



# Class dan Object

**PEMROGRAMAN BERORIENTASI OBJEK**

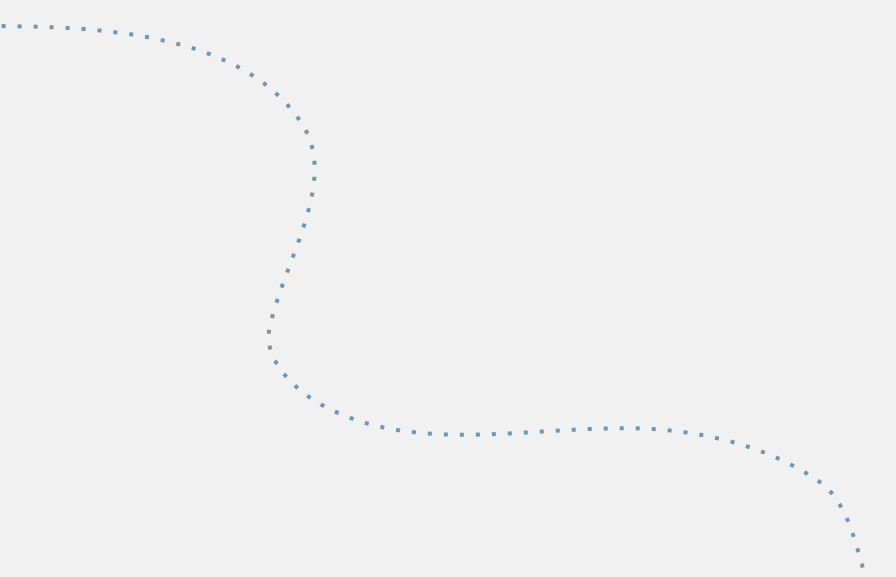
**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**



## — OBJECT

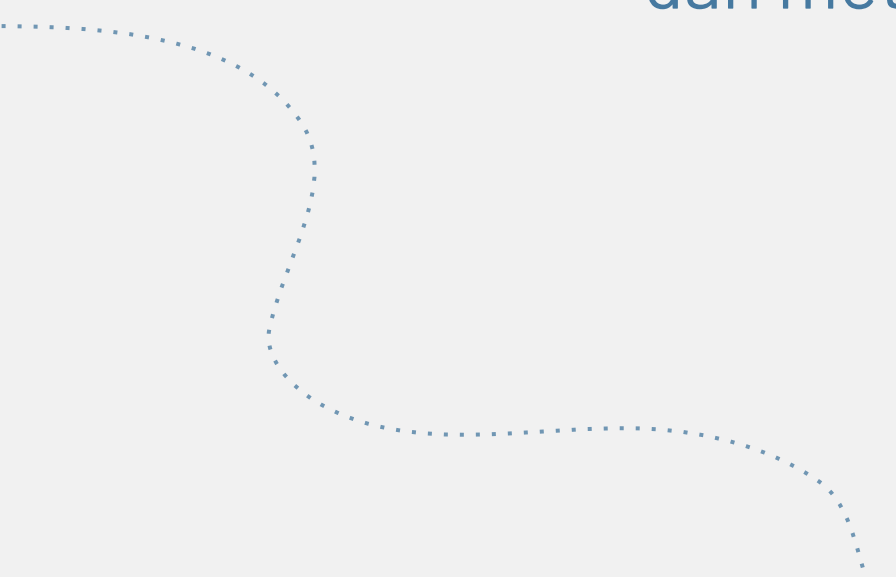
Objek adalah representasi dari setiap entitas yang terlibat dalam sistem





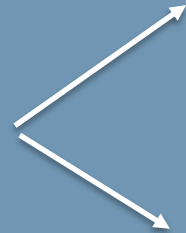
## — **CLASS**

Blueprint/template/cetakan yang mendefinisikan karakteristik (atribut dan method) objek pada class





**KARAKTERISTIK**



**Apa yang dimiliki?**

**Apa yang bisa dilakukan?**



## Apa yang dimiliki sebuah objek mahasiswa?



- Nama
- NIM
- Tanggal Lahir
- Jenis Kelamin
- Alamat



### —**ATRIBUT:**

Variabel/ciri/status/sifat yang **dimiliki** oleh suatu objek

## Apa yang bisa dilakukan oleh/terhadap objek mahasiswa?



- Memilih mata kuliah
- Melihat nilai
- Mengajukan cuti akademik



### —METHOD:

Prosedur/fungsi/perilaku/  
proses yang bisa **dilakukan**  
oleh/terhadap suatu objek

# Aturan penulisan nama class

- Berupa kata benda
- Diawali dengan **HURUF KAPITAL**,
- Jika terdiri dari lebih dari 1 kata, maka setiap kata **disambungkan**, dan huruf awal dari tiap kata menggunakan **HURUF KAPITAL**
- Contoh: Mahasiswa, TenagaKependidikan

# Implementasi Class

```
<modifier> class <nama_class> {  
  
    //deklarasi atribut dan method  
  
}
```

- Untuk mendeklarasikan suatu class, digunakan keyword class lalu diikuti dengan nama class yang akan dibuat, e.g:

```
public class Mahasiswa{  
  
  
  
}
```



## Aturan penulisan nama atribut

- Berupa **kata benda** atau **kata sifat**,
- Diawali dengan **HURUF KECIL**
- Jika terdiri dari lebih dari 1 kata, maka setiap kata **disambungkan**, dan huruf awal dari tiap kata menggunakan **HURUF BESAR**.
- Contoh: cetakBiodata()

# Deklarasi Atribut

- Untuk melakukan deklarasi atribut:

```
<modifier> <tipe> <nama_atribut> ;
```

- Exp :

```
public String nim;
```

```
public String nama;
```

```
public String alamat;
```

## Aturan penulisan nama method

- Berupa **kata kerja**,
- Diawali dengan **HURUF KECIL**,
- Jika terdiri dari lebih dari 1 kata, maka setiap kata **disambungkan**, dan huruf awal dari tiap kata menggunakan **HURUF BESAR**.

# Deklarasi Method

- Untuk melakukan pendeklarasian method dapat dilakukan dengan sintaks sebagai berikut :

```
<modifier> <return_type> <nama method>(param1, param2, ...)
{
    //statements
}
```

# Return Type

- Method dengan **return type void**, berarti tidak memiliki return value, sehingga tidak memerlukan kata kunci **return** di dalamnya.
- Method dengan **return type selain void**, berarti memerlukan return value, sehingga harus ada nilai yang dikembalikan dengan kata kunci **return** di dalamnya

```
public void sayHello(){  
    System.out.println("Hello World!!");  
}
```

TIDAK PERLU  
RETURN /  
TIDAK ADA NILAI  
KEMBALIAN

```
public int tambah (int a, int b){  
    int hasil = a+b;  
    return hasil;  
}
```

- TIPE DATA METHOD INT, BERARTI METHOD TSB HARUS MENGEMBALIKAN NILAI INT
- HARUS ADA RETURN

# Constructor

- Constructor → method istimewa yang digunakan untuk melakukan instansiasi objek (membuat objek baru)
- Istimewa:
  1. Nama method sama dengan nama class
  2. Tidak memiliki return type
  3. Hanya bisa dijalankan/dipanggil pada proses instansiasi
- Jika sebuah class tidak memiliki constructor secara eksplisit, maka secara default Java compiler akan menyediakan constructor tanpa parameter

# Constructor

```
public class Donat{  
    public String topping;  
}
```



```
public class Donat{  
    public String topping;  
  
    public Donat(){  
    }  
}
```



# Instansiasi Object

- Object adalah instansiasi dari sebuah class
- Instansiasi dilakukan dengan memanggil constructor menggunakan keyword **new**

```
NamaClass namaObject = new NamaClass();
```

- Contoh:

```
Mahasiswa mahasiswa1 = new Mahasiswa();
```

```
Mahasiswa ani = new Mahasiswa();
```

# Contoh

```
public class Mahasiswa{  
    public String nim;  
    public String nama;  
    public String alamat;  
  
    public void cetakBiodataMahasiswa(){  
        System.out.println("Biodata Mahasiswa");  
        System.out.println("Nim      : " + nim);  
        System.out.println("Nama      : " + nama);  
        System.out.println("Alamat    : " + alamat);  
    }  
}
```

# Instansiasi Objek

```
public class DemoMahasiswa{  
    public static void main(String[] args){  
        Mahasiswa mhs = new Mahasiswa();  
        mhs.nim = "14324";  
        mhs.nama = "Very Sugiarto";  
        mhs.alamat = "Malang";  
        mhs.cetakBiodataMahasiswa();  
    }  
}
```

# Constructor

```
public class Donat{  
    public String topping;  
}
```



```
public class Donat{  
    public String topping;  
  
    public Donat(){  
    }  
}
```

# Instansiasi Objek

```
public class DemoDonat{
```

```
    public static void main(String[] args){
```

```
        Donat donat1 = new Donat();
```

```
        donat1.topping = "Strawberry sprinkles";
```

```
    }
```

```
}
```



# Constructor berparameter

- Constructor berparameter digunakan untuk menginstansiasi objek baru dengan kondisi/nilai tertentu.
- Java compiler tidak akan menyediakan default constructor (tanpa parameter) jika constructor lain sudah dibuat oleh programmer

```
public class Donat{  
    public String topping;  
  
    Donat(String selectedTopping){  
        topping = selectedTopping  
    }  
}
```

# Instansiasi Objek

```
public class DemoDonat{  
    public static void main(String[] args){  
        Donat donat1 = new Donat("Strawberry sprinkles");  
    }  
}
```



```
public class Donat{  
    public String topping;  
}
```



```
public class Donat{  
    public String topping;  
  
    public Donat(){  
    }  
}
```

Jika sebuah class tidak memiliki constructor secara eksplisit, maka secara default Java compiler akan menyediakan constructor tanpa parameter

```
public class DemoDonat{  
    public static void main(String[] args){  
        Donat donat1 = new Donat();  
        donat1.topping = "Strawberry sprinkles";  
    }  
}
```



Java compiler tidak akan menyediakan default constructor (tanpa parameter) jika constructor lain sudah dibuat oleh programmer

```
public class Donat{  
    public String topping;  
  
    Donat(String selectedTopping){  
        topping = selectedTopping;  
    }  
}
```



```
public class Donat{  
    public String topping;  
  
    Donat(String selectedTopping){  
        topping = selectedTopping;  
    }  
}
```

# Class Diagram

NamaClass

atribut1: tipeData1  
atribut2: tipeData2

method1(parameter1:tipeData1): returnType1  
method2(): returnType2

```

public class Sepeda {
    String merk;
    int kecepatan;

    public int tambahKecepatan(float increment) {
        kecepatan += increment;
        return kecepatan;
    }

    public int kurangiKecepatan(float decrement) {
        kecepatan -= decrement;
        return kecepatan;
    }

    public void cetakInfo() {
        System.out.println("Merk: " + merk);
        System.out.println("Kecepatan: " + kecepatan);
    }
}

```

Sepeda
merk: String kecepatan: int
tambahKecepatan(increment:int): int kurangiKecepatan(decrement:int): int cetakInfo(): void

```

public class Sepeda {
    String merk;
    int kecepatan;

    public int tambahKecepatan(float increment) {
        kecepatan += increment;
        return kecepatan;
    }

    public int kurangiKecepatan(float decrement) {
        kecepatan -= decrement;
        return kecepatan;
    }

    public void cetakInfo() {
        System.out.println("Merk: " + merk);
        System.out.println("Kecepatan: " + kecepatan);
    }
}

```

Sepeda
merk: String kecepatan: int
tambahKecepatan(): int kurangiKecepatan(): int cetakInfo(): void

# Latihan

- Buatlah class diagram untuk setiap class yang dibuat pada Tugas 01
- Perhatikan best practice penamaan class, method, dan atribut (kata benda/sifat/kerja dan huruf kapital)