

Solusinya adalah dengan mengubah modifier atribut nama pada class **'Pegawai'** menjadi public agar bisa diakses oleh class **'Manajer'**, lalu menghapus public pada class **'Manajer'** menjadi class **'Manajer'** saja.

#### Percobaan 3 :

Percobaan berikut ini menunjukkan penggunaan konstruktor yang tidak diwariskan. Mengapa

```

public class Parent {
    // kosong
}

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

```

terjadi error, dan bagaimana solusinya?

#### Hasil Analisa :

Percobaan tersebut menghasilkan error karena konstruktor di kelas **'Child'** mencoba mendefinisikan variabel lokal **'x'** dengan nilai 5, yang bukan merupakan inisialisasi variabel instance **'x'** dari kelas tersebut. Sebagai hasilnya, variabel instance **'x'** dari kelas **'Child'** tidak pernah diinisialisasi, yang dapat menyebabkan masalah ketika mencoba mengaksesnya. Solusinya adalah dengan menghapus deklarasi variabel lokal **int x = 5;** di dalam konstruktor **'Child'**, sehingga konstruktor dapat menggunakan dan menginisialisasi variabel instance **x** yang diwarisi dari kelas **'Parent'**.

#### 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 {
    //tambahkan attribute 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());
    }
}
```

### Hasil Analisa :

Program akan error bila tidak mengimport terlebih dahulu library **import java.util.Date;**, selebihnya kode program berjalan dengan lancar. **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 :

- 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();
}
```

### Hasil Analisa :

Terdapat error pada program dikarenakan setiap class dinyatakan public. Solusinya adalah menghapus setiap class yang dinyatakan public dan hanya menyisakan class **MoodyObject** dengan modifier public. Selebihnya program berjalan sesuai dengan fungsi fungsi yg telah dibuat.

### 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);
    }
}

```

### Hasil Analisa :

Terdapat dua class yaitu kelas A sebagai parent dan class B sebagai subclass dari parent A. Pada percobaan ini class A dan class B dijalankan dalam file berbeda. Class B masih dapat mengakses kelas A karena pada dasarnya modifier default membuat class B dapat mengakses class A yang terdapat pada satu package yang sama.

### Percobaan 7 :

```
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;
    int b;
    int a;

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

public class InheritanceExample {
    public static void main(String[] args) {
        Bapak objectBapak = new Bapak();
        Anak objectAnak = new Anak();

        objectBapak.a=1;
        objectBapak.b=2;
        System.out.println("Object Bapak (Superclass)");

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

pbo_inheritance.Anak > show_variabel >
```

```
run X
PSO_Inheritance (run) X PSO - D:\PSO X
Object Bapak (Superclass)
Nilai a=1
Nilai b=2
Object Anak (Superclass dari Bapak)
Nilai a=0
Nilai b=0
Nilai c=5
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Hasil Analisa :

Walaupun sudah melakukan modifikasi pada method show\_variabel pada class anak dengan menggunakan super untuk menampilkan nilai a dan b nilainya akan tetap 0. Karena nilai dasarnya 0. Jadi, objek subclass tidak akan melakukan "Override" pada objek Bapak selama masih 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");
    }
}
```

## Hasil Analisa :

Pada kelas Parent menurunkan Baby. Terdapat super() pada fungsi constructor yang akan mengoverride class parent-nya. this.babyName = babyName untuk melempar nilai babyName pada objek dengan parameter constructor babyName atau bisa juga Class Baby mewariskan class

Parent. Pada class Baby terdapat super untuk meng-override class Parent nya. Atribut babyName diset pada constructor. Ketika class Baby dibuat akan menampilkan print dari constructor kelas Parent juga.