Pertemuan 7 :
Prinsip Dasar Pemrograman Berbasis Obyek

Tessy Badriyah, SKom. MT.

Pokok Bahasan

- Information Hiding
- Encapsulation
- Constructor
- Overloading constructor

Review : Ciri OOP

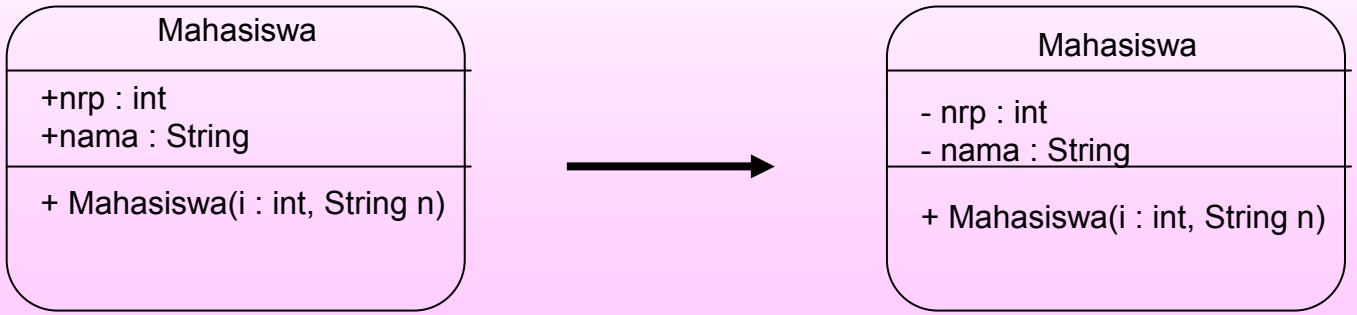
- Suatu program disebut dengan pemrograman berbasis obyek (OOP) karena terdapat :
  - Encapsulation (pembungkusan)
  - Inheritance (pewarisan)
  - Polymorphism (polimorfisme – perbedaan bentuk)

Mengapa harus di-Encapsulasi ?

- Karena ada informasi yang harus disembunyikan.
- Artinya : anggota dari suatu class bisa dilindungi agar tidak bisa diakses dari luar
- Caranya adalah dengan mengganti modifier yang tadinya public menjadi private

Penyembunyian Informasi

- Penyembunyian informasi atau information hiding adalah mengganti metode akses (modifier) dari public menjadi private
- Contoh : atribut nrp dan nama yang tadinya dideklarasikan public menjadi private
- Perubahan dalam UML Class Diagram :

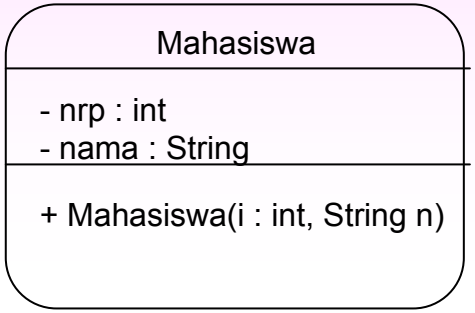


Konsekuensi dari Encapsulasi

- Akibat dari encapsulation :
  - Detail informasi dari suatu class dapat dilindungi
  - Sehingga untuk mengakses informasi tersebut diperlukan suatu perantara
  - Perantara yang diperlukan berupa method yang bisa diakses oleh user

Implementasi UML Class Diagram dengan Information Hiding

- UML Class diagram untuk Mahasiswa setelah di-encapsulasi :

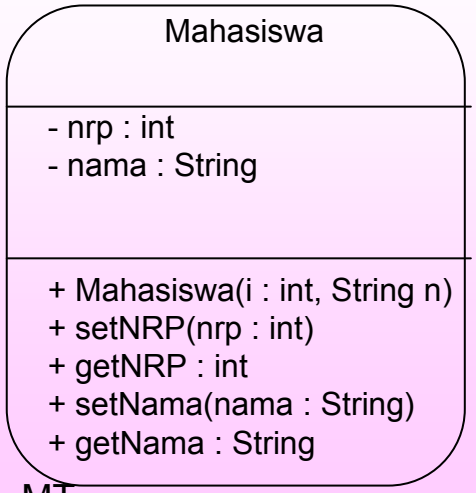


- Implementasi dari UML Class diagram diatas :

```
public class Mahasiswa {
   private int nrp;
   private String nama;
   public Mahasiswa(int i, String n) {
      this.nrp=i;
      this.nama=n;
   }
```

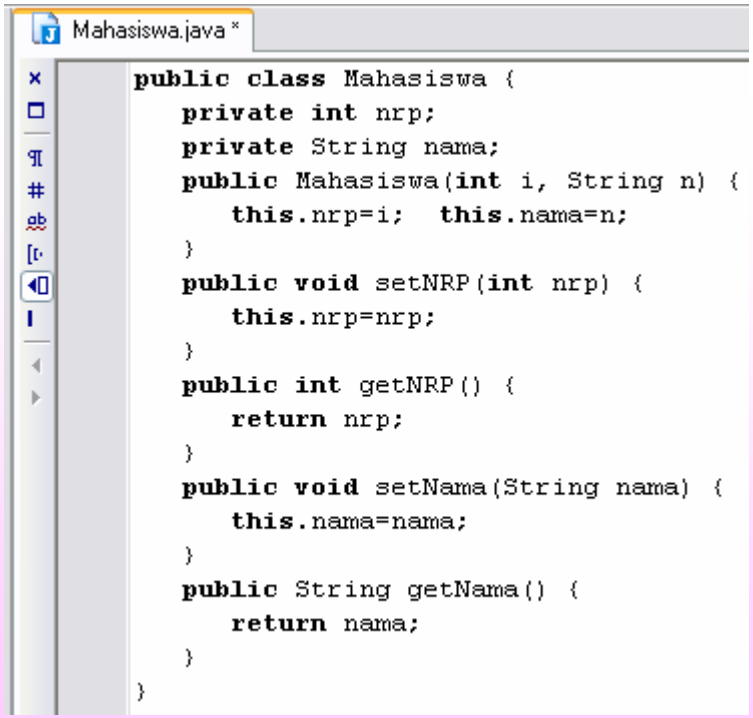
Method untuk Mengakses Informasi

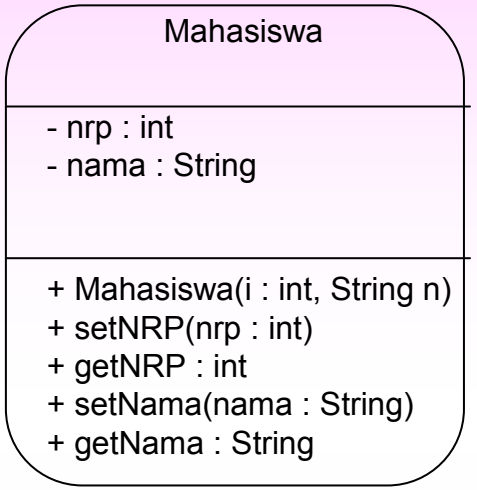
- Untuk mengakses informasi, diperlukan dua buah method untuk setiap atribut :
  - Method untuk mengeset nilai atribut
  - Method untuk mengambil nilai atribut
- Sehingga UML Class Diagram menjadi :



Implementasi UML Class Diagram dengan Enkapsulasi

- Implementasi dari :





Percobaan 1

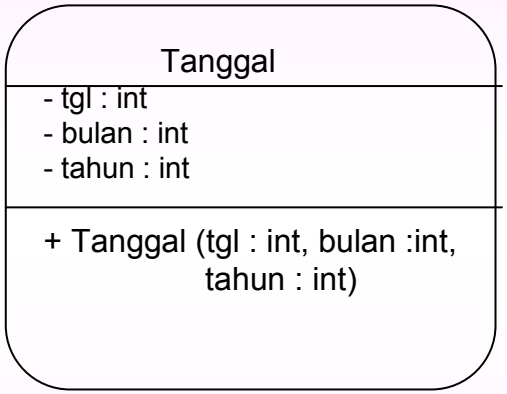
- Uji UML Class Diagram yang telah diencapsulasi !

Mahasiswa.java | TesMahasiswa.java

```
class TesMahasiswa {
public static void main(String [] args) {
   int nomer;
   String nm;
   Mahasiswa siswa = new Mahasiswa(123,"A");
   System.out.println(siswa.getNRP());
   System.out.println(siswa.getNama());
   siswa.setNama("Ani");
   siswa.setNRP(111);
   System.out.println(siswa.getNRP());
   System.out.println(siswa.getNama());
 }
}
```

Percobaan 2

- Implementasikan Class Tanggal berikut :



Tanggal.java

```
public class Tanggal {
        private int tgl;
        private int bulan;
        private int tahun;
        public Tanggal(int tgl, int bulan, int tahun) {
            this.tgl=tgl;
            this.bulan=bulan;
            this.tahun=tahun;
        }
}
```

TUGAS

Tugas

1. Buatlah program untuk menguji Class Diagram pada percobaan 2
2. Dari UML Class diagram Tanggal pada Percobaan 2, implementasikan class yang memenuhi konsep Overloading Constructor berikut :