

# PEMROGRAMAN BERORIENTASI OBJECT

Konsep Dasar OOP

# Outline

---

- PBO vs Struktural
- Konsep dasar PBO

# Matakuliah PBO

- Pemrograman Berbasis Objek (2 SKS/4x50 menit)
- Praktikum Pemrograman Berbasis Objek (2 SKS/4x50 menit)
- Capaian Pembelajaran
  - ▣ Mampu membuat program dengan menggunakan prinsip-prinsip OOP menggunakan bahasa pemrograman Java

# Silabus

Pertemuan Ke-	Pembahasan
1	Pengantar Konsep Dasar OOP
2	Class dan Object
3	Enkapsulasi
4	Relasi Class
5	Kuis 1
6	Inheritance
7	Overriding dan Overloading
8	Ujian Tengah Semester (UTS)
9	Abstract Class
10	Interface
11	Polimorfisme
12	Kuis 2
13-16	OOP pada Bahasa pemrograman lain
17	Ujian Akhir Semester (UAS)



**Pemrograman  
Berorientasi Objek**

**Vs**

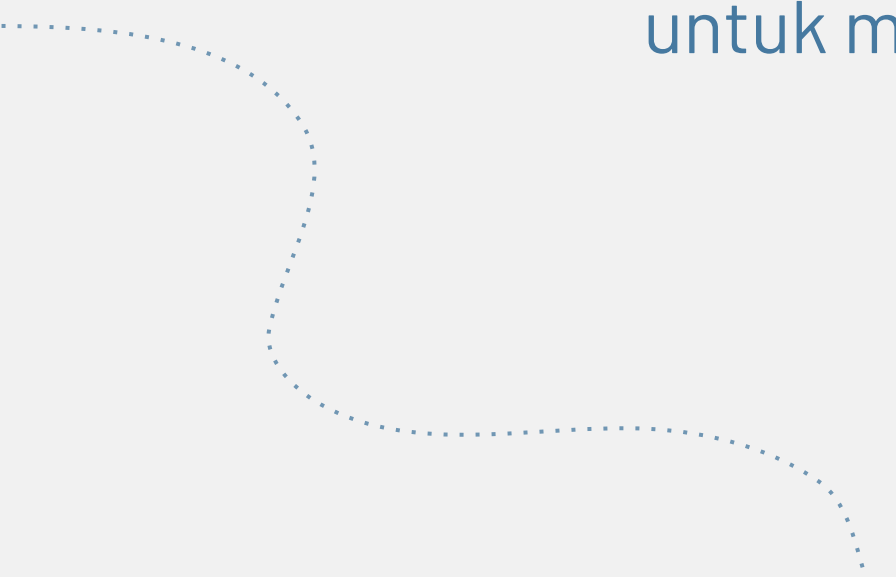
**Pemrograman  
Struktural**





## — PEMROGRAMAN STRUKTURAL

Paradigma pemrograman dengan konsep bahwa program merupakan urutan langkah-langkah yang dilakukan untuk menyelesaikan suatu masalah



```
public static void main(String[] args)
{
    String merek;
    int kecepatan, gear;

    merek = "Poligone";
    kecepatan = 10;
    gear = 1;

    kecepatan = tambahKecepatan(kecepatan, 10);

    System.out.println("Merek: " + merek);
    System.out.println("Kecepatan: " + kecepatan);
}

public static int tambahKecepatan(int kecepatan, int increment)
{
    kecepatan += increment;

    return kecepatan;
}

public static int kurangiKecepatan(int kecepatan, int decrement)
{
    kecepatan -= decrement;

    return kecepatan;
}
```

# Pemrograman Struktural

- Bagaimana jika ada dua sepeda di game?
  - Tambahkan variabel merek2, kecepatan2, gear2
  - Coba manipulasi nilai-nilai variabelnya kemudian tampilkan ke layar
- Kode program →





```
public static void main(String[] args)
{
    String merek, merek2;
    int kecepatan, kecepatan2, gear, gear2;

    merek = "Poligone";
    kecepatan = 10;
    gear = 1;

    merek2 = "Wiim Cycle";
    kecepatan2 = 15;
    gear2 = 3;

    kecepatan = tambahKecepatan(kecepatan, 10);
    kecepatan2 = tambahKecepatan(kecepatan2, 5);

    System.out.println("Merek: " + merek);
    System.out.println("Kecepatan: " + kecepatan);

    System.out.println("Merek: " + merek2);
    System.out.println("Kecepatan: " + kecepatan2);
}

public static int tambahKecepatan(int kecepatan, int increment)
{
    kecepatan += increment;

    return kecepatan;
}

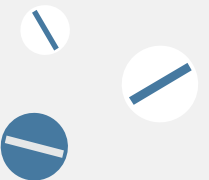
public static int kurangiKecepatan(int kecepatan, int decrement)
{
    kecepatan -= decrement;

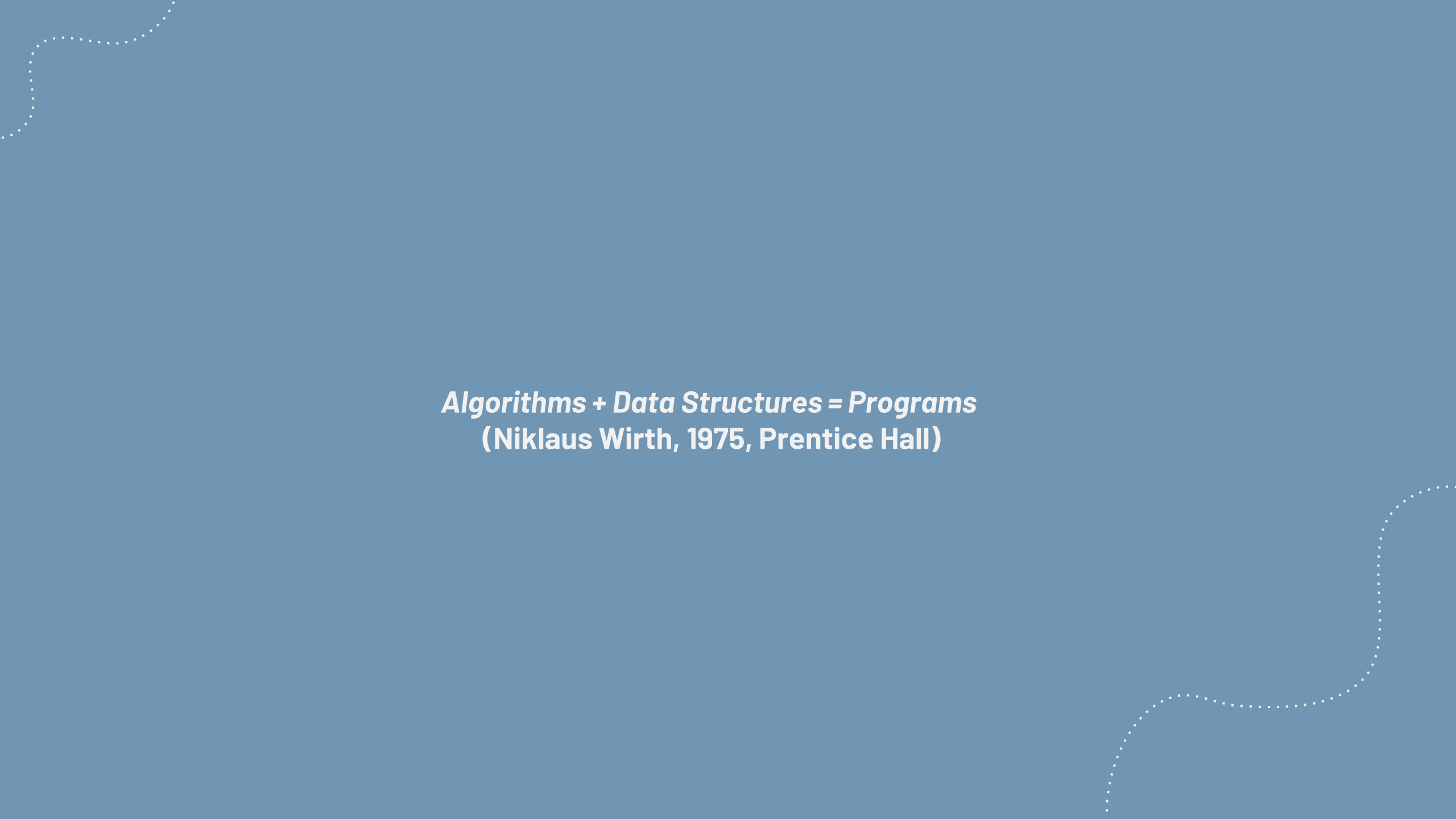
    return kecepatan;
}
```

12  
baris

# Pemrograman Struktural

- Bagaimana jika ada sepuluh sepeda?
  - Tambahkan variabel merek3, kecepatan3, gear3 .....  
merek10, kecepatan10, gear10





***Algorithms + Data Structures = Programs***  
**(Niklaus Wirth, 1975, Prentice Hall)**



## — PEMROGRAMAN BERORIENTASI OBJEK

Paradigma pemrograman yang berfokus pada objek

Data + Algorithm



The background features two decorative dotted lines, one in the top-left and one in the top-right, both curving downwards. There are also two light blue plus signs: one in the top-right area and one in the bottom-left area.

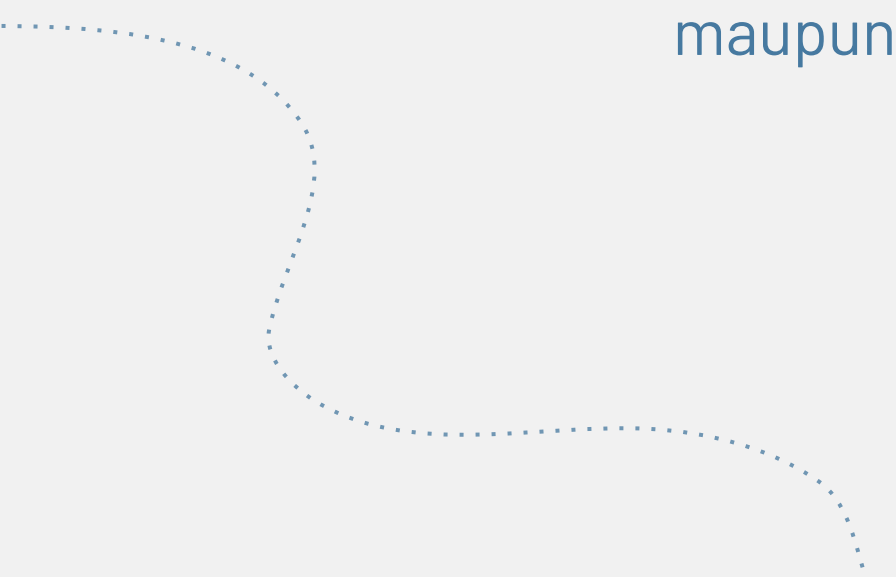
# CLASS & OBJECT



## — OBJEK

Siapa dan apa saja yang terlibat dalam business process?

Objek adalah representasi dari setiap entitas yang terlibat dalam sistem (baik yang nyata maupun tidak nyata)



# SISTEM INFORMASI AKADEMIK

Aljabar Linear



AKUNTANSI



Kalkulus 2



Analisis Numerik  
Matematika Diskrit



TEKNIK SIPIL  
TEKNIK KIMIA



AKUNTANSI



# SISTEM INFORMASI **AKADEMIK**



Analisis Numerik  
Matematika Diskrit  
Aljabar Linear  
Kalkulus 2



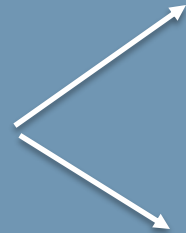
AKUNTANSI  
TEKNIK INFORMATIKA  
TEKNIK SIPIL  
TEKNIK KIMIA







**KARAKTERISTIK**



**Apa yang dimiliki?**

**Apa yang bisa dilakukan?**



## Apa yang dimiliki sebuah objek mahasiswa?



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



### —**ATRIBUT:**

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



	productID	productName	categoryName	supplier
▶	1	Chai	Beverages	Specialty Biscuits, Ltd.
	2	Chang	Beverages	Exotic Liquids
	3	Aniseed Syrup	Condiments	Exotic Liquids
	4	Chef Anton's Cajun Seasoning	Condiments	New Orleans Cajun Delights
	5	Chef Anton's Gumbo Mix	Condiments	New Orleans Cajun Delights
	6	Grandma's Boysenberry Spread	Condiments	Grandma Kelly's Homestead
	7	Uncle Bob's Organic Dried Pears	Produce	Grandma Kelly's Homestead
	8	Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead
	9	Mishi Kobe Niku	Meat/Poultry	Tokyo Traders
	10	Ikura	Seafood	Tokyo Traders

## 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

# SISTEM INFORMASI AKADEMIK



Analisis Numerik  
Matematika Diskrit  
Aljabar Linear  
Kalkulus 2



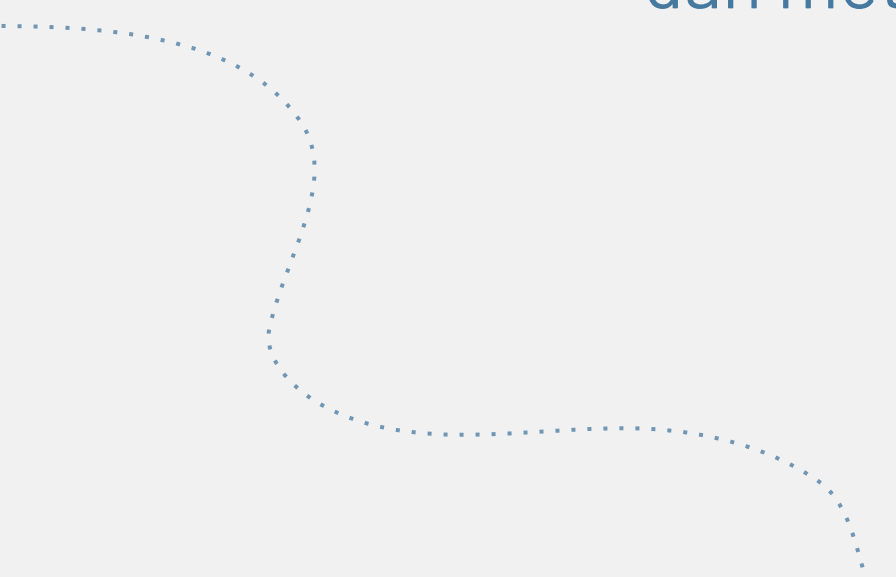
AKUNTANSI  
TEKNIK INFORMATIKA  
TEKNIK SIPIL  
MATEMATIKA





## — **CLASS**

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



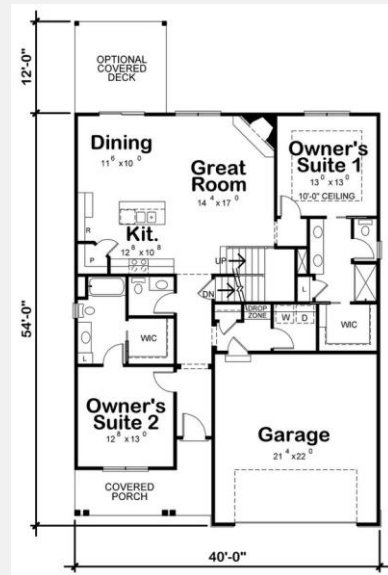
## Class Donat



## 6 Objek pada Class Donat



## Class Rumah



## 3 Objek pada Class Rumah





```
public class Sepeda {
    public String merk;
    public int kecepatan,gear;

    public Sepeda(String m, int k,int g){
        merk = m;
        kecepatan = k;
        gear = g;
    }

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

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

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

    public static void main(String[] args) {
        // sepeda pertama
        Sepeda spd1 = new Sepeda("Poligon",10,1);

        spd1.tambahKecepatan(10);
        spd1.info();

        //sepeda kedua
        Sepeda spd2 = new Sepeda("Wim Cycle", 15,3);

        spd2.tambahKecepatan(5);
        spd2.info();
    }
}
```

6  
baris

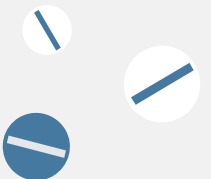
# Kesimpulan

- Struktural
  - Program dipecah ke dalam fungsi/prosedur
  - Perubahan fitur → kemungkinan mengganggu keseluruhan program
- Object Oriented
  - Program dipecah ke dalam object
  - Didalamnya terdapat state dan behavior
  - Perubahan fitur → tidak mengganggu keseluruhan program



# Aturan Penulisan Nama Class

- Berupa **kata benda**
- Menggunakan Pascal Case
  - 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



# Deklarasi Class

- Syntax:

```
class <NamaClass>{  
    .....  
}
```

- Contoh:

```
class Mahasiswa{  
    .....  
}
```



# Aturan Penulisan Nama Atribut

- Berupa **kata benda** atau **kata sifat**
- Menggunakan Camel Case
  - Diawali dengan **HURUF KECIL**
  - Jika terdiri dari lebih dari 1 kata, maka setiap kata **disambungkan**, dan huruf awal dari tiap kata menggunakan **HURUF KAPITAL**
- Contoh: discontinued, tanggalLahir



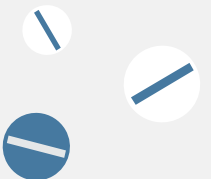
# Deklarasi Atribut

- Syntax:  
**<tipeData> <namaAtribut>**
- Contoh:  
**String namaLengkap;**  
**boolean discontinued;**  
**Date tanggalLahir;**



# Aturan Penulisan Nama Method

- Berupa **kata kerja**
- Menggunakan Camel Case
  - Diawali dengan **HURUF KECIL**
  - Jika terdiri dari lebih dari 1 kata, maka setiap kata **disambungkan**, dan huruf awal dari tiap kata menggunakan **HURUF KAPITAL**
- Contoh: `getKeliling()`, `displayInfo()`



# Deklarasi Method

- Syntax:

```
<returnType> <namaMethod>(){  
    .....  
}
```

- Seperti function pada umumnya
  - Method dapat memiliki parameter atau tidak
  - Method dapat memiliki return value atau tidak





# Contoh

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



# Constructor

- Constructor adalah method istimewa yang digunakan untuk melakukan instansiasi objek (membuat objek baru)
- Keistimewaan:
  - Nama method sama dengan nama class
  - Tidak memiliki return type
  - 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



# Instansiasi Objek

- Object adalah hasil dari instansiasi dari sebuah class
- Instansiasi object → pembuatan object baru dan inisiasi nilai untuk atribut
- Instansiasi dilakukan dengan memanggil constructor menggunakan keyword **new**

- Syntax:

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

- Contoh:

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

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



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

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



```
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";  
    }  
}
```



# 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 developer



# Constructor Berparameter

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

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



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

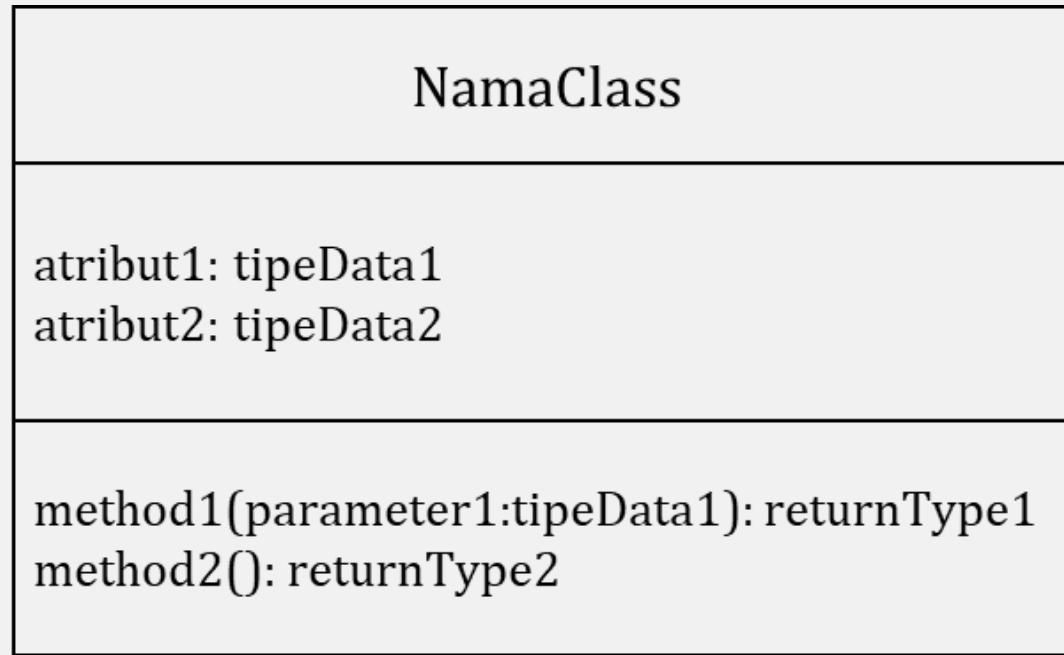
```
public class Donat{  
    public String topping;  
  
    Donat(String selectedTopping){  
        topping = selectedTopping;  
    }  
}
```



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



# Class Diagram



```

public class Sepeda {
    String merk;
    int kecepatan;

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

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

    public int kurangiKecepatan(int 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

# Referensi

- Horstmann, C. S., & Cornell, G. (2007). Core Java Volume I–Fundamentals, Eighth Edition. Network Circle, Santa Clara: Prentice Hall.
- Horstmann, C. S., & Cornell, G. (2008). Core Java Volume II–Advanced Features, Eighth Edition. Network Circle, Santa Clara: Prentice Hall.
- <https://www.javatpoint.com/java-oops-concepts>
- Dapat di download di <http://libgen.io/>



# Latihan

- Carilah objek apa saja di dunia nyata sebanyak 4 jenis (tidak harus berhubungan)
- Untuk setiap jenis object:
  - Tentukan nama classnya
  - Tentukan minimal 4 state/atribut dan 3 behavior/method
  - Buatlah class diagram dengan memperhatikan best practice penamaan class, method, dan atribut (kata benda/sifat/kerja dan huruf kapital)

