

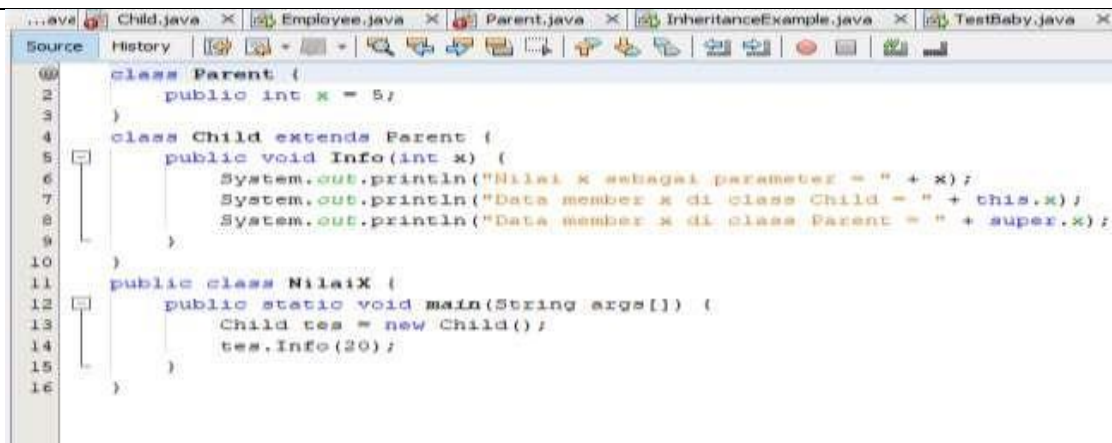
Nama : Tirawati
NIM : 20210040078
Semester/Kelas : 3 (Ganjil) / TI21F

Program Studi : Teknik Informatika
Hari/Tanggal : Kamis, 27 Oktober 2022
Mata Kuliah : Pemrograman
Berorientasi Objek

Praktikum Inheritance Percobaan

1

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



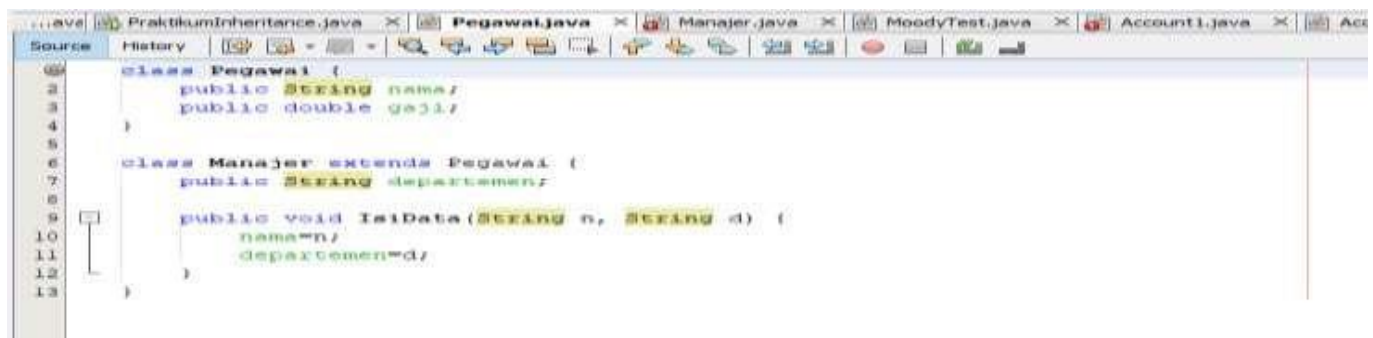
Output :

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

class Parent sebagai induk class yang memiliki atribut integer x = 5, child sebagai sub class dan didalam class Child terdapat sebuah nilai parameter 20, karena ditentukan dari tes info, dan ada data member dari class Parent bernilai 5, kenapa nilainya 5 karena “super” mengambil nilai integer dari class Parent.

Percobaan 2 :

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



Solusinya yaitu mengganti “private String nama” menjadi “ public String nama” tidak error namun tidak bisa di run dikarenakan tidak terdapat main method.

Percobaan 3 :

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



Percobaan ke 3 sama dengan percobaan 2 tidak error namun tidak bisadi run dikarenakan tidak terdapat main method.

Percobaan 4 :

```
import java.util.Date;

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

    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; }
    public Date GetbirthDate(){return birthDate; }
}

class Manager extends Employee {
//tambahan attribrute untuk kelas manager
private String department;

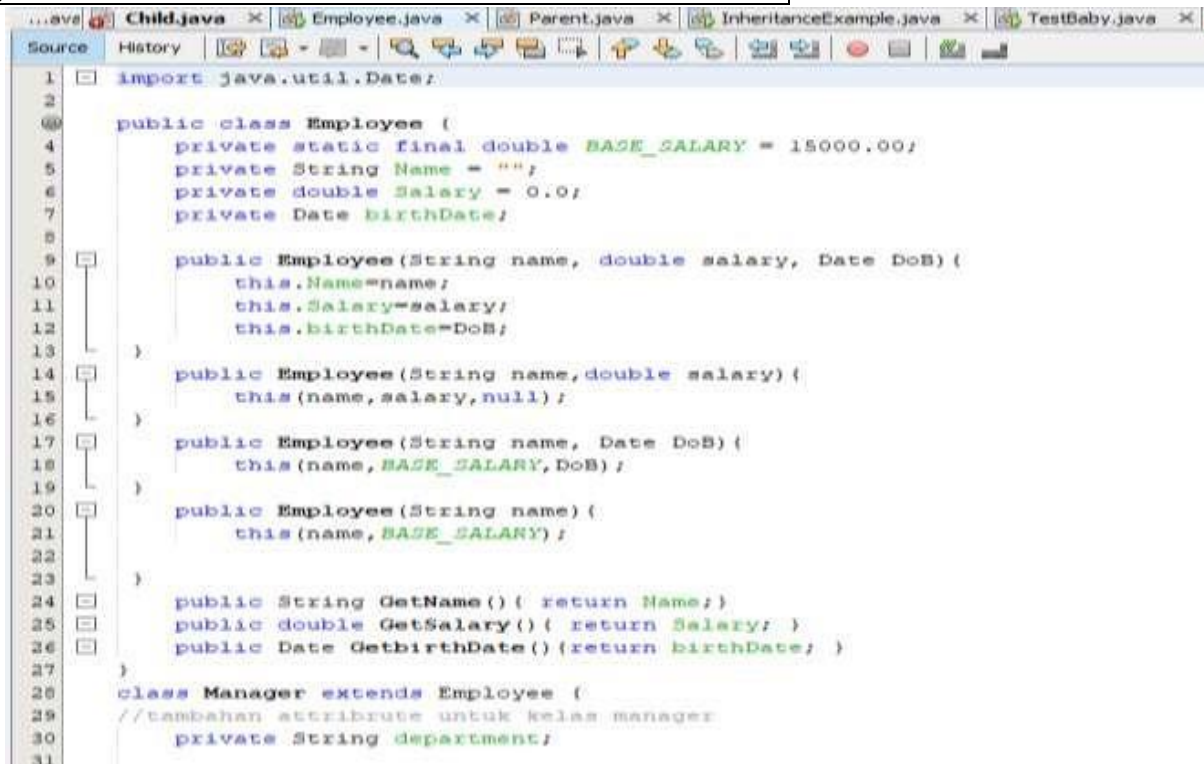
    public Manager(String name, double salary, Date DoB) {
super(name, salary, DoB);
    }
    public Manager(String n,String dept){
super(n);
        department=dept;
    }
    public Manager(String dept, int par, String
financial){        super(dept);        department=dept;
    }
    public String GetDept(){
return department;
    }
}

class TestManager {
    public static void main(String[] args) {
        Manager Utama = new Manager("John","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());
}
}

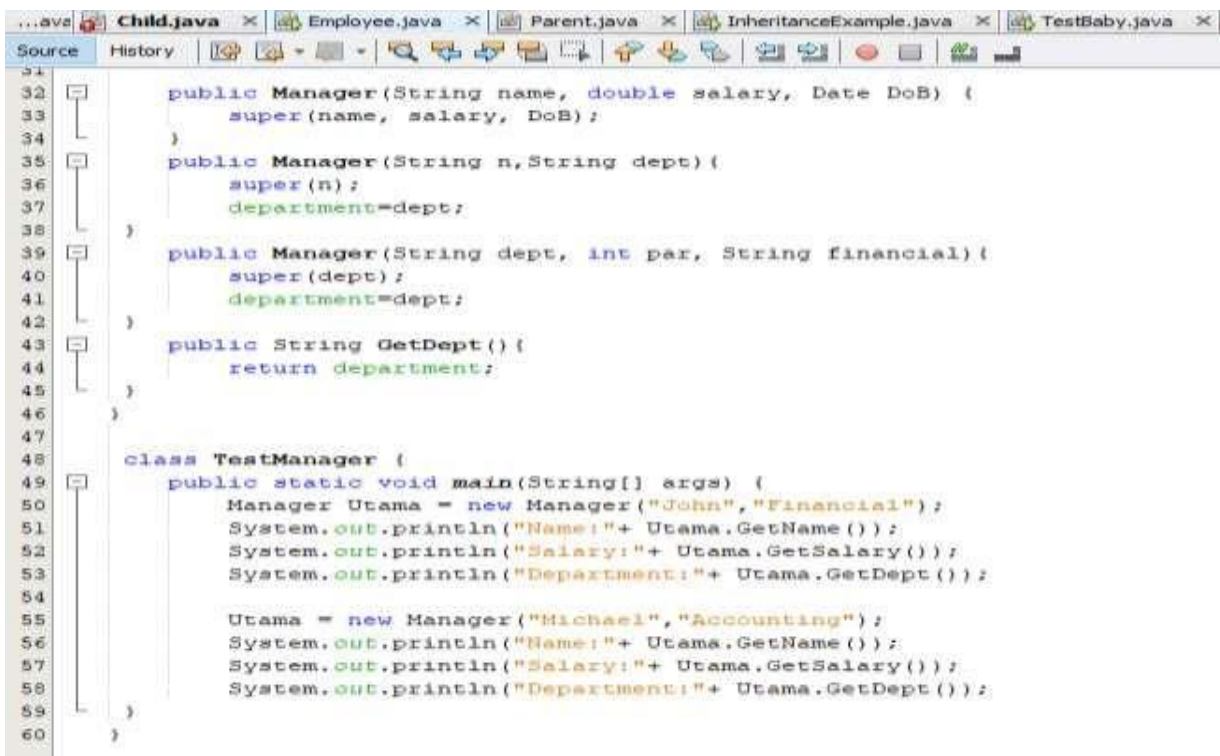
```



```

1  import java.util.Date;
2
3  @
4  public class Employee {
5      private static final double BASE_SALARY = 15000.00;
6      private String Name = "";
7      private double Salary = 0.0;
8      private Date birthDate;
9
10     public Employee(String name, double salary, Date DoB) {
11         this.Name=name;
12         this.Salary=salary;
13         this.birthDate=DoB;
14     }
15     public Employee(String name,double salary){
16         this(name,salary,null);
17     }
18     public Employee(String name, Date DoB){
19         this(name, BASE_SALARY, DoB);
20     }
21     public Employee(String name){
22         this(name, BASE_SALARY);
23     }
24     public String GetName(){ return Name;}
25     public double GetSalary(){ return Salary;}
26     public Date GetbirthDate(){return birthDate;}
27 }
28 class Manager extends Employee {
29     //tambahan attribute untuk kelas manager
30     private String department;
31 }

```



```

31
32     public Manager(String name, double salary, Date DoB) {
33         super(name, salary, DoB);
34     }
35     public Manager(String n,String dept){
36         super(n);
37         department=dept;
38     }
39     public Manager(String dept, int par, String financial){
40         super(dept);
41         department=dept;
42     }
43     public String GetDept(){
44         return department;
45     }
46 }
47
48 class TestManager {
49     public static void main(String[] args) {
50         Manager Utama = new Manager("John","Financial");
51         System.out.println("Name:"+ Utama.GetName());
52         System.out.println("Salary:"+ Utama.GetSalary());
53         System.out.println("Department:"+ Utama.GetDept());
54
55         Utama = new Manager("Michael","Accounting");
56         System.out.println("Name:"+ Utama.GetName());
57         System.out.println("Salary:"+ Utama.GetSalary());
58         System.out.println("Department:"+ Utama.GetDept());
59     }
60 }

```

Percobaan ini menunjukkan penggunaan kelas Employee dan subkelasManager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji jalannya sebuah program tersebut.


Percobaan 5 :

```
public class MoodyObject {
protected String getMood(){
return "moody";
}
public void speak(){
    System.out.println("I am : "+getMood());
} }
public class SadObject extends MoodyObject{

    protected String getSad(){
return "sad";
}
    public void cry(){
        System.out.println("Hoo hoo : " + getSad());
    } }
public class HappyObject extends MoodyObject{

    protected String getHappy(){
return"happy";
}
    public void laugh(){
        System.out.println("Hahaha : " + getHappy());
    } }
public class MoodyTest {
    public static void main(String[] args) {
        MoodyObject m = new MoodyObject();
        SadObject Sad = new SadObject();
        HappyObject Happy = new HappyObject();

        m.speak();
Sad.cry();
        Happy.laugh();
    } }
```



```
2 class MoodyObject {
3     protected String getMood(){
4         return "moody";
5     }
6     public void speak(){
7         System.out.println("I am : "+getMood());
8     }
9 }
10 class SadObject extends MoodyObject{
11     protected String getSad(){
12         return "sad";
13     }
14     public void cry(){
15         System.out.println("Hoo hoo : " + getSad());
16     }
17 }
18 class HappyObject extends MoodyObject{
19     protected String getHappy(){
20         return"happy";
21     }
22     public void laugh(){
23         System.out.println("Hahaha : " + getHappy());
24     }
25 }
26 }
```

```

26     }
27     public class MoodyTest {
28     public static void main(String[] args) {
29         MoodyObject m = new MoodyObject();
30         SadObject Sad = new SadObject();
31         HappyObject Happy = new HappyObject();
32
33         m.speak();
34         Sad.cry();
35         Happy.laugh();
36     }
37 }

```

Output :

```

run:
I am : moody
Hoo hoo : sad
Hahaha : happy
BUILD SUCCESSFUL (total time: 0 seconds)
|

```

Pada Percobaan ini menunjukkan penggunaan kelas MoodyObject dengan subkelas HappyObject dan SadObject. Kelas MoodyTest digunakan untuk menguji kelas dan subkelas dalam menjalankan sebuah Program

1. SadObject berisi : sad, method untuk menampilkan pesan, tipe public 2.

HappyObject berisi : laugh, method untuk menampilkan pesan, tipe public

3. MoodyObject berisi :

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

Percobaan 6 :

```

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

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

```

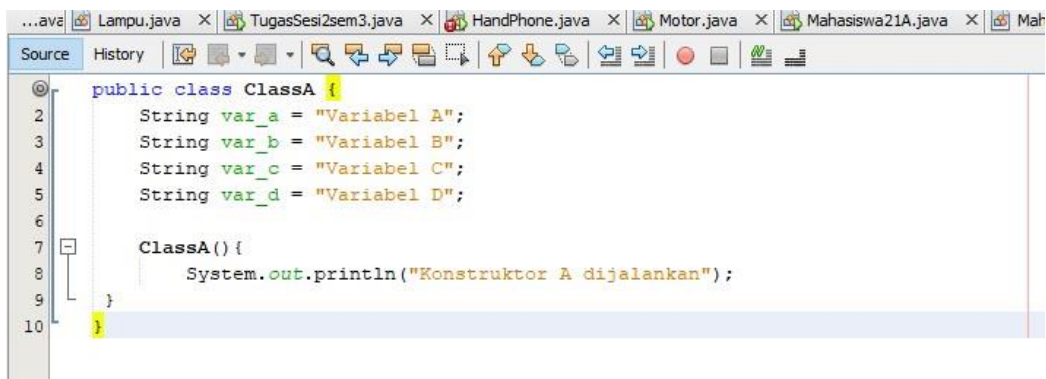


```

public class ClassB extends ClassA{
    ClassB(){
        System.out.println("Konstruktor B dijalankan ");
        var_a = "Var_a dari class B";      var_b = "Var_a
dari class B";      var_c = "Var_a dari class B";
        var_d = "Var_a dari class B";
    }
    public static void main(String args[]){
        System.out.println("Objek A dibuat");
        ClassA aa= new ClassA();
        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");
        ClassB bb= new ClassB();
        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);
    }
}

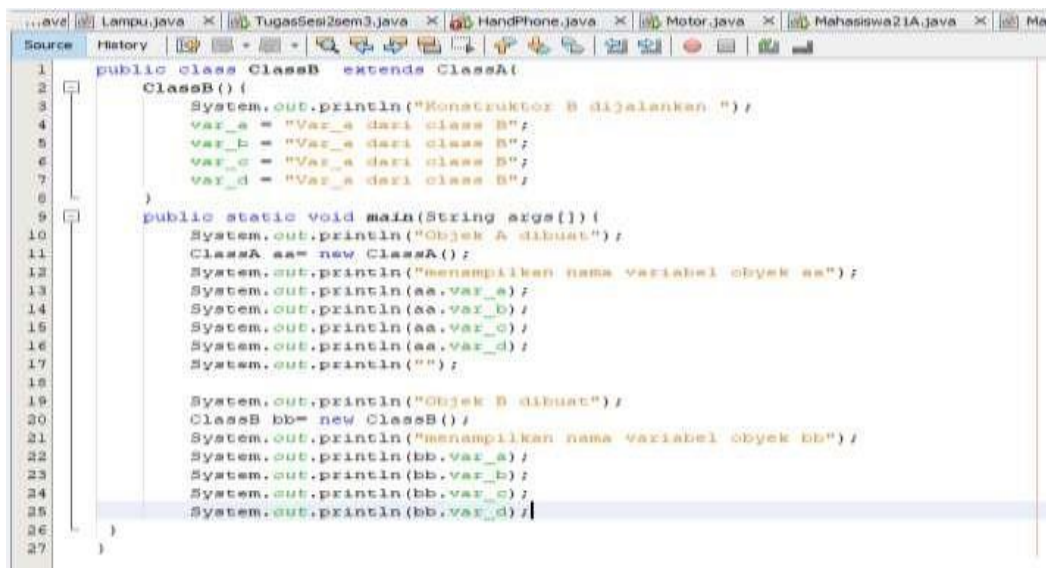
```



```

...ava Lampu.java X TugasSesi2sem3.java X HandPhone.java X Motor.java X Mahasiswa21A.java X Mah
Source History
1 public class ClassA {
2     String var_a = "Variabel A";
3     String var_b = "Variabel B";
4     String var_c = "Variabel C";
5     String var_d = "Variabel D";
6
7     ClassA() {
8         System.out.println("Konstruktor A dijalankan");
9     }
10 }

```



```

...ava Lampu.java X TugasSesi2sem3.java X HandPhone.java X Motor.java X Mahasiswa21A.java X Mah
Source History
1 public class ClassB extends ClassA{
2     ClassB() {
3         System.out.println("Konstruktor B dijalankan ");
4         var_a = "Var_a dari class B";
5         var_b = "Var_a dari class B";
6         var_c = "Var_a dari class B";
7         var_d = "Var_a dari class B";
8     }
9
10    public static void main(String args[]){
11        System.out.println("Objek A dibuat");
12        ClassA aa= new ClassA();
13        System.out.println("menampilkan nama variabel obyek aa");
14        System.out.println(aa.var_a);
15        System.out.println(aa.var_b);
16        System.out.println(aa.var_c);
17        System.out.println(aa.var_d);
18        System.out.println("");
19
20        System.out.println("Objek B dibuat");
21        ClassB bb= new ClassB();
22        System.out.println("menampilkan nama variabel obyek bb");
23        System.out.println(bb.var_a);
24        System.out.println(bb.var_b);
25        System.out.println(bb.var_c);
26        System.out.println(bb.var_d);
27    }
28 }

```

Output :

```
Output - praktikum-inheritance (run)

run:
Objek A dibuat
Konstruktor A dijalankan
menampilkan nama variabel obyek aa
Variabel A
Variabel B
Variabel C
Variabel D

Objek B dibuat
Konstruktor A dijalankan
Konstruktor B dijalankan
menampilkan nama variabel obyek bb
Var_a dari class B
Var_a dari class B
Var_a dari class B
Var_a dari class B
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada percobaan ini menunjukkan penggunaan kelas A dan dengan subkelas B. kemudian simpan file tersebut dalam class yang berbeda dan dalam satu package. Kemudian proses pemanggilan konstruktor dan pemanggilan variabel dalam program tersebut.

Percobaan 7 :

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

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

```
public class Anak extends Bapak{
    int c;
    public 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();

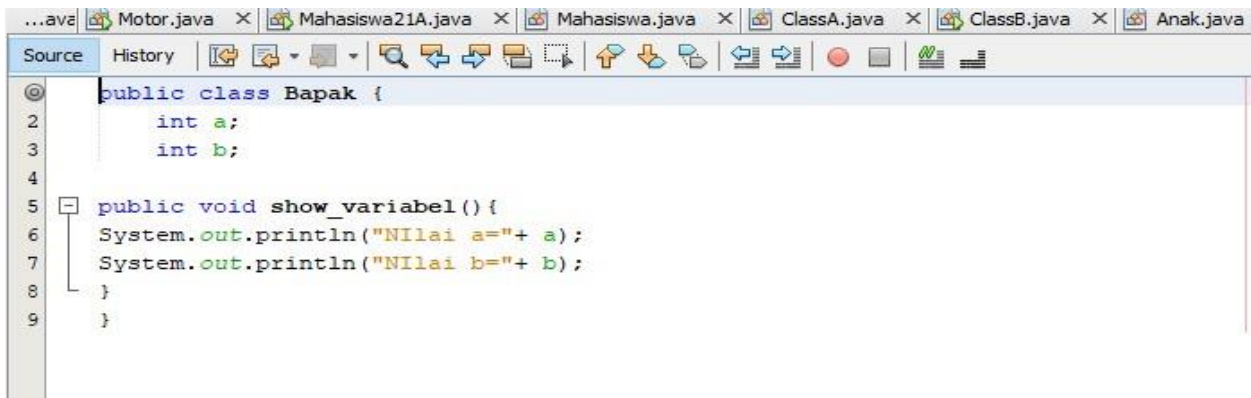
        objectBapak.a=1;
        objectBapak.b=1;
        System.out.println("Object Bapak (Superclass):");
        objectBapak.show_variabel();
    }
}
```



```

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

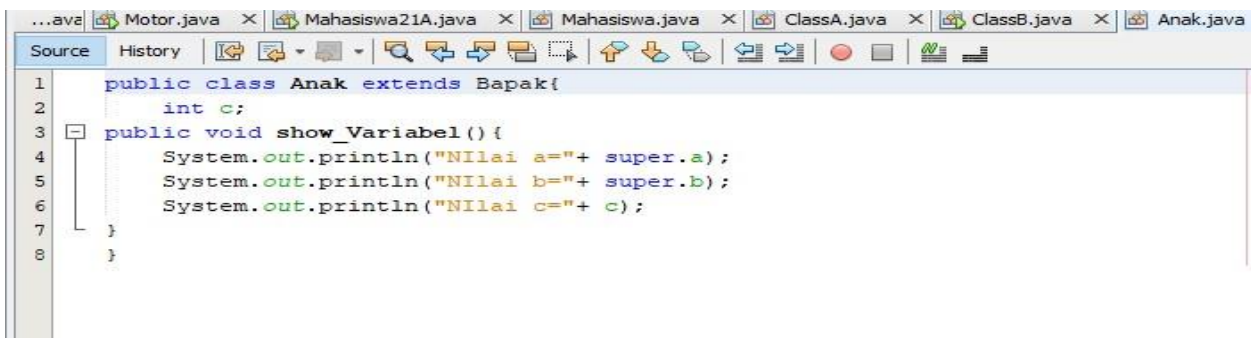
```



```

...ava Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java X Anak.java
Source History
public class Bapak {
2   int a;
3   int b;
4
5   public void show_variabel() {
6       System.out.println("Nilai a="+ a);
7       System.out.println("Nilai b="+ b);
8   }
9 }

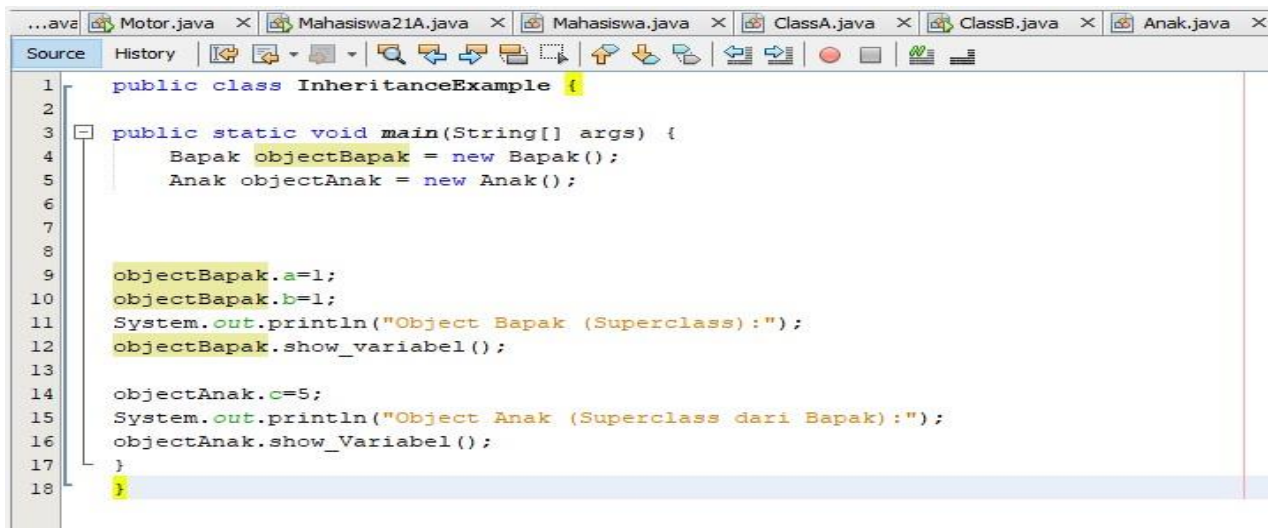
```



```

...ava Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java X Anak.java
Source History
1   public class Anak extends Bapak{
2       int c;
3   public void show_Variabel() {
4       System.out.println("Nilai a="+ super.a);
5       System.out.println("Nilai b="+ super.b);
6       System.out.println("Nilai c="+ c);
7   }
8 }

```




```

...ava Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java X Anak.java X
Source History
1   public class InheritanceExample {
2
3   public static void main(String[] args) {
4       Bapak objectBapak = new Bapak();
5       Anak objectAnak = new Anak();
6
7
8
9       objectBapak.a=1;
10      objectBapak.b=1;
11      System.out.println("Object Bapak (Superclass):");
12      objectBapak.show_variabel();
13
14      objectAnak.c=5;
15      System.out.println("Object Anak (Superclass dari Bapak):");
16      objectAnak.show_Variabel();
17  }
18 }

```

Output :



```

: Output - praktikum-inheritance (run)
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 seconds)

```

Di percobaan ini, terjadi override pada method show_variabel. Terjadi di perubahan nilai pada variabel a, b, dan c. Kemudian dilakukan modifikasi pada sebuah method show_variabel() di class Anak dan gunakan

super untuk menampilkan nilai a dan b. Pada percobaan subclass anak nilai a,b yang mewarisi nilai bapak dan c yaitu nilai dari objek si anak atau buka nilai warisan.

Percobaan 8 :

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

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

```
public class Baby extends Parent{
    String babyName;

    public String getBabyName() {
        return 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");
    }
}
```

```
public class TestBaby {    public static void main(String
args[]){        Baby x = new Baby("Nurul Intan");
        x.cry();
    }
}
```

```
...ava HandPhone.java X Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java
Source History
public class Parent {
    String parentName;
    public Parent() {}

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

```
...ava HandPhone.java X Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java
Source History
public class Baby extends Parent {
    String babyName;

    public String getBabyName() {
        return 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");
    }
}
```

```
...ava HandPhone.java X Motor.java X Mahasiswa21A.java X Mahasiswa.java X ClassA.java X ClassB.java
Source History
public class TestBaby {
    public static void main(String args[]) {
        Baby x = new Baby("Nurul Intan");
        x.cry();
    }
}
```

Output :

```
Output - praktikum-inheritance (run)
run:
Konstruktor Baby
Nurul Intan
owek owek
BUILD SUCCESSFUL (total time: 0 seconds)
```

Percobaan ini menggunakan metode Overriding pada Kelas Parent dan subclass Baby(extends)

Kemudian cara menguji kinerja dari program tersebut dengan membuat class test baby dan program pun akhirnya dapat berjalan.

