

PRAKTIKUM
PEMROGRAMAN BERBASIS OBJEK
LEMBAR KERJA MAHASISWA
Inheritance (Pewarisan Class)



oleh:

Refangga Lintar Prayoga (IS-05-01 - 1204220137)

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI DAN BISNIS
INSTITUT TEKNOLOGI TELKOM SURABAYA
2022

Contoh Inheritance Mewariskan Attribute Class

```
package testt;  
public class Employee {  
    int salary = 4000000;  
}
```

```
package testt;  
public class Marketing  
    extends Employee {  
    int bonus = 1000000;  
}
```

```
package testt;  
public class Testt {  
    public static void main(String[] args) {  
        // Create Marketing Object  
        Marketing budi = new Marketing();  
  
        // Access Marketing(Own) attribute  
        System.out.println("Bonus: " + budi.bonus);  
  
        // Access Employee(Parent) attribute  
        System.out.println("Salary: " + budi.salary);  
    }  
}
```

Output:

```
Output - Testt (run)  
run:  
Bonus: 1000000  
Salary: 4000000  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan:

Pada baris pertama, kita mendefinisikan sebuah paket bernama "testt". Paket digunakan untuk mengelompokkan kelas-kelas terkait bersama-sama. Dalam hal ini, kelas "Testt" akan berada di dalam paket "testt". Kemudian, kita mendefinisikan sebuah kelas bernama "Testt" dengan menggunakan kata kunci "class". Kelas ini bersifat publik (public), sehingga dapat diakses dari kelas-kelas lain di luar paket.

Di dalam kelas "Testt", terdapat sebuah metode utama (main method) yang digunakan sebagai titik awal eksekusi program. Method ini harus ada dalam setiap aplikasi Java dan memiliki tanda tangan yang spesifik seperti di atas. Pada baris-baris di bawahnya, kita akan menulis kode yang akan dijalankan ketika program dijalankan. Pada baris ini, kita membuat sebuah objek bernama "budi" dari kelas "Marketing".

Kode ini menggunakan sintaksis "new Marketing()" untuk membuat objek baru dan kemudian menginisialisasi variabel "budi" dengan objek tersebut. Dengan membuat objek "budi", kita dapat mengakses atribut dan metode yang ada dalam kelas "Marketing". Dalam hal ini kita memiliki 2 public class dalam 1 folder yang bisa dipanggil yaitu employee dan marketing. Pada baris-baris ini, kita mengakses atribut dari objek "budi" yang telah dibuat sebelumnya. Kita menggunakan operator titik (.) untuk mengakses atribut dari objek. Dalam contoh ini, kita mencetak nilai atribut "bonus" dan "salary" menggunakan pernyataan "System.out.println()".

Contoh Inheritance Mewariskan Method Class

```
package testtt;  
//Parent class  
public class Animal {  
    //Method  
    void eat(){  
  
        System.out.println("Eating.");  
    }  
}
```

```
package testtt;  
// Child Class extends Parent  
Class  
public class Dog extends Animal {  
    // Method  
    void bark(){  
  
        System.out.println("Barking!!!");  
    }  
}
```

```
package testtt;  
public class Testtt {  
    public static void main(String[] args) {  
        Dog myDog = new Dog();  
  
        myDog.bark(); //Access Dog(Own)  
        myDog.eat(); // Access Animal  
        (Parent) method  
    }  
}
```

Output:

Output - Testtt (run)

```
run:  
Barking!!!  
Eating.  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan:

Pada baris pertama, kita mendefinisikan sebuah paket bernama "testtt". Paket digunakan untuk mengelompokkan kelas-kelas terkait bersama-sama. Dalam hal ini, kelas "Testtt" akan berada di dalam paket "testtt". Kemudian, kita mendefinisikan sebuah kelas bernama "Testtt" dengan menggunakan kata kunci "class". Kelas ini bersifat publik (public), sehingga dapat diakses dari kelas-kelas lain di luar paket.

Di dalam kelas "Testtt", terdapat sebuah metode utama (main method) yang digunakan sebagai titik awal eksekusi program. Method ini harus ada dalam setiap aplikasi Java dan memiliki tanda tangan yang spesifik seperti di atas. Pada baris-baris di bawahnya, kita akan menulis kode yang akan dijalankan ketika program dijalankan. Pada baris ini, kita membuat sebuah objek bernama "myDog" dari kelas "Dog". Kode ini menggunakan sintaksis "new Dog()" untuk membuat objek baru dan kemudian menginisialisasi variabel "myDog" dengan objek tersebut. Dengan membuat objek "myDog", kita dapat mengakses metode yang ada dalam kelas "Dog". Disini terdapat 2 class java yaitu animal dan dog.

Pada baris-baris ini, kita memanggil metode dari objek "myDog" yang telah dibuat sebelumnya. Kita menggunakan operator titik (.) untuk memanggil metode dari objek. Dalam contoh ini, kita memanggil metode "bark()" yang merupakan metode kelas "Dog" sendiri, dan juga memanggil metode "eat()" yang merupakan metode kelas "Animal" yang menjadi superclass dari "Dog".

Contoh Inheritance Lainnya

```
package testttt;
// Parent Class
class Bicycle {
    // Attributes
    public int gear;
    public int speed;

    //Constructor
    public Bicycle(int gear, int speed){
        this.gear = gear;
        this.speed = speed;
    }

    //Method
    public void applyBrake(int decrement){
        speed -= decrement;
    }

    //Method
    public void speedUp(int increment){
        speed += increment;
    }

    //Method
    public String toString(){
        return "No of gears are " + gear +
            "\n" + "speed of bicycle is " + speed;
    }
}
```

```
// Child Class extends Parent Class
package testttt;
public class MountainBike extends Bicycle
{
    // Attribute
    public int seatHeight;

    //Constructor

    public MountainBike(int gear, int
        speed, int startHeight) {
        // Call Parent Constructor
        super(gear, speed);

        seatHeight = startHeight;
    }

    // Method
    public void setHeight(int
        newValue){
        seatHeight = newValue;
    }

    @Override
    public String toString(){
        return super.toString() +
            "\nseat height is" + seatHeight;
    }
}
```

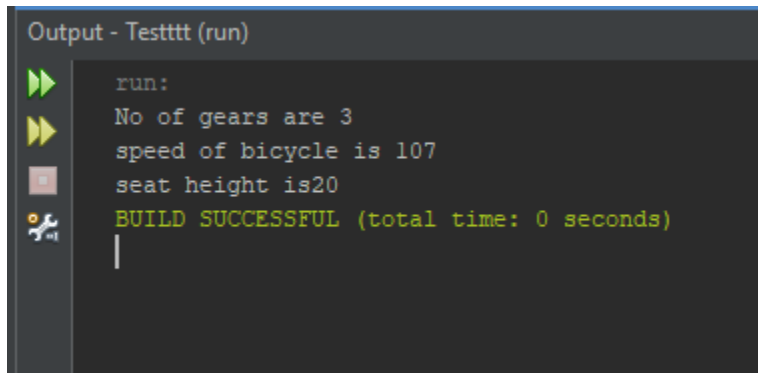
```
package testttt;
public class Testttt {
    public static void main(String[] args) {
        // Create MountainBike Object
        MountainBike mb = new MountainBike(3,100,25);

        // Access parent method
        mb.applyBrake(3);
        mb.speedUp(10);

        // Access own method
        mb.setHeight(20);

        System.out.println(mb.toString());
    }
}
```

Output:



```
Output - Testttt (run)
run:
No of gears are 3
speed of bicycle is 107
seat height is20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan:

Pada baris pertama, kita mendefinisikan sebuah paket bernama "testttt". Paket digunakan untuk mengelompokkan kelas-kelas terkait bersama-sama. Dalam hal ini, kelas "Testttt" akan berada di dalam paket "testttt". Kemudian, kita mendefinisikan sebuah kelas bernama "Testttt" dengan menggunakan kata kunci "class". Kelas ini bersifat publik (public), sehingga dapat diakses dari kelas-kelas lain di luar paket.

Di dalam kelas "Testttt", terdapat sebuah metode utama (main method) yang digunakan sebagai titik awal eksekusi program. Method ini harus ada dalam setiap aplikasi Java dan memiliki tanda tangan yang spesifik seperti di atas. Pada baris-baris di bawahnya, kita akan menulis kode yang akan dijalankan ketika program dijalankan.

Pada baris ini, kita membuat sebuah objek bernama "mb" dari kelas "MountainBike". Kode ini menggunakan sintaksis "new MountainBike(3,100,25)" untuk membuat objek baru dan menginisialisasi variabel "mb" dengan objek tersebut. Dalam hal ini, kita memberikan nilai 3, 100, dan 25 sebagai argumen untuk konstruktor "MountainBike". Disana terdapat 2 class yaitu Bicycle dan MountainBike. Pada baris-baris ini, kita memanggil metode dari objek "mb" yang telah dibuat sebelumnya. Kita menggunakan operator titik (.) untuk memanggil metode dari objek. Dalam contoh ini, kita memanggil metode "applyBrake(3)" dan "speedUp(10)" untuk melakukan operasi pada objek "mb". Pada baris ini, kita memanggil metode "setHeight(20)" yang merupakan metode dari kelas "MountainBike" sendiri. Metode ini tidak ada pada superclass-nya. Pada baris ini,

kita menggunakan pernyataan "System.out.println()" untuk mencetak keluaran ke layar. Kita memanggil metode "toString()" pada objek "mb" untuk mendapatkan representasi string dari objek tersebut, dan kemudian mencetaknya.

Latihan

6. Latihan



Jika ada 3 class seperti di atas yang memiliki atribut dan method sedemikian rupa, Terapkan konsep inheritance sehingga kode program menjadi optimal tidak ada redundansi. Buat program dari konsep inheritance yang telah dibuat sehingga dapat menghasilkan output seperti tampilan di bawah ini.

```
Name: Justin
NIP : 123456789
Salary : 12000000
Action: Manage the Team!

Name: John
NIP : 0987654321
Salary : 10000000
Action: Lets make some code!
```

- Method `setName()`, `setNip()`, dan `setSalary()` adalah method Setter untuk menetapkan nilai dari atribut `name`, `nip` dan `salary`.
- Method `show()` digunakan untuk menampilkan text dan nilai dari `Name`, `NIP` dan `Salary`.
- method `manage()` digunakan untuk menampilkan text "Action: Manage the Team!".
- method `coding()` digunakan untuk menampilkan text "Action: Lets make some code!".

Contoh Syntax Program

```
package employs;

public class Employs {
    String name;
    int nip;
    int salary;

    public void setEmploys(String name, int nip, int salary) {
        setName(name);
        setNip(nip);
        setsalary(salary);
    }

    // Getter dan Setter
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getNip() {
        return nip;
    }

    public void setNip(int nip) {
        this.nip = nip;
    }

    public int getsalary () {
        return salary;
    }

    public void setsalary(int salary) {
        this.salary = salary;
    }

    public void show(){
        System.out.println("Nama: "+getName());
        System.out.println("NIP: "+getNip());
        System.out.println("Salary: "+getsalary());
    }

    public static void main(String[] args) {
        Manager manager1 = new Manager();
        manager1.setEmploys("Justin", 123456789, 120000000);
        Programmer programmer1 = new Programmer();
        programmer1.setEmploys("John", 987654321, 100000000);
        manager1.show();
        System.out.println("");
        programmer1.show();
    }
}

class Manager extends Employs{

    public void manage(){
        System.out.println("Action: Manage the team!");
    }

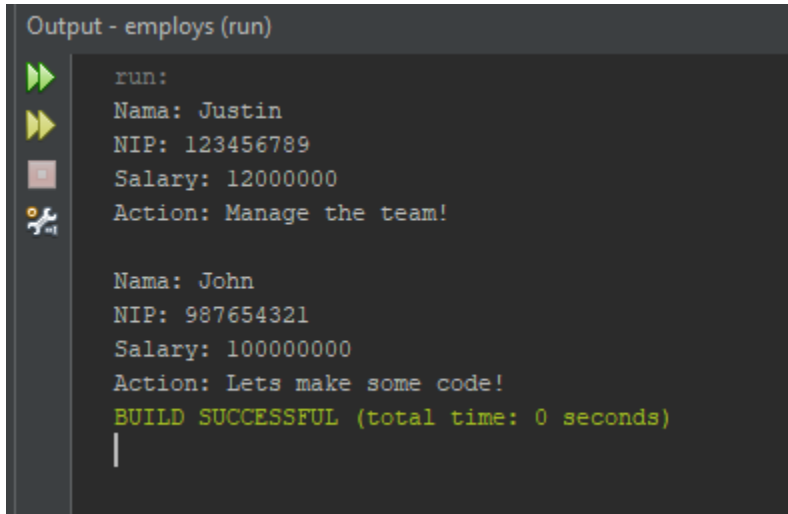
    public void show(){
        System.out.println("Nama: " + getName());
        System.out.println("NIP: " + getNip());
        System.out.println("Salary: " + getsalary());
        manage();
    }
}

class Programmer extends Employs{

    public void coding(){
        System.out.println("Action: Lets make some code!");
    }

    public void show(){
        System.out.println("Nama: " + getName());
        System.out.println("NIP: " + getNip());
        System.out.println("Salary: " + getsalary());
        coding();
    }
}
```

Output:



```
Output - employs (run)

run:
Nama: Justin
NIP: 123456789
Salary: 12000000
Action: Manage the team!

Nama: John
NIP: 987654321
Salary: 100000000
Action: Lets make some code!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan:

Pada baris pertama, kita mendefinisikan sebuah paket bernama "employs". Paket digunakan untuk mengelompokkan kelas-kelas terkait bersama-sama. Dalam hal ini, semua kelas dalam file ini akan berada di dalam paket "employs". Kemudian, kita mendefinisikan sebuah kelas bernama "Employs" dengan menggunakan kata kunci "class". Kelas ini bersifat publik (public), sehingga dapat diakses dari kelas-kelas lain di luar paket.



```
String name;
int nip;
int salary;
```

Pada bagian ini, kita mendeklarasikan tiga variabel instance, yaitu "name" (nama), "nip" (nomor induk pegawai), dan "salary" (gaji). Variabel ini akan digunakan untuk menyimpan informasi tentang seorang pegawai.

```

public void setEmploys(String name, int nip, int salary) {
    setNama(name);
    setNip(nip);
    setsalary(salary);
}

```

Di dalam kelas "Employs", kita mendefinisikan sebuah metode bernama "setEmploys" yang berfungsi untuk mengatur nilai dari variabel instance dengan argumen yang diberikan. Metode ini digunakan untuk menginisialisasi objek pegawai dengan nilai-nilai yang diberikan.

```

public String getName() {
    return name;
}

public void setNama(String name) {
    this.name = name;
}

public int getNip() {
    return nip;
}

public void setNip(int nip) {
    this.nip = nip;
}

public int getsalary () {
    return salary;
}

public void setsalary(int salary) {
    this.salary = salary;
}

```

Dalam kelas "Employs", kita memiliki sejumlah metode getter dan setter untuk mengakses dan mengatur nilai dari variabel instance. Metode getter digunakan untuk mengambil nilai variabel, sedangkan metode setter digunakan untuk mengatur nilai variabel.

```
public void show(){  
    System.out.println("Nama: "+getName());  
    System.out.println("NIP: "+getNip());  
    System.out.println("Salary: "+getsalary());  
}
```

Dalam kelas "Employs", kita memiliki metode "show" yang digunakan untuk mencetak informasi pegawai seperti nama, NIP, dan gaji ke layar.

```
public static void main(String[] args) {  
    Manager manager1 = new Manager();  
    manager1.setEmploys("Justin", 123456789, 120000000);  
    Programmer programmer1 = new Programmer();  
    programmer1.setEmploys("John", 987654321, 1000000000);  
}
```

Merupakan method main yang berisi untuk membuat objek dari class-class tersebut