Jobsheet 09

Overloading and Overriding Method

1. Kompetensi

   Setelah menempuh pokok bahasan ini, mahasiswa mampu :

   1) Memahami konsep overloading dan overriding,
   2) Memahami perbedaan overloading dan overriding,
   3) Mengimplementasikan method overloading dan overriding.

2. Pendahuluan

   a) Method overloading adalah konsep dalam pemrograman java yang mengijinkan satu class mempunyai dua atau lebih method dengan **nama** yang sama akan tetapi mempunyai **argument** yang berbeda. Dan method ini terjadi pada waktu proses compilasi (bukan run-time).

      Perbedaan list argument pada method overloading bisa berupa :

      1) Jumlah dari parameter,
      2) Tipe data dari parameter,
      3) Sequence dari tipe data pada parameter.

   b) Method overriding terjadi jika class anak mempunyai method dengan **nama**, **argument** dan **tipe** yang sama pada class induknya. Method ini terjadi pada waktu run-time.

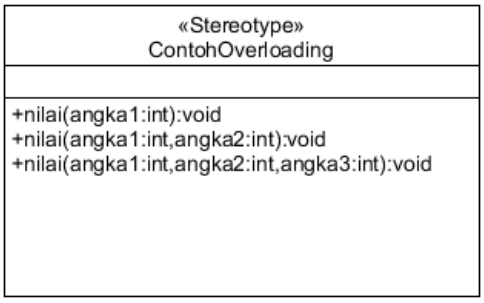
   c) Perbedaan method overloading dan overriding

      1) Overloading terjadi pada waktu compile – time, sedangkan overriding pada waktu run – time.
      2) Static method bisa di overloading, tetapi tidak di overriding.
      3) Overloading method berada pada class yang sama, sedangkan overriding method berada pada class induk dan class anak.
      4) Secara performa, overloading lebih baik daripada overriding.
      5) Private dan final bisa di overloading, tetapi tidak di overriding.

3. Percobaan

   Method overloading

   Contoh percobaan overloading dengan jumlah parameter yang berbeda.



   Implementasi :

```
public class ContohOverloading {
    public void nilai(int angka1){
        System.out.println("Nilai : "+angka1);
    }
    public void nilai(int angka1, int angka2){
        System.out.println("Nilai : "+angka1+" dan "+angka2);
```

```
        }
        public void nilai(int angka1, int angka2, int angka3){
            System.out.println("Nilai : "+angka1+" dan "+angka2+" dan
    "+angka3);
        }
}
```

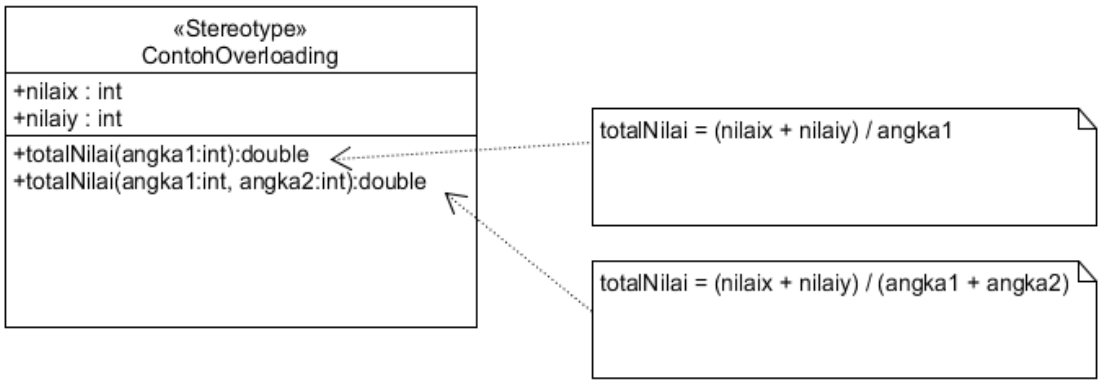
```
    public class Implementasi {
        public static void main(String[] args) {
            ContohOverloading ovr = new ContohOverloading();
            ovr.nilai(10);
            ovr.nilai(11, 12);
            ovr.nilai(13, 14, 15);
        }
}
```

Percobaan 1:

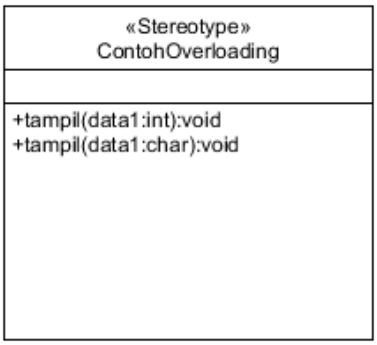
Implementasikan konsep overloading dari class diagram dibawah ini kedalam bahasa pemrograman java :

Langkah percobaan :

a) Buatlah class dengan nama ContohOverloading (atau bisa dengan nama yang lain "sesuai dengan keinginan kalian").
b) Buatlah dua attribute public dengan nama (nilaix dan nilaiy), bertipe integer.
c) Buatlah dua method , dengan ketentuan seperti berikut ini :
   Method pertama dan kedua mempunyai nama yang sama yaitu totalNilai, dan bertipe double (mempunyai return value dengan tipe double).
   Method pertama hanya mempunyai satu argument yaitu angka1 dengan tipe integer, sedangkan method kedua mempunyai dua argument (angka1 dan angka2) dan sama – sama bertipe integer.
   Return value dari method yang pertama dan kedua bisa dilihat pada note dari uml dibawah ini.
d) Buatlah main class untuk memanggil object dari class yang sudah anda buat sebelumnya.
e) Amati dan catat output dari percobaan.



Contoh percobaan overloading dengan tipe data yang berbeda.




```
public class ContohOverloading {
          public void tampil(int data1){
              System.out.println(data1);
          }
          public void tampil(char data1){
              System.out.println(data1);
          }
}
```

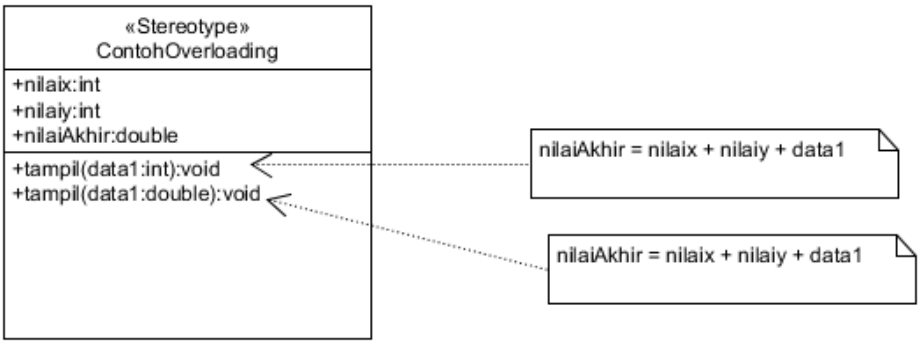
```
public class Implementasi {
          public static void main(String[] args) {
              ContohOverloading ovr = new ContohOverloading();
              ovr.tampil('x');
              ovr.tampil(1);
          }
}
```

Percobaan 2:

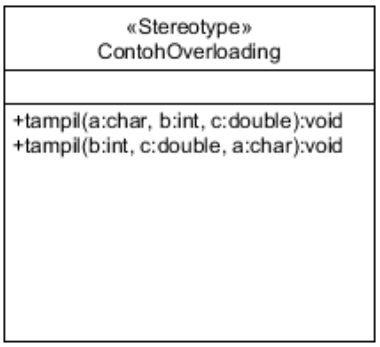
Implementasikan konsep overloading dari class diagram dibawah ini kedalam bahasa pemrograman java :

Langkah percobaan :

a) Buatlah class dengan nama ContohOverloading (atau bisa dengan nama yang lain "sesuai dengan keinginan kalian"),
b) Buatlah tiga attribute public , dengan nama attribute nilaix, nilaiy dan nilaiAkhir. Tipe data untuk setiap attribute bisa dilihat pada uml class diagram.
c) Buatlah dua method public dengan tipe void.
d) Method pertama dan kedua hanya memiliki perbedaan pada tipe argument (bukan banyaknya argument). Argument method yang pertama bertipe integer , sedangkan yang kedua bertipe double.
e) Buatlah main class untuk memanggil object dari class yang sudah anda buat sebelumnya.



Contoh percobaan overloading secara sequence tipe datanya pada argument.




```
public class ContohOverloading {
          public void tampil(char a, int b, double c){
              System.out.println("Urutan Pertama :");
              System.out.println("Char    : "+a);
              System.out.println("Int     : "+b);
              System.out.println("Double  : "+c);
          }
          public void tampil(int b, double c, char a){
              System.out.println("Urutan Kedua   :");
              System.out.println("Int     : "+b);
              System.out.println("Double  : "+c);
              System.out.println("Char    : "+a);
          }
}
```

```
public class Implementasi {
          public static void main(String[] args) {
              ContohOverloading3 ovr = new ContohOverloading3();
              ovr.tampil('A', 2, 3.7);
              ovr.tampil(2, 3.7, 'A');
          }
}
```

Note :
Hal yang perlu diperhatikan dalam implementasi dari method overloading adalah :

1) Method tidak diperbolehkan mempunyai argument dengan jumlah yang sama dan dengan tipe data yang sama, walaupun nama argument berbeda. Contoh :

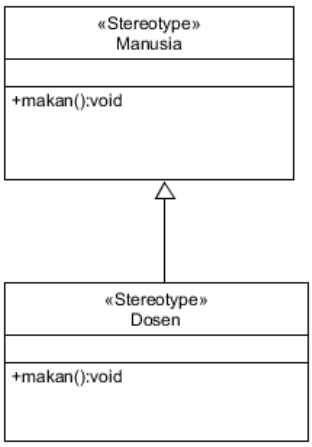
```
public class ContohSalah {
            public int myMethod(int a, int b){
                  return a * b;
            }
            public int myMethod(int num1, int num2){
                  return num1 * num2;
      }
}
```

2) Method juga tidak diperbolehkan mempunyai argument dengan jumlah yang sama, tipe data yang sama, walaupun dengan nama argument yang berbeda dan tipe return value yang berbeda pula. Contoh :

```
public class ContohSalah {
            public int myMethod(int a, int b){
                  return a * b;
            }
            public double myMethod(int num1, int num2){
                  return num1 * num2;
            }
}
```

Method Overriding

Contoh percobaan method overriding




```
public class Manusia {
    public void makan(){
        System.out.println("Manusia harus makan");
    }
}
```

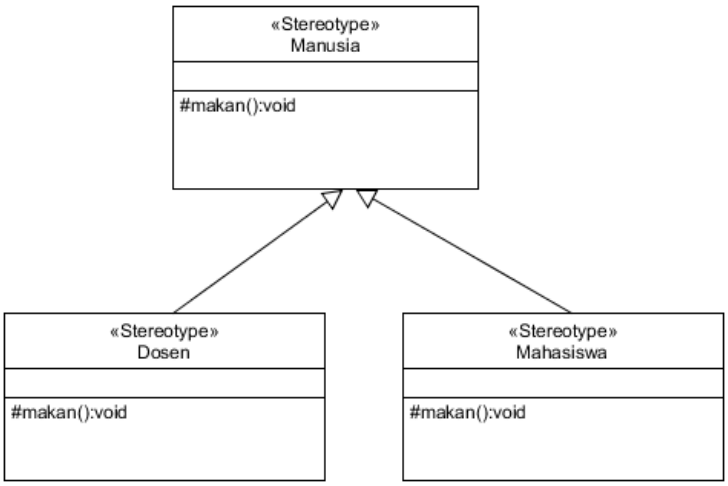
```
public class Dosen extends Manusia{
    public void makan(){
        System.out.println("Dosen harus makan");
    }
}
```

```
public class Implementasi {
    public static void main(String[] args) {
        Dosen dosen = new Dosen();
        dosen.makan();
    }
}
```

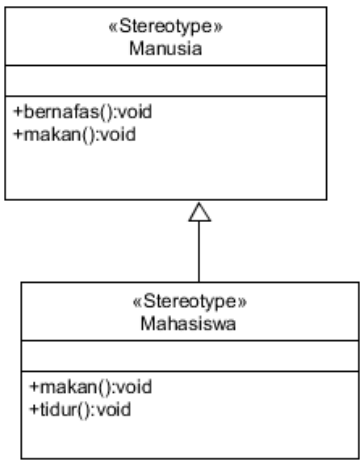
Percobaan 3:

Implementasikan konsep overriding dari class diagram berikut ini :
Langkah percobaan :

1) Buatlah parent class dengan nama manusia yang didalamnya terdapat method makan dengan akses modifier protected dan bertipe void.
2) Buatlah dua child class dengan nama Mahasiswa & Dosen, yang keduanya mempunyai method makan dengan akses modifier protected dan bertipe void.



Contoh method overriding dengan menggunakan tehnik dynamic method dispatch (menggunakan reference dari class parent pada object child nya)




```
public class Manusia {
    public void bernafas(){
        System.out.println("Manusia harus bernafas");
    }
```

```
    public void makan(){
        System.out.println("Manusia harus makan");
    }
}
```

```
public class Mahasiswa extends Manusia{
    public void makan(){
        System.out.println("Mahasiswa harus makan");
    }
    public void tidur(){
        System.out.println("Mahasiswa sering tidur");
    }
}
```

```
public class Implementasi {
    public static void main(String[] args) {
        Manusia mns = new Mahasiswa();
        mns.bernafas();
        mns.makan();
    }
}
```

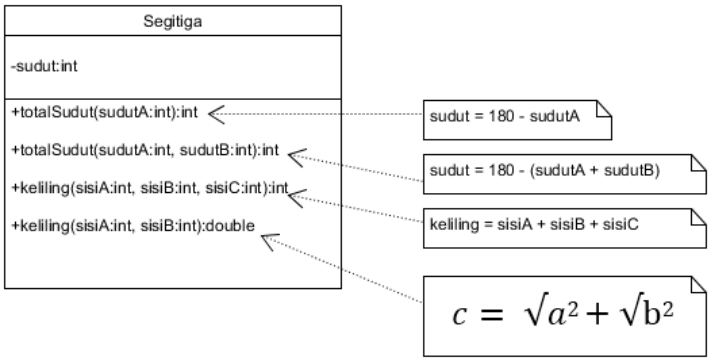
Note :

Pada tehnik ini tidak diperbolehkan untuk melakukan invokasi (pemanggilan method) pada method **tidur** pada class mahasiswa.

4. Latihan

Overloading

Implementasikan konsep overloading pada class diagram dibawah ini :



Overriding

Implementasikan class diagram dibawah ini dengan menggunakan tehnik dynamic method dispatch :