

Nama : Yulva Cintakandida
NIM : 20210040079
Kelas : TI 21 A

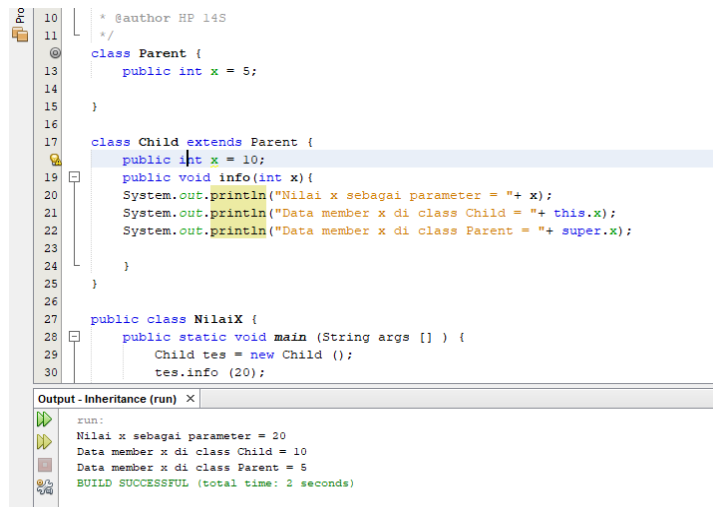
Praktikum 3 Inheritance (Pewarisan)

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);  
    }  
}
```

Jawab :



```
10  * @author HP 14S  
11  */  
12  class Parent {  
13      public int x = 5;  
14  }  
15  
16  class Child extends Parent {  
17      public int x = 10;  
18      public void info(int x) {  
19          System.out.println("Nilai x sebagai parameter = " + x);  
20          System.out.println("Data member x di class Child = " + this.x);  
21          System.out.println("Data member x di class Parent = " + super.x);  
22      }  
23  }  
24  
25  public class NilaiX {  
26      public static void main (String args [] ) {  
27          Child tes = new Child ();  
28          tes.info (20);  
29      }  
30  }
```

Output - Inheritance (run) x

Run:
Nilai x sebagai parameter = 20
Data member x di class Child = 10
Data member x di class Parent = 5
BUILD SUCCESSFUL (total time: 2 seconds)

Class parent sebagai induk class yang memiliki atribut `int = 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:

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;
    }
}

```

Jawab :

Suatu Parent class dapat tidak mewariskan sebagian member nya kepada sub class nya.

Contoh nya pada program diatas

Private string nama;

Hal ini dipengaruhi oleh acces modifier

Ini membuktikan bahwa class Manajer tidak mewarisi data member nama dari parent class nya (pegawai)

Solusinya adalah mengubahnya dengan *public String nama;*

dan keyword *public class* diubah menjadi *class*, karena didalam sebuah class dapat mendeklarasikan suatu variabel.

```

public 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;
    }
}

```

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;
    }
}

```

Jawab :

Kenapa kostruktor terjadi error karena konstruktor tersebut berada di subclass

Sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan constructor superclass terlebih dahulu. Hal ini terjadi karena secara implisit pada constructor subclass ditambahkan pemanggilan `super()` yang bertujuan memanggil constructor superclass oleh kompiler.

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 kelasmanager
    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());
    }
}

```

Jawab :

```

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

```

```

/**
 *
 * @author HP 14S
 */
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());
    }
}

```

Output - Inheritance (run) x

```

run:
Name:John
Salary:5000000.0
Department:Financial
Name:Michael
Salary:15000.0
Department:Accounting
BUILD SUCCESSFUL (total time: 0 seconds)

```

Dari percobaan diatas *public class* testmanager tidak perlu memakai public lagi karena tipe dari class testmanager sudah tipe *public*.

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();
    }
}

```

```

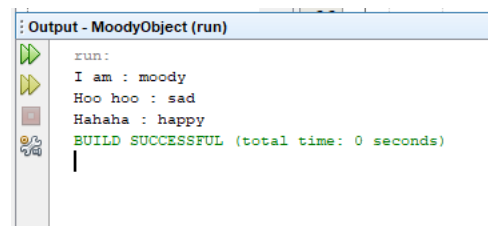
        //test inheritance class
        m = new HappyObject();
        m.speak();
        m.laugh();

        //test inheritance class
        m=new SadObject();
        m.speak();
        m.cry();
    }
}

```

Jawab :

Percobaan diatas menunjukan penampilan pesan mood, dimana kelas ini diuji dengan moody test yang digunakan untuk menguji kelas dan subkelas.



```

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

```

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";
    }

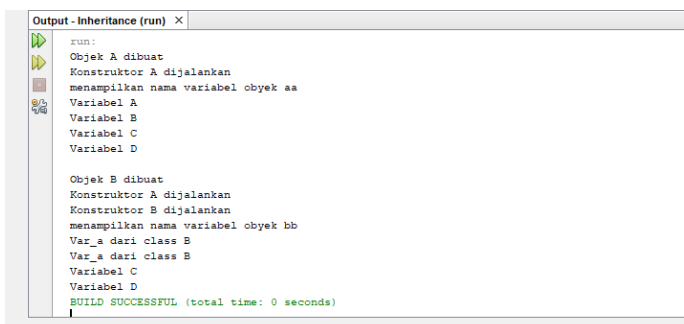
    public static void main(String args[]) {

        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);
    }
}
```

Jawab :

Percobaan ini menunjukkan penggunaan class A dan dengan subclass B. Simpan kedua class ini dalam 2 file class yang berbeda (A.java dan B.java) dan dalam satu package.



```
Output - Inheritance (run) X
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
Variabel C
Variabel D
BUILD SUCCESSFUL (total time: 0 seconds)
```

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();

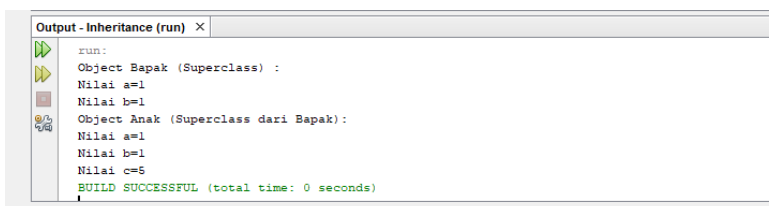
        objectBapak.a=1;
        objectBapak.b=1;
        System.out.println("Object Bapak (Superclass):");

        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. Gunakan super untuk menampilkan nilai a dan b (memanfaatkan method yang sudah ada pada superclass).

Jawab :



```
Output - Inheritance (run) ×
run:
Object Bapak (Superclass) :
Nilai a=1
Nilai b=1
Object Anak (Superclass dari Bapak):
Nilai a=1
Nilai b=1
Nilai c=5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ketika satu program diberikan public pada subclass akan terjadi error kecuali di berbeda program. Percobaan diatas menampilkan bilangan a dan b juga perubahan variabel a,b,c yang telah di modifikasi di class anak. Mengapa class anak dan inheritExample tidak diberi public disamping class, karena satu program dengan kelas bapak

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() {}

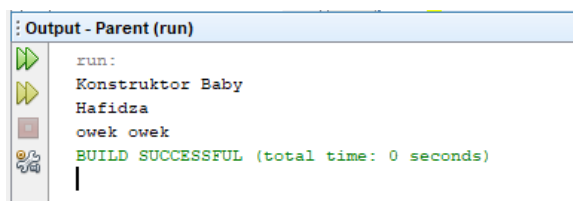
    Parent(String parentName) {
        this.parentName = parentName;
        System.out.println("Konstruktor parent");
    }
}

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");
    }
}
```

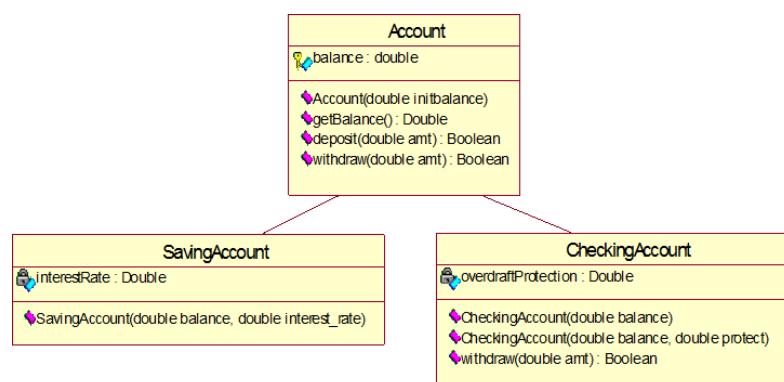
Jawab :



```
run:
Konstruktor Baby
Hafidza
owek owek
BUILD SUCCESSFUL (total time: 0 seconds)
```

Percobaan diatas pemanggilan konstruktor dengan menggunakan super, dimana memanggil konstruktor Baby dengan nama hafidza dan menangis dengan kata owek owek berhasil dijalankan.

Tugas: Pembuatan kelas Account dan subkelas SavingAccount, CheckingAccount



Buat kelas Account sesuai dengan diagram UML untuk kelas Account sebelumnya, dengan definisi :

- Atribut *balance* tipe double, dan sifat protected
- Constructor *Account* untuk memberi nilai awal *balance*
- Method *getBalance* untuk mendapatkan nilai *balance*
- Method *deposit* untuk menambah nilai *balance*
- Method *withdraw* untuk mengambil nilai *balance*

Buat subkelas *SavingAccount* sesuai dengan diagram UML sebelumnya dengan definisi :

- Kelas *SavingAccount* merupakan turunan kelas *Account*, gunakan keyword *extends*.
- Atribut *interestRate*, tipe double, sifat private
- Constructor *SavingAccount* dengan parameter *balance* dan *interest_rate*. Constructor ini harus passing parameter *balance* ke parent constructor dengan menggunakan *super(balance)* dan mengeset nilai variabel *interestRate* dengan nilai *interest_rate*.

Buat kelas *CheckingAccount* sesuai dengan diagram UML sebelumnya dengan definisi :

- Kelas *CheckingAccount* merupakan turunan kelas *Account*, gunakan keyword *extends*.
- Atribut *overdraftProtection*, tipe double, sifat private
- Terdapat public constructor dengan dua parameter: *balance* and *protect*. Constructor ini harus passing parameter *balance* ke parent constructor dengan menggunakan *super(balance)* dan mengeset nilai variabel *overdraftProtection* dengan nilai *protect*.
- Constructor dengan satu parameter yaitu *balance*. Constructor ini harus passing parameter *balance* ke lokal constructor dengan menggunakan *this*. Perhatikan bahwa constructor lain yang ada adalah constructor dengan dua parameter. Maka buat nilai *protect default* adalah -1.0 yang berarti bahwa pada account tidak terdapat *overdraftProtection*.
- $\text{Saldo} = \text{balance} + \text{overdraftProtection}$
- *overdraftProtection* = Saldo minimal, yaitu saldo yang diharapkan tidak boleh diambil pada suatu rekening, kecuali bila konsumen ingin menutup rekening.
- Class *CheckingAccount* harus mengoverride method *withdraw*. Method *withdraw* harus melakukan cek terhadap saldo (*balance*) apakah jumlahnya cukup bila terjadi pengambilan sejumlah uang (*amount*). Cek yang dilakukan adalah sebagai berikut :
- Jika $\text{balance} - \text{amount} \Rightarrow 0.0$ maka proses pengambilan diperbolehkan dan mengembalikan nilai *true*. Dan selanjutnya set $\text{balance} = \text{balance} - \text{amount}$;
- Jika $\text{balance} - \text{amount} < 0.0$ maka lakukan cek sebagai berikut:
- Jika tidak ada *overdraftProtection* (nilai = -1.0) atau $\text{overdraftProtection} < \text{overdraftNeeded} (\text{amount} - \text{balance})$ maka gagalkan proses pengambilan uang dengan mengembalikan nilai *false*.
- Jika terdapat *overdraftProtection* atau $\text{overdraftProtection} > \text{overdraftNeeded} (\text{amount} - \text{balance})$ maka proses pengambilan uang berhasil dengan mengembalikan nilai *true*. Dan selanjutnya set $\text{balance} = 0.0$; $\text{overdraftProtection} = \text{overdraftProtection} - \text{overdraftNeeded}$;
- Constructor dengan satu parameter yaitu *balance*. Constructor ini harus passing parameter *balance* ke lokal constructor dengan menggunakan *this*. Perhatikan bahwa constructor lain yang ada adalah constructor dengan dua parameter. Maka buat nilai *protect default* adalah -1.0 yang berarti bahwa pada account tidak terdapat *overdraftProtection*.
- $\text{Saldo} = \text{balance} + \text{overdraftProtection}$, *overdraftProtection* adalah saldo minimal, yaitu saldo yang diharapkan tidak boleh diambil pada suatu rekening, kecuali bila konsumen ingin menutup rekening.
- *Overdraft Protection* (Proteksi Cerukan) yaitu fasilitas kredit kepada nasabah penyimpan dana untuk menutupi cerukan; fasilitas kredit bank tersebut memungkinkan nasabah untuk menarik cek yang melebihi dana tersedia pada saldo akunnya sehingga kelebihan penarikan dana tersebut dikenakan bunga harian; apabila

kelebihan penarikan dana ditutup dengan fasilitas kreditnya, kelebihan penarikan itu tidak dikenakan bunga harian.