

Nama : Muhammad Raihan Izharul Haq

NIM : 20210040075

Kelas : TI21F

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);
    }
}
```

Pada percobaan 1 class Child mewariskan class Parent. Fungsi tes.info(20) akan menset nilai x menjadi 20 yang terdapat pada class Child. 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.

Output :

```
PS D:\Coding\Java\OOP\Myfirstproject> & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_1.NilaiX'
Nilai x sebagai parameter = 20
Data member x di class Child = 10
Data member x di class Parent = 5
PS D:\Coding\Java\OOP\Myfirstproject> █
```

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;  
    }  
}
```

Error yang pertama terjadi karena class Pegawai dinyatakan sebagai public seharusnya public digunakan ketika kelas pegawai diletakan pada filenya sendiri. Error kedua terjadi karena fungsi IsiData di class Manajer menset nama dari class Pegawai akan tetapi data nama di private sehingga terjadi error.

Solusi :

Mengubah modifier atribut nama pada class Pegawai menjadi public agar bisa diakses oleh class Manajer, lalu menghapus public pada class Pegawai menjadi class Pegawai saja.

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;  
    }  
}
```

Tidak terjadi error walaupun class Parent mempunyai constructor.

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());

    }

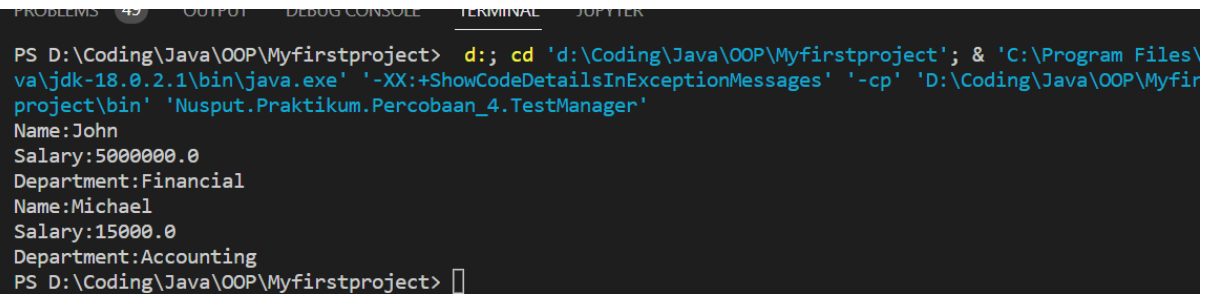
}

```

```
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());
```

Untuk percobaan keempat terdapat error. Sehingga sesuai intruksi tambahkan atribut untuk class Manager yaitu private String department. Pemanggilan objek pertama menggunakan constructor dengan 3 parameter yaitu nama, salary, dan Dept. Sedangkan objek kedua menggunakan constructor dengan 2 parameter yaitu nama dan Dept. Untuk outputnya seperti ini :



```
PS D:\Coding\Java\OOP\Myfirstproject> d:; cd 'd:\Coding\Java\OOP\Myfirstproject'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_4.TestManager'
Name:John
Salary:5000000.0
Department:Financial
Name:Michael
Salary:15000.0
Department:Accounting
PS D:\Coding\Java\OOP\Myfirstproject> 
```

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();
    }
}

```

Percobaan ke 5 menghasilkan object m. Adapun hasil outputnya akan menjadi seperti ini :

```

PS D:\Coding\Java\OOP\Myfirstproject> & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_5.MoodyTest'
I ammoody
I amhappy
Hahaha
I amsad
Hoo hoo
PS D:\Coding\Java\OOP\Myfirstproject>

```

Supaya lebih rapih untuk output bisa disesuaikan dengan menambahkan spasi.

```
PS D:\Coding\Java\OOP\Myfirstproject> d:; cd 'd:\Coding\Java\OOP\Myfirstproject'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_5.MoodyTest'
I am moody
I am happy
Hahaha
I am sad
Hoo hoo
PS D:\Coding\Java\OOP\Myfirstproject>
```

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";
    }

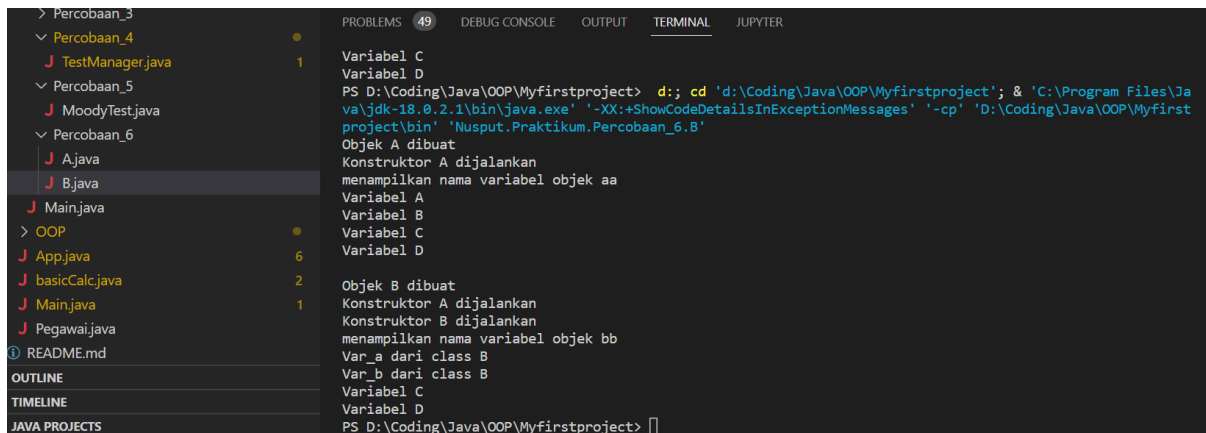
    public static void main(String args[]){

        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);
    }
}
```

Pada constructor B attribute var_b="Var_a dari class B" seharusnya var_b="Var b dari class B". 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

Output :



```
PS D:\Coding\Java\OOP\Myfirstproject> d;; cd 'd:\Coding\Java\OOP\Myfirstproject'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_6.B'
```

Objek A dibuat
Konstruktor A dijalankan
menampilkan nama variabel objek aa
Variabel A
Variabel B
Variabel C
Variabel D

Objek B dibuat
Konstruktor A dijalankan
Konstruktor B dijalankan
menampilkan nama variabel objek bb
Var_a dari class B
Var_b dari class B
Variabel C
Variabel D

Pada output dari objek B menampilkan “konstruktor A dijalankan” hal ini terjadi karena class B mewariskan class A dimana pada constructor class A menampilkan hal tersebut.

Percobaan 7:

Percobaan berikut ini menunjukkan penggunaan Inheritance dan Overriding method pada kelas Bapak dan subkelas Anak. Terjadi override pada method show_variabel. Perhatikan perubahan nilai pada variabel a, b, dan c.

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

public class InheritExample {

    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();
    }
}
```

Kemudian lakukan modifikasi pada method `show_variabel()` pada class `Anak`. Gunakansuper untuk menampilkan nilai a dan b (memanfaatkan method yang sudah ada pada superclass).

Kode sebelum memakai super :

```
package Nusput.Praktikum.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;
    void show_variabel(){
        System.out.println("Nilai a="+a);
        System.out.println("Nilai b="+b);
        System.out.println("Nilai c="+c);
    }
}

public class inheritExample {
    Run | Debug
    public static void main(String[] args) {
        Bapak objectBapak=new Bapak();
        Anak objectAnak=new Anak();
    }
}
```

Outputnya :

```
PS D:\Coding\Java\OOP\Myfirstproject> d:; cd 'd:\Coding\Java\OOP\Myfirstproject'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_7.inheritExample'
Object Bapak (Superclass) :
Nilai a=1
Nilai b=1
Object Anak (Subclass dari Bapak)
Nilai a=0
Nilai b=0
Nilai c=5
PS D:\Coding\Java\OOP\Myfirstproject>
```


Kode menggunakan super:

```
package Nusput.Praktikum.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;
    void show_variabel(){
        super.show_variabel();
        System.out.println("Nilai c="+c);
    }
}

public class inheritExample {
    Run | Debug
    public static void main(String[] args) {
        Bapak objectBapak=new Bapak();
    }
}
```

Ouputnya:

```
PS D:\Coding\Java\OOP\Myfirstproject> d:;; cd 'd:\Coding\Java\OOP\Myfirstproject'; & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Coding\Java\OOP\Myfirstproject\bin' 'Nusput.Praktikum.Percobaan_7.inheritExample'
Object Bapak (Superclass) :
Nilai a=1
Nilai b=1
Object Anak (Subclass dari Bapak)
Nilai a=0
Nilai b=0
Nilai c=5
PS D:\Coding\Java\OOP\Myfirstproject>
```

Print nilai a dan b dalam fungsi show_variabel di class Anak diganti, jadi menggunakan super. Sehingga terjadi override fungsi show_variabel.

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");
    }
}
```

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.