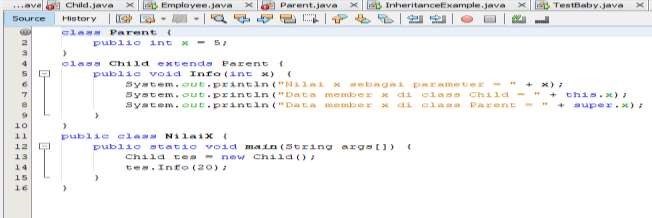
Nama: Nadia Zafira Zahra

NIM: 20210040174

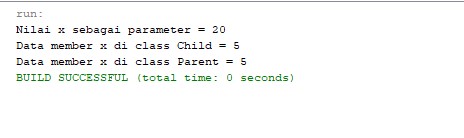
Kelas: TI21F

**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);  }  } |

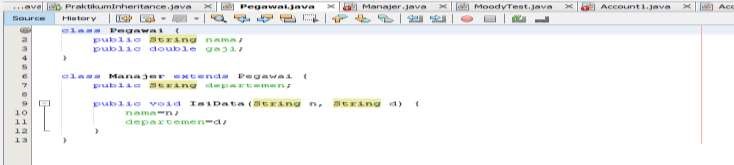


Output :

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 :**

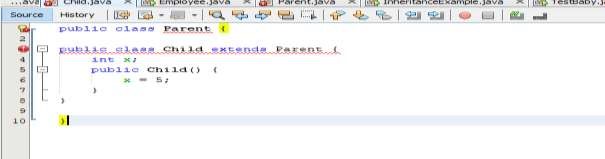
public class Parent {

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()); |

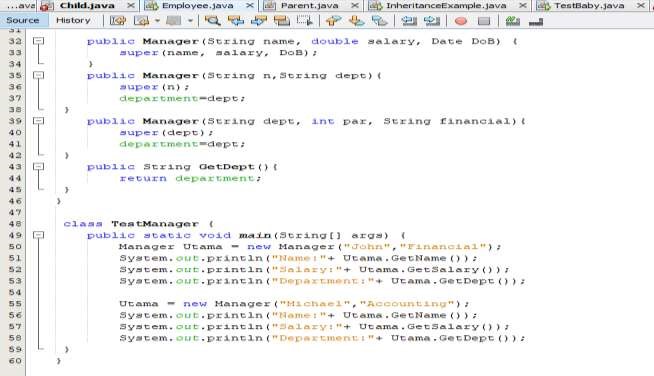
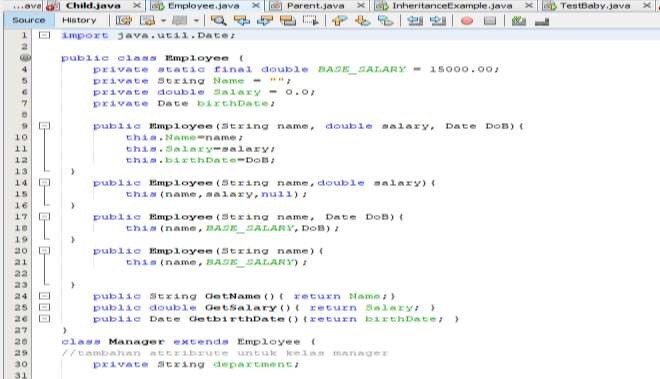
Utama = new Manager("Michael","Accounting");

System.out.println("Name:"+ Utama.GetName());

System.out.println("Salary:"+ Utama.GetSalary());

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

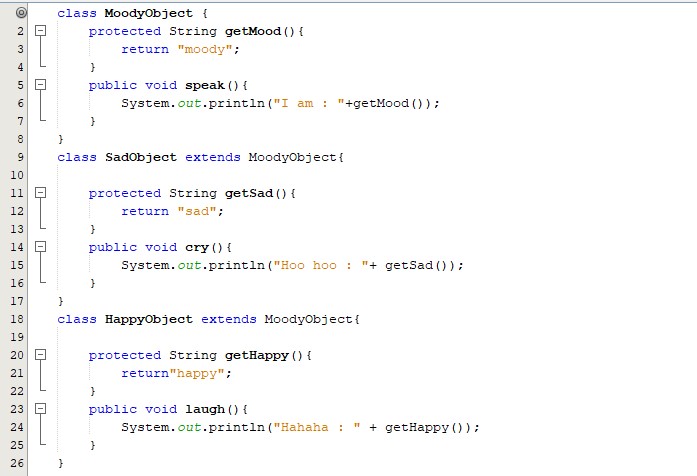
}

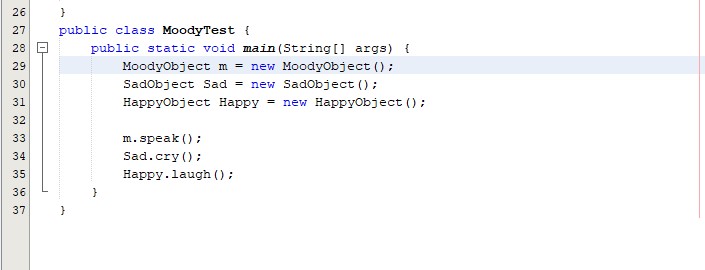


Percobaan ini menunjukkan penggunaan kelas Employee dan subkelasManager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji jalannya sebuah program tersebut.

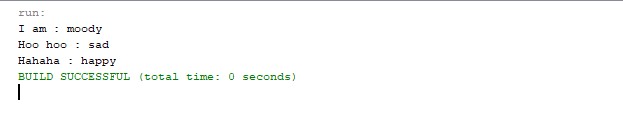
**Percobaan 5 :**

|  |
| --- |
| 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();  } } |





Output :



Pada Percobaan ini menunjukkan penggunaan kelas MoodyObject dengansubkelas 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";

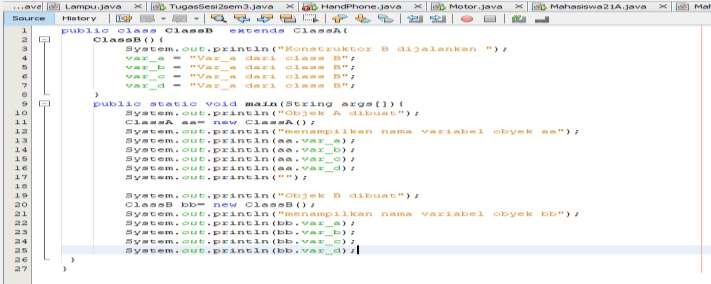
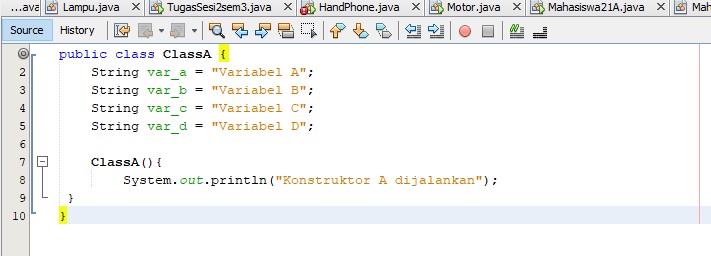
String var\_c = "Variabel C";

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);  }  } |



Output :



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);  }  } |

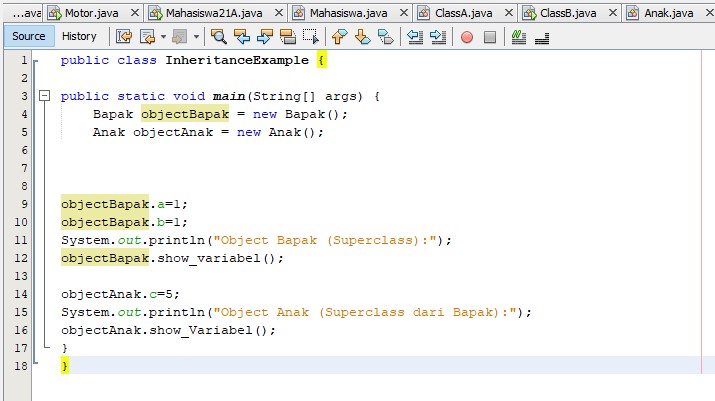
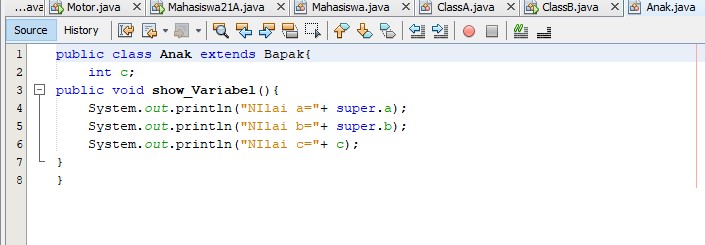
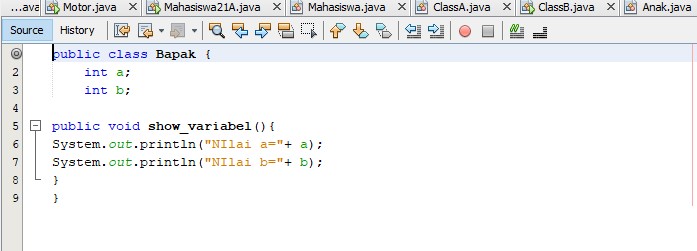
|  |
| --- |
| public class InheritanceExample {    public static void main(String[] args) { Bapak objectBapak = new Bapak();  Anak objectAnak = new Anak();        objectBapak.a=1; objectBapak.b=1;  System.out.println("Object Bapak (Superclass):"); objectBapak.show\_variabel(); |

objectAnak.c=5;

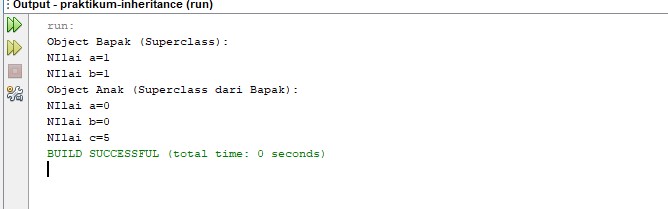
System.out.println("Object Anak (Superclass dari Bapak):"); objectAnak.show\_Variabel();

}

}



**Output :**



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");

}

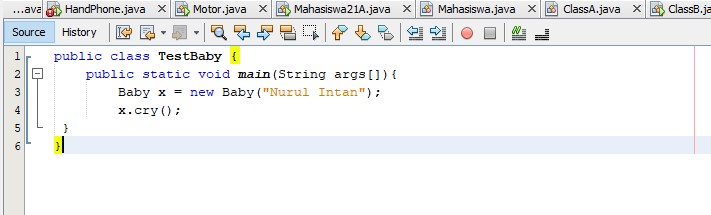
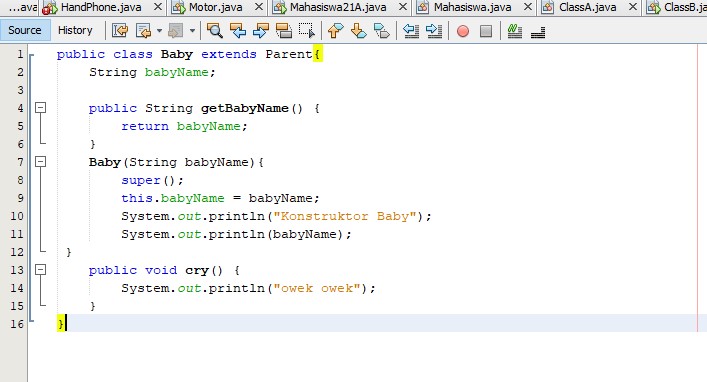
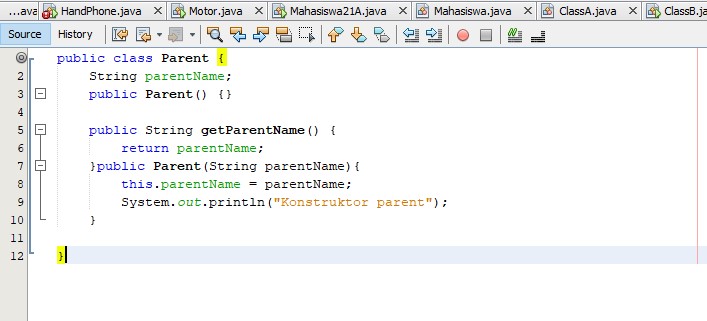
}

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

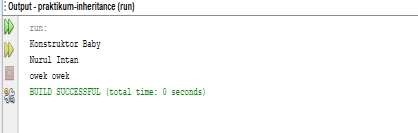
x.cry();

}

}



Output :



Percobaan ini menggunakan methode Overidding pada Kelas Parent dan subclass Baby(extends)

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