Nama : Nurani Istiaen

NIM : 20210040085

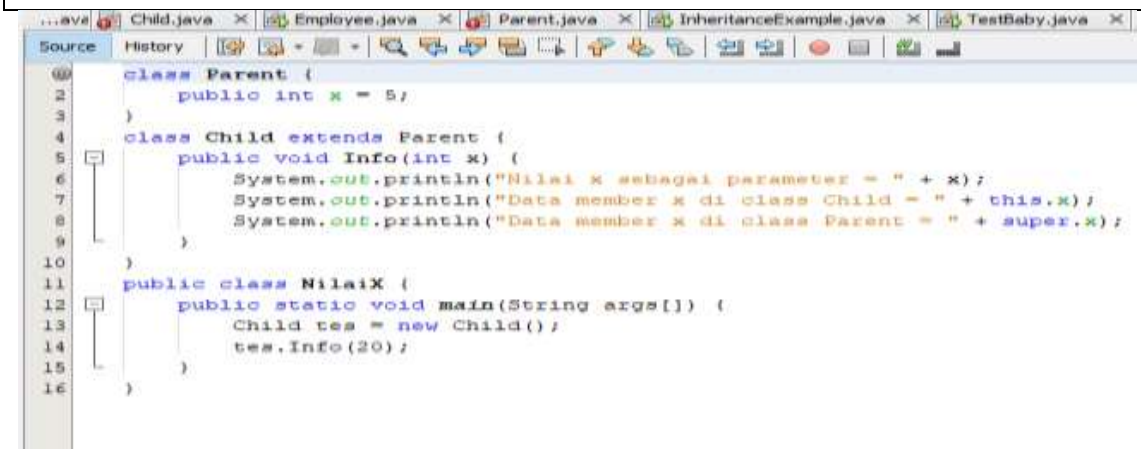
Kelas : TI21A

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)
```

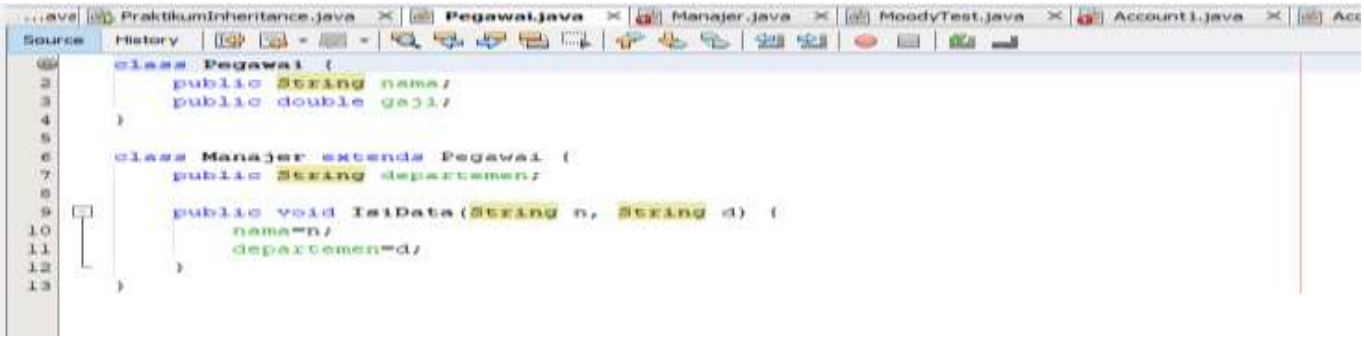
class Parent sebagai induk class yang memiliki atribut integer x = 5, child sebagai sub class dan didalam class Child terdapat sebuah nilai parameter 20, karena ditentukan dari tes info, dan ada data member dari class Parent bernilai 5, kenapa nilainya 5 karena “super” mengambil nilai integer dari class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



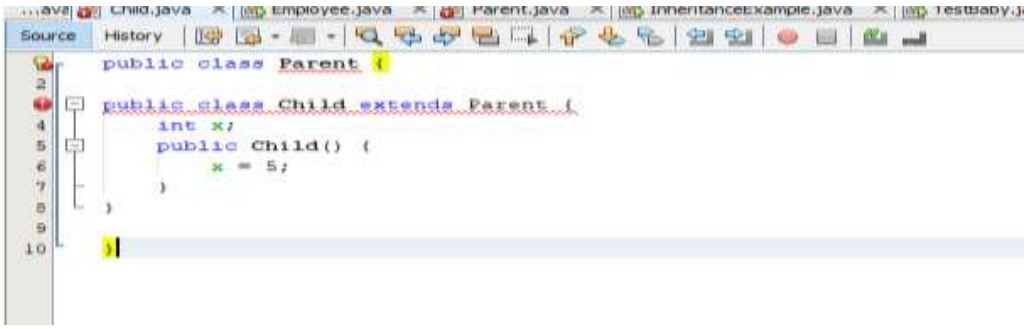
Solusinya yaitu mengganti “private String nama” menjadi “ public String nama’’ tidak error namun tidak bisa di run dikarenakan tidak terdapat main method.

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());
 }
}
```

```
...ava  Child.java  Employee.java  Parent.java  InheritanceExample.java  TestBaby.java
Source  History
1   import java.util.Date;
2
3   public class Employee {
4       private static final double BASE_SALARY = 15000.00;
5       private String Name = "";
6       private double Salary = 0.0;
7       private Date birthDate;
8
9       public Employee(String name, double salary, Date DoB){
10          this.Name=name;
11          this.Salary=salary;
12          this.birthDate=DoB;
13      }
14      public Employee(String name,double salary){
15          this(name,salary,null);
16      }
17      public Employee(String name, Date DoB){
18          this(name,BASE_SALARY,DoB);
19      }
20      public Employee(String name){
21          this(name,BASE_SALARY);
22
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
```

```
...ava  Child.java  Employee.java  Parent.java  InheritanceExample.java  TestBaby.java
Source  History
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();
    }
}
```

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

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

    protected String getHappy(){
        return"happy";
    }
    public void laugh(){
        System.out.println("Hahaha : " + getHappy());
    }
}
```

```
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 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

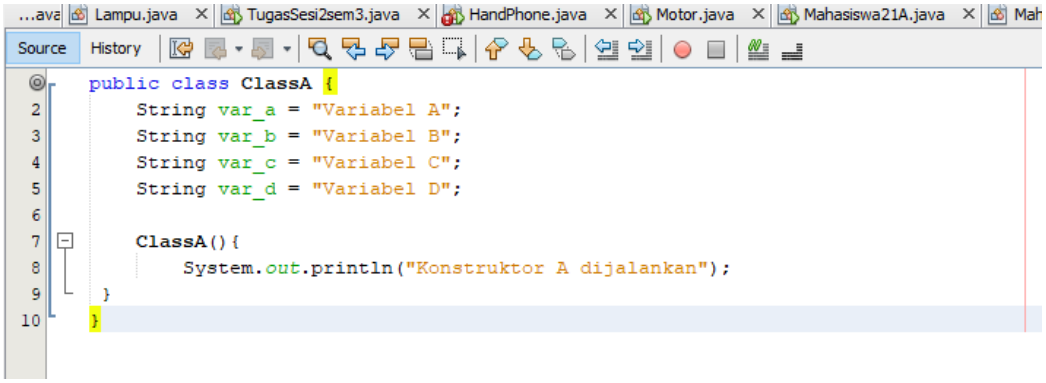
Percobaan 6 :

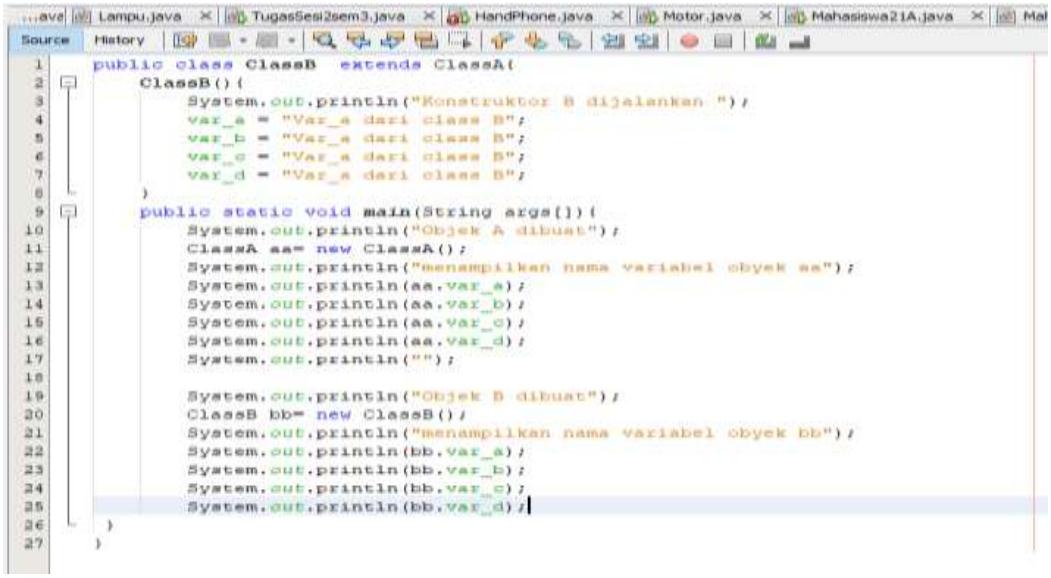
```
public class ClassA {
   String var_a = "Variabel A";
   String var_b = "Variabel B";
   String var_c = "Variabel C";
   String var_d = "Variabel D";

   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 :

```
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();
}
}
```

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

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

```java
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);
}
}
```

```java
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 :

```
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();
 }
}
```

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

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

}
```

```java
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");
    }
}
```

```java
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 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.