

Praktikum 5 Pemrograman Berbasis Object

Overloading,Kata Kunci Static,Inheritance

1. Overloading

Method Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, yang membedakan adalah parameternya.

Pada method overloading perbedaan parameter mencakup :

1. Jumlah parameter
2. Tipe data dari parameter
3. Urutan dari tipe data parameter

Contoh program Java tentang overloading

silahkan buka netbean

Buat class dengan nama class ContohOverlading

```
1 package overloading;
2 /**
3  *
4  * @author agung
5  */
6 public class ContohOverlading {
7
8     public void jumlah (int a, int b){
9         System.out.println("Jumlah 2 angka =" + (a + b));
10    }
11
12    //overloading perbedaan jumlah parameter
13    public void jumlah (int a, int b, int c){
14        System.out.println("Jumlah 3 angka =" + (a + b + c));
15    }
16
17    //overloading perbedaan tipe data parameter
18    public void jumlah (double a, int b){
19        System.out.println("Jumlah 2 angka (double+int) = " + (a + b));
20    }
21
22    //overloading perbedaan urutan tipe data parameter
23    public void jumlah (int b, double a){
24        System.out.println("Jumlah 2 angka (int+double) = " + (a + b));
25    }
26
27 }
```

Buat object dengan nama JalanOverloading

```
package overloading;

/**
 *
 * @author agung
 */
public class JalanOverloading {
    public static void main(String[] args) {

        ContohOverloading co = new ContohOverloading();

        co.jumlah(83,32);
        co.jumlah(34,454,432);
        co.jumlah(34.43,34);
        co.jumlah(28,33.23);

    }
}
```

Run program tersebut

```
run:
Jumlah 2 angka =115
Jumlah 3 angka =920
Jumlah 2 angka (double+int) = 68.43
Jumlah 2 angka (int+double) = 61.23
BUILD SUCCESSFUL (total time: 4 seconds)
```

2. Kata Kunci Static

Penerapan keyword static pada method dan variable , tujuannya agar kita tidak perlu lagi membuat objek baru ,ketika digunakan oleh class lain.

Contoh program Java tentang overloading

silahkan buka netbean

Buat sebuah class dengan nama Cetak

```
package keywordstatic;

/**
 *
 * @author agung
 */
public class Cetak {

    static void tampilKata(){
        //membuat method void dengan keyword static
        System.out.println("Halo");
        System.out.println("Selamat Pagi");
    }

}
```

Buat sebuah object dengan nama CobaCetak

```
package keywordstatic;

/**
 *
 * @author agung
 */
public class CobaCetak {

    public static void main(String[] args) {

        Cetak.tampilKata();

    }

}
```

Run Program tersebut

```
Output - keywordstatic (run) x
run:
Halo
Selamat Pagi
BUILD SUCCESSFUL (total time: 0 seconds)
```

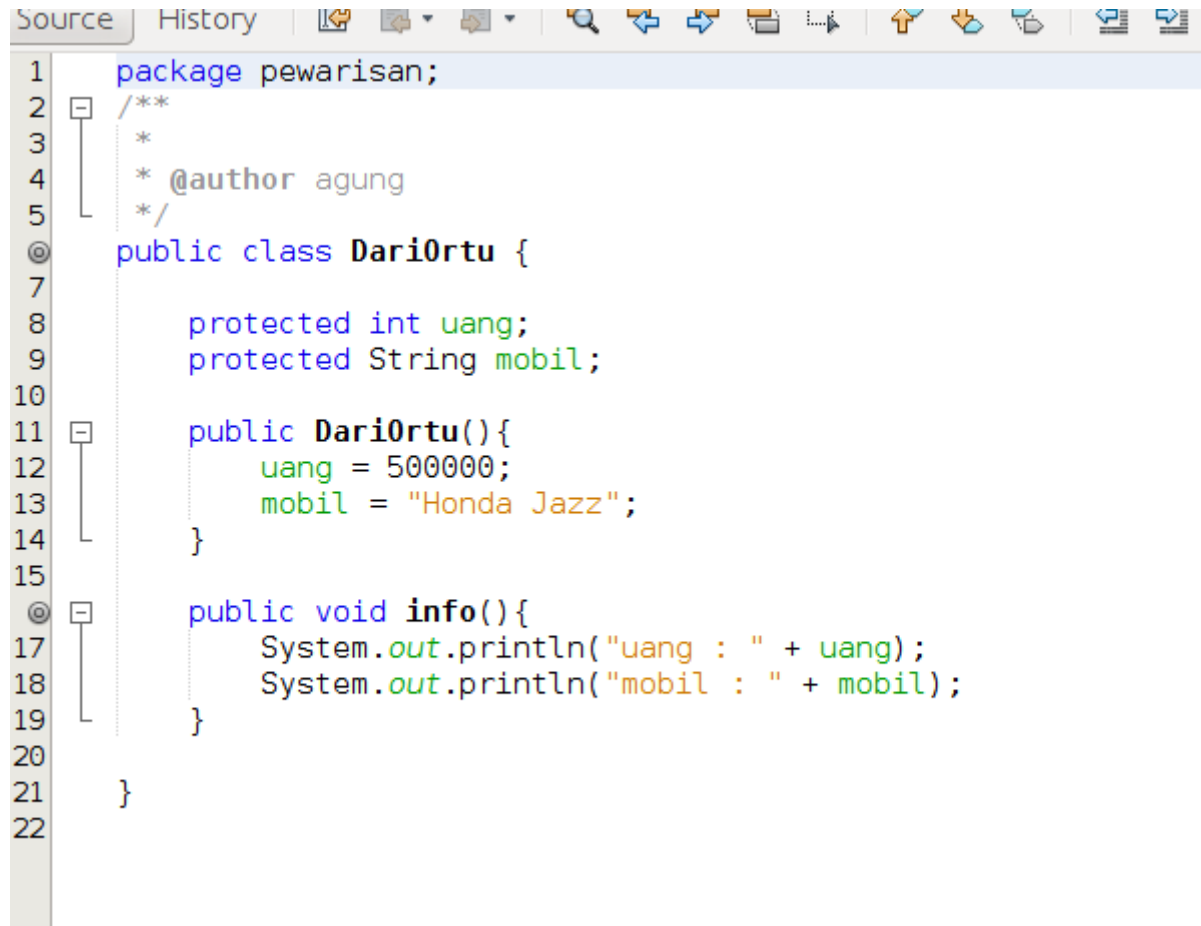
3. Inheritance

Inheritance / Pewarisan adalah suatu mekanisme yang memungkinkan fitur-fitur di suatu class diturunkan ke class lain.

Di java, class yang mewariskan dinamakan superclass, sedangkan class yang diwarisi disebut subclass.

Contoh program java inheritance

class DariOrtu sebagai Parent

A screenshot of a Java IDE window showing the source code for a class named 'DariOrtu'. The window has a title bar with 'Source' and 'History' tabs. The code is as follows:

```
1 package pewarisan;
2 /**
3  *
4  * @author agung
5  */
6 public class DariOrtu {
7
8     protected int uang;
9     protected String mobil;
10
11     public DariOrtu(){
12         uang = 500000;
13         mobil = "Honda Jazz";
14     }
15
16     public void info(){
17         System.out.println("uang : " + uang);
18         System.out.println("mobil : " + mobil);
19     }
20 }
21
22
```

Class MilikSendiri sebagai child

```

1 package pewarisan;
2 /**
3  *
4  * @author agung
5  */
6 public class MilikSendiri extends DariOrtu{
7     private String sepeda;
8     public MilikSendiri(){
9         sepeda = "Pacific x233";
10    }
11
12    public void info(){
13        System.out.println("uang : " + uang);
14        System.out.println("mobil : " + mobil);
15        System.out.println("sepeda : " + sepeda);
16    }
17 }
18

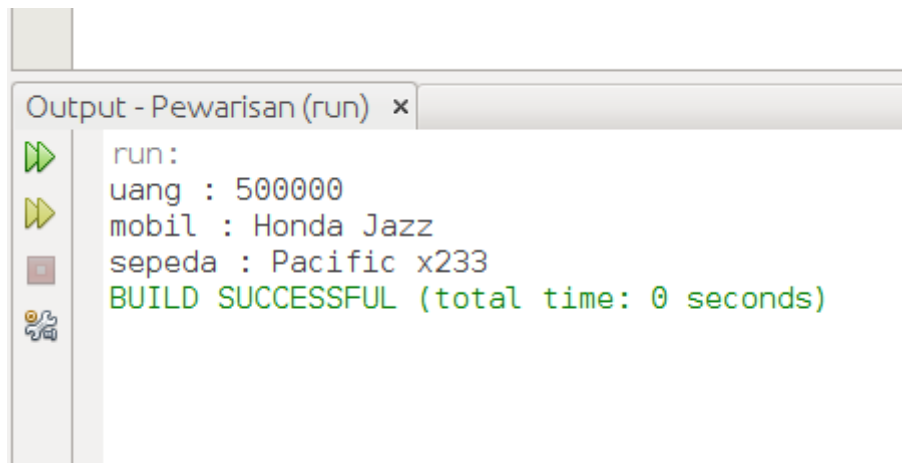
```

Driver class (Object)

```

Source History
1 package pewarisan;
2 /**
3  *
4  * @author agung
5  */
6 public class Pewarisan {
7     public static void main(String[] args) {
8
9         MilikSendiri bendaku = new MilikSendiri();
10
11         bendaku.info();
12     }
13 }
14
15

```



2, Inheritance & Konstruktor

class Pegawai sebagai Parent

```
1 package pewarisandua;
2 /**
3  *
4  * @author agung
5  */
6
7 public class Pegawai {
8     protected String nip;
9     protected String nama;
10
11     public Pegawai(String nip,String nama){
12         this.nip = nip;
13         this.nama = nama;
14     }
15
16     public void info(){
17         System.out.println("Nip : " + nip);
18         System.out.println("Nama : " + nama);
19     }
20
21 }
22
```

class PegawaiAsing sebagai child

```
Source History
1 package pewarisandua;
2 /**
3  *
4  * @author agung
5  */
6 public class PegawaiAsing extends Pegawai {
7     private String noPasport;
8     public PegawaiAsing(String nip,String nama,
9                          String noPasport){
10         super(nip,nama);
11         this.noPasport = noPasport;
12     }
13
14     public void info(){
15         super.info();
16         System.out.println("Pasport : " + noPasport);
17     }
18 }
19
```

Driver Class

```
source History
1 package pewarisandua;
2 /**
3  *
4  * @author agung
5  */
6 public class Pewarisandua {
7     public static void main(String[] args) {
8
9         PegawaiAsing a = new PegawaiAsing("111","Robert","A12345");
10
11         a.info();
12     }
13 }
14
15
```

```
Output - Pewarisandua (run) x
run:
Nip : 111
Nama : Robert
Pasport : A12345
BUILD SUCCESSFUL (total time: 0 seconds)
```