



Bootcamp Java 297

Object-Oriented Programming (OOP)

Pengenalan OOP



Object Oriented Programming (**OOP**) atau Pemprograman Berbasis Objek (**PBO**) merupakan sebuah paradigma yang berorientesikan **Objek**.

OOP merupakan serangkaian fungsi dan atribut yang dibungkus dalam suatu **Objek** dan **Class**.



Kelas



Kelas adalah prototype atau blueprint atau rancangan dari suatu obyek

```
• • •  
class Animal {  
    //Any attributes and methods  
}
```

Obyek

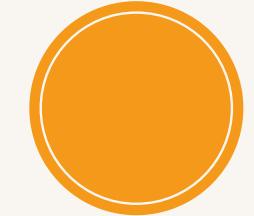


Obyek adalah manifestasi atau instance dari kelas.

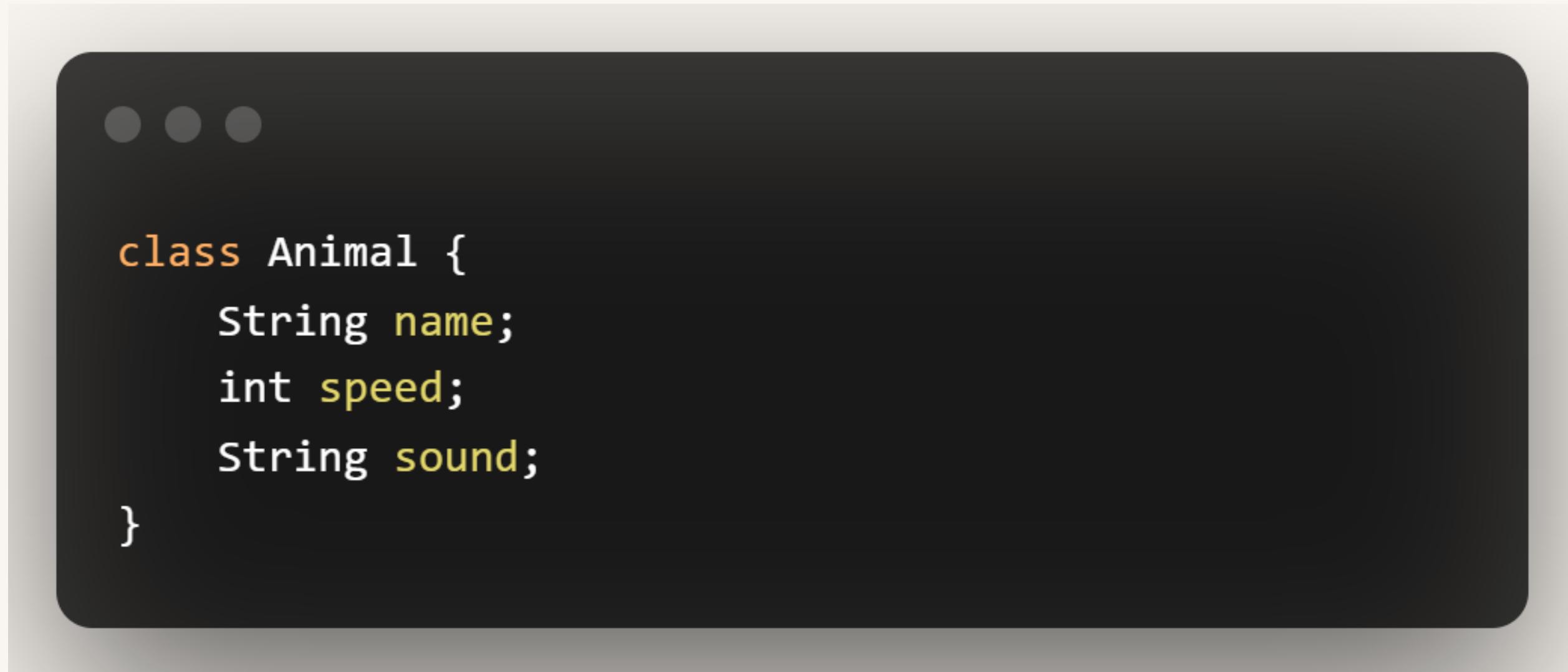
```
public class Codeternity {
    public static void main(String [] args) {
        Animal cat = new Animal();
    }
}

class Animal {
    //Any attributes and methods
}
```

Atribut



Atribut adalah variabel pada kelas yang menyatakan properti atau karakteristik atau ciri dari suatu obyek.



```
class Animal {  
    String name;  
    int speed;  
    String sound;  
}
```

Mengakses Atribut

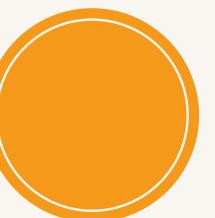


Untuk mengakses atribut diluar obyek dengan menuliskan **nama obyek** diikuti tanda **titik atau dot (.)** diikuti nama atribut. Contoh: lion.name.

```
public class Animal {
    String name;
    int speed;
    String sound;

    public static void main(String[] args) {
        Animal lion = new Animal();
        lion.name = "Lion";
        lion.speed = 90;
        lion.sound = "Rrrrr...";
        System.out.println("Animal: " + lion.name + ", speed: " + lion.speed + "km/h, sound: " +
lion.sound);

        Animal cat = new Animal();
        cat.name = "Cat";
        cat.speed = 50;
        cat.sound = "Meaw...";
        System.out.println("Animal: " + cat.name + ", speed: " + cat.speed + "km/h, sound: " +
cat.sound);
    }
}
```

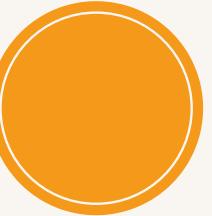


Method

Method adalah fungsi pada kelas yang menggambarkan perilaku dari obyek.

```
• • •  
  
class Animal {  
    String name;  
    int speed;  
    String sound;  
  
    public void run() {  
        System.out.println(name + " is running at " + speed + " km/h");  
    }  
  
    public void makeSound() {  
        System.out.println(name + " is making a sound " + sound);  
    }  
}
```

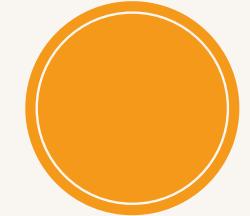
Deklarasi Method



Method memiliki beberapa bagian:

- 1. Nama method** - Nama yang dideklarasikan untuk memanggil method.
- 2. Deretan parameter** - Parameter adalah nilai yang dilewatkan pada method untuk diproses di dalam method.
- 3. Badan method** - Block yang berisi berbagai pernyataan pemrograman.
- 4. Return value** - Nilai output hasil dari operasi method. Return value bersifat opsional, method tanpa return value pada dasarnya memiliki return value dengan tipe data void.

Mengakses Method



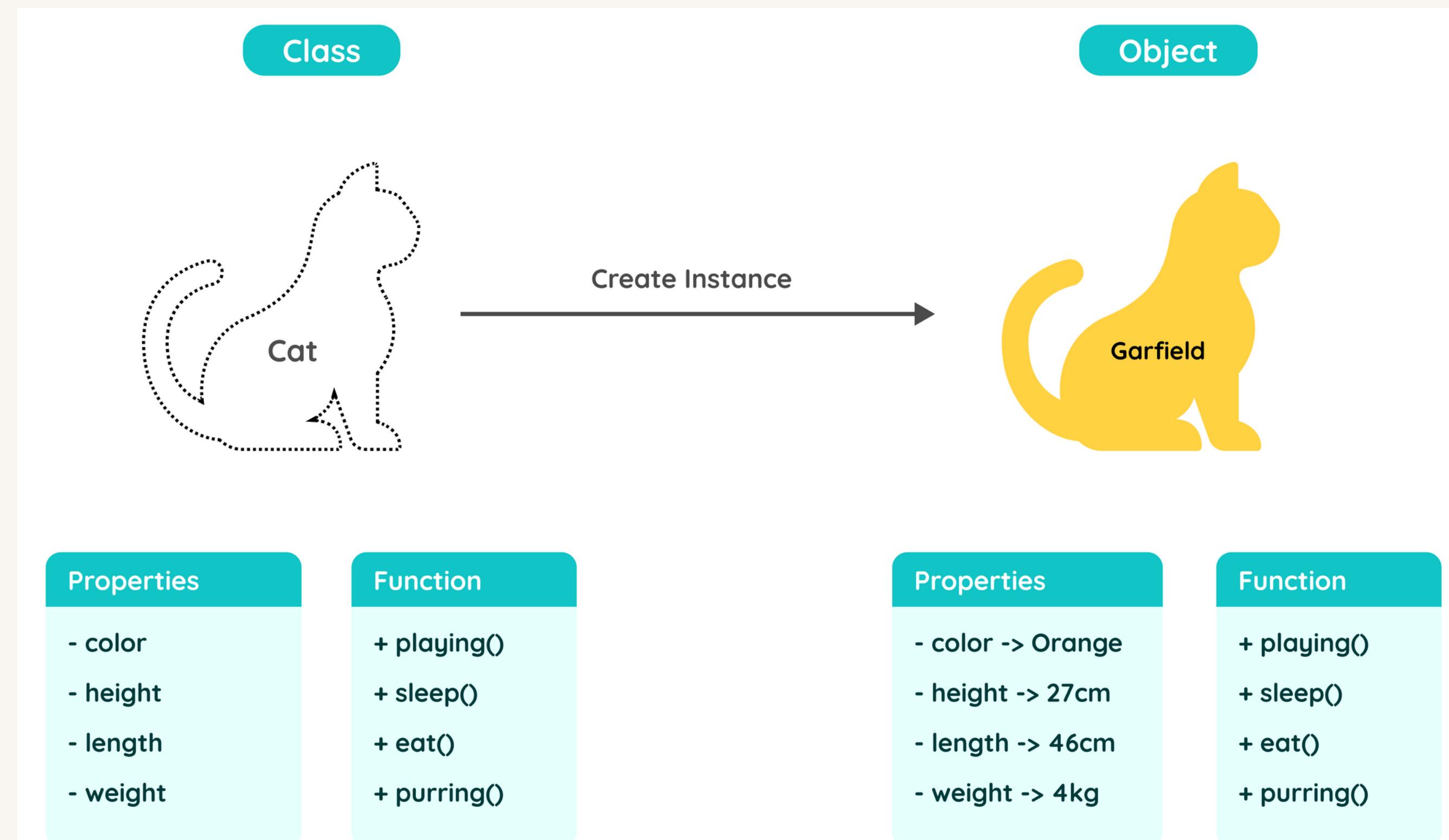
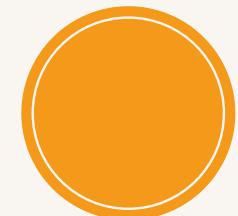
Pemanggilan method dilakukan dengan **nama obyek** diikuti tanda **titik atau dot (.)** kemudian dipanggil methodnya. Contoh: lion.run()

```
public class Animal {
    String name;
    int speed;
    String sound;

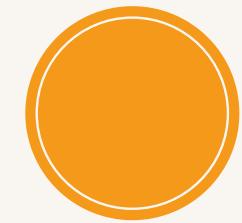
    public static void main(String[] args) {
        Animal lion = new Animal();
        lion.name = "Lion";
        lion.speed = 90;
        lion.sound = "Rrrrr...";
        System.out.println("Animal: " + lion.name + ", speed: " + lion.speed + "km/h, sound: " +
lion.sound);

        Animal cat = new Animal();
        cat.name = "Cat";
        cat.speed = 50;
        cat.sound = "Meaw...";
        System.out.println("Animal: " + cat.name + ", speed: " + cat.speed + "km/h, sound: " +
cat.sound);
    }
}
```

Contoh Visualisasi OOP



Penjelasan Visualisasi OOP

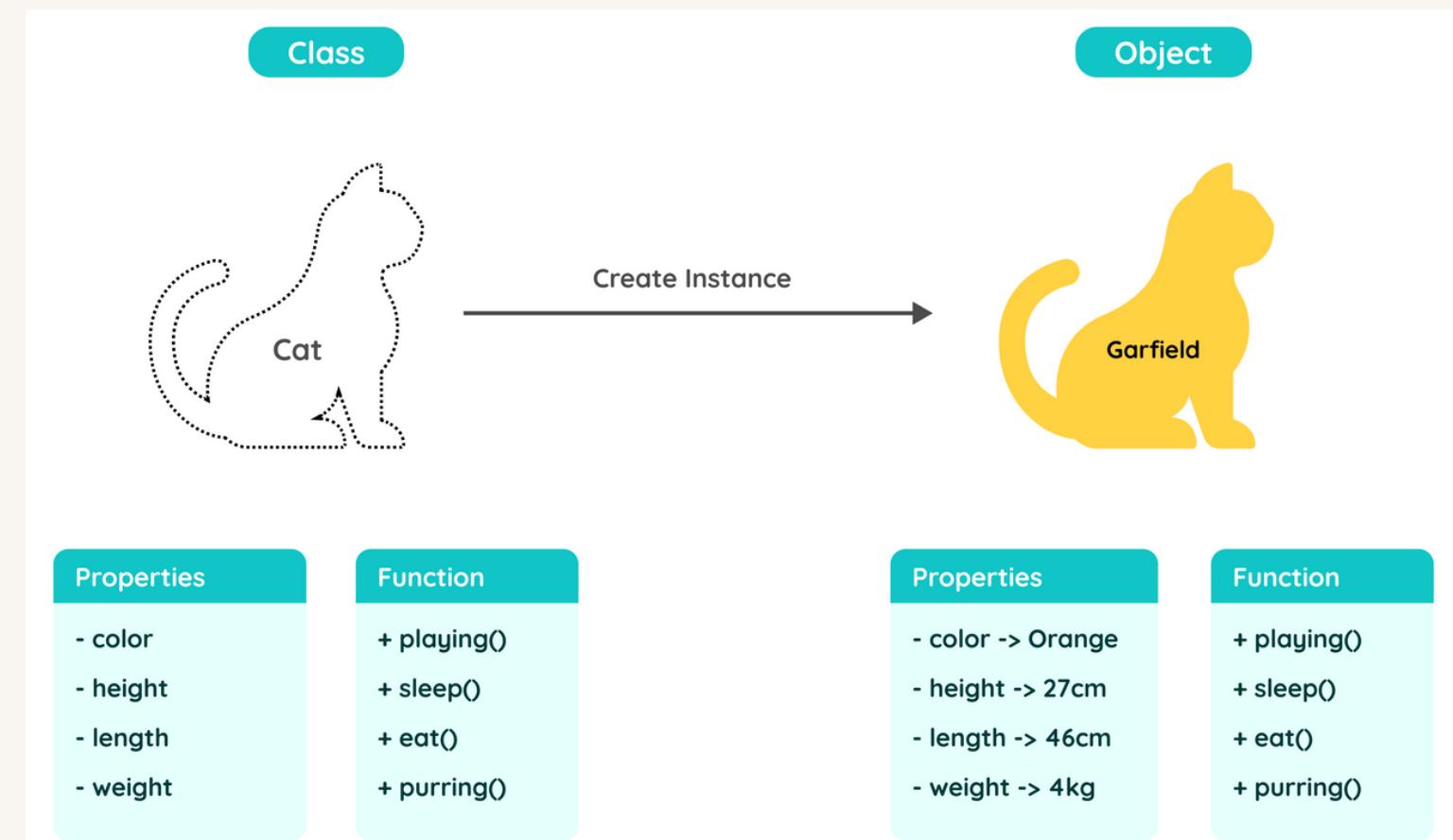


Class merupakan sebuah **blueprint**. **Class** juga bisa disebut sebagai **kumpulan dari objek**.

Objek merupakan **hasil realisasi** dari sebuah **blueprint**

Atribut adalah data yang terdapat dalam sebuah **class** yang mempresentasikan **karakteristik** dari **class** tersebut.

Method merupakan sebuah **prosedur** yang memiliki keterkaitan dengan **pesan dan objek**.



Praktek Membuat Class Kendaraan



1. Buat Class dengan nama **Kendaraan** di package **oop.latihandasar**
2. Buatlah Atribut dengan nama - Tipe Data
 - a. **model** - String
 - b. **jumlahRoda** - int
 - c. **harga** - long
3. Buatlah Method dengan nama - fungsi
 - a. **getInfo()** - mencetak "Kendaraan {model} memiliki jumlah roda {jumlahRoda} dengan harga {harga}
 - b. **tawarHarga(long tawaran)** - mengurangi harga kendaraan sebanyak {tawaran}, lalu mengembalikan value harga
4. Buatlah Class **KendaraanDasarMain** lalu buatlah obyek berdasarkan kelas **Kendaraan**, lalu:
 - a. **Akses** dan **isi** attribut-attribut di obyek
 - b. **Akses** dan **cetak** isi attribut di obyek
 - c. **Panggil** method **getInfo()**
 - d. **Panggil** method **tawarHarga()**, simpan ke dalam variabel **long hasilTawar**, lalu cetak "Tawaran diterima, harga sekarang {hasilTawar}.



Method Main

Method main (main method) adalah method spesial dimana method ini adalah method yang **pertama kali dipanggil** ketika program Java dijalankan.

```
public class Codernity {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

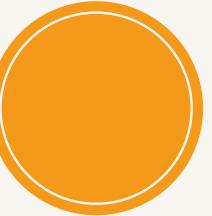
Konstruktor



Konstruktor atau **constructor** adalah method spesial yang berfungsi untuk inisialisasi ketika pembuatan obyek.

```
public class Animal {  
    String name;  
  
    //The constructor of Animal  
    public Animal(String animalName) {  
        name = animalName;  
    }  
}
```

Deklarasi Konstuktor



Konstuktor memiliki beberapa bagian:

1. **Nama konstruktor** - Nama konstruktor sama dengan nama kelas.
2. **Deretan parameter** - Nilai yang dilewatkan pada konstruktor untuk menginisialisasi obyek.
3. **Badan konstuktor** - Block yang berisi berbagai pernyataan pemrograman untuk menginisialisasi obyek.
4. Konstruktor **tidak** memiliki return value

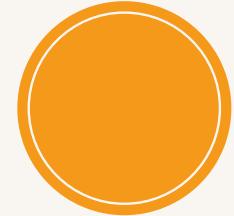
Mengakses Konstruktor



Constructor dipanggil saat membuat obyek baru.

```
public class Animal {  
    String name;  
    int speed;  
  
    //The constructor of Animal  
    public Animal(String animalName, int animalSpeed) {  
        name = animalName;  
        speed = animalSpeed;  
    }  
  
    public static void main(String[] args) {  
        Animal lion = new Animal("Lion", 90);  
    }  
}
```

Latihan Konstruktor



1. Salin Class **Kendaraan** ke package **oop.latihankonstruktor**
2. Pada Class **Kendaraan**, Buatlah Konstruktor dengan deretan parameter:
 - a. **modelParam** - **String**,
 - b. **jumlahRodaParam** - **int** dan
 - c. **hargaParam** - **long**.
3. Inisiakan deretan parameter tersebut kedalam atribut Class **Kendaraan**.
4. Buatlah Class **KendaraanKonstruktorMain** lalu buatlah obyek berdasarkan kelas Kendaraan, lalu:
 - a. **Inisiasi** dengan Konstruktor yang sudah dibuat
 - b. **Akses** dan **cetak** isi attribut di obyek
 - c. **Panggil** method **getInfo()**
 - d. **Panggil** method **tawarHarga()**, simpan ke dalam variabel long **hasilTawar**, lalu cetak "Tawaran diterima, harga sekarang {hasilTawar}.



Atribut Statis

Atribut statis adalah atribut yang dilengkapi dengan modifier **static**.

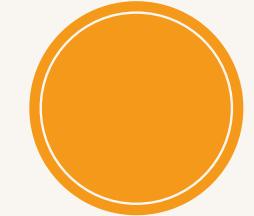
Atribut statis menyatakan bahwa atribut yang telah dideklarasikan merupakan **variabel kelas**, bukan variabel obyek.

```
public class Codeternity {
    public static void main(String args[]) {
        int carNumberOfTyre = Car.numberOfTyre; //no need to instantiate to access static variable

        System.out.println(carNumberOfTyre);
    }
}

class Car {
    static int numberOfTyre = 4;
}
```

Method Statis

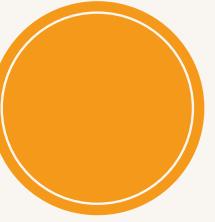


Method statis (static method) memiliki ciri dan sifat yang mirip dengan atribut statis dimana method yang dideklarasikan dengan modifier static menjadikan method tersebut sebagai **method milik kelas**, bukan milik obyek.

```
public class Codernity {
    public static void main(String args[]) {
        int result = Math.add(5, 6);
        System.out.println(result); //output 50
    }
}

class Math {
    public static int add(int value1, int value2) {
        return value1 + value2;
    }
}
```

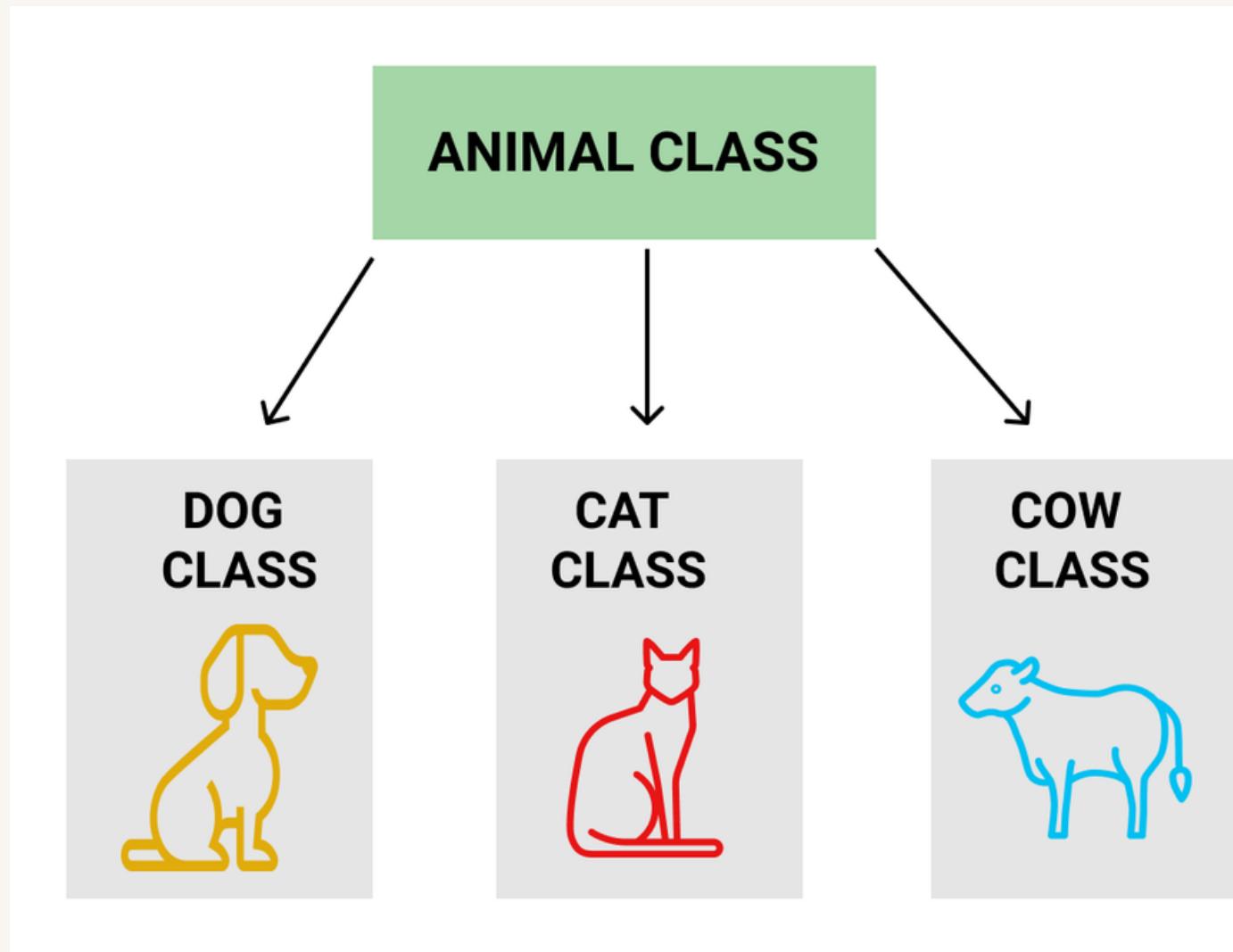
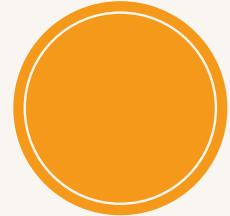
Empat Pillar dalam OOP



- 1. Inheritance**
- 2. Encapsulation**
- 3. Polymorphism**
- 4. Abstraction (Abstract Class & Interface)**



Inheritance

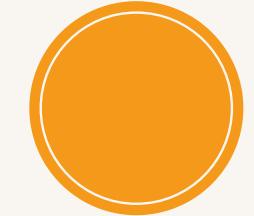


Ketika satu objek memperoleh semua properti dan perilaku dari objek **parent/induk**, itu dikenal sebagai **Inheritance**.

Dalam penerapan inheritance, kita akan sering mendengar beberapa istilah berikut:

1. **SuperClass**
2. **SubClass**

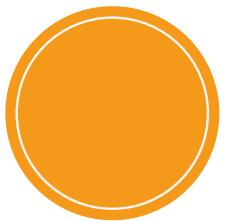
Contoh Inheritance



Untuk membuat kelas turunan dari kelas induk digunakan keyword **extends**.

```
...  
  
class Vehicle {  
    //superclass (parent)  
}  
  
class Car extends Vehicle {  
    //subclass (child)  
}
```

Implementasi Inheritance



```
package codernity;

public class Codernity {
    public static void main(String args[]) {
        System.out.println("-- Car --");
        Car car = new Car();
        car.turnLeft();
        car.openSunroof();

        System.out.println("-- Airplane --");
        Airplane airplane = new Airplane();
        airplane.turnLeft();
        airplane.fly();
    }
}

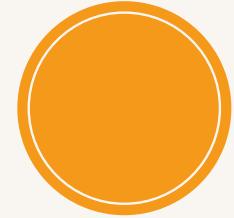
//Parent class
class Vehicle {
    public void turnLeft() {
        System.out.println("Turn left");
    }

    public void turnRight() {
        System.out.println("Turn right");
    }
}

//Child class 1
class Car extends Vehicle {
    public void openSunroof() {
        System.out.println("Open sunroof");
    }
}

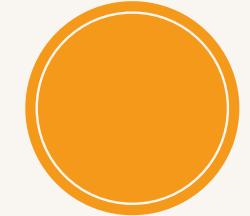
//Child class 2
class Airplane extends Vehicle {
    public void fly() {
        System.out.println("Fly");
    }
}
```

Latihan Pewarisan



1. Salin Class **Kendaraan** ke package **oop.latihanpewarisan**
2. Buatlah Class **Mobil**, lalu jadikan Class **Mobil SubClass** dari Class **Kendaraan**. setelah itu buatlah nama method - fungsi sebagai berikut:
 - a. **bukaPintu()** - Cetak "Pintu Terbuka buat BosQ"
3. Buatlah Class **Motor**, lalu jadikan Class **Motor SubClass** dari Class Kendaraan. setelah itu buatlah nama method - fungsi sebagai berikut:
 - a. **standarGanda()** - Cetak "Motor Sudah berstandar Ganda"
4. Buatlah method di Class **Kendaraan**, dengan nama method - fungsi sebagai berikut:
 - a. **gasPol()** = Cetak "Gas Puoooll~!!"
5. Buatlah Class **KendaraanPewarisanMain** lalu buatlah obyek berdasarkan kelas **Mobil** dan **Motor**, pada setiap objek panggilah fungsi berikut:
 - a. **Mobil** -> Panggil Method **bukaPintu()** lalu **gasPol()**
 - b. **Motor** -> Panggil Method **standarGanda()** lalu **gasPol()**

Keyword this



Keyword **this** digunakan untuk menunjuk ke obyek aktif / obyek sekarang

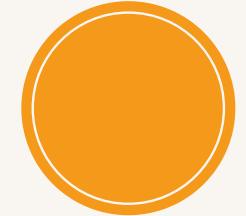
```
public class Codernity {
    public static void main(String args[]) {
        Human human = new Human();
        human.printGreeting();
    }
}

class Human {
    String greeting = "Hello..";

    void printGreeting() {
        String greeting = "Holla..";

        System.out.println(this.greeting);
        System.out.println(greeting);
    }
}
```

Keyword super



Keyword **super** digunakan untuk **menunjuk obyek superclass** dari obyek aktif.

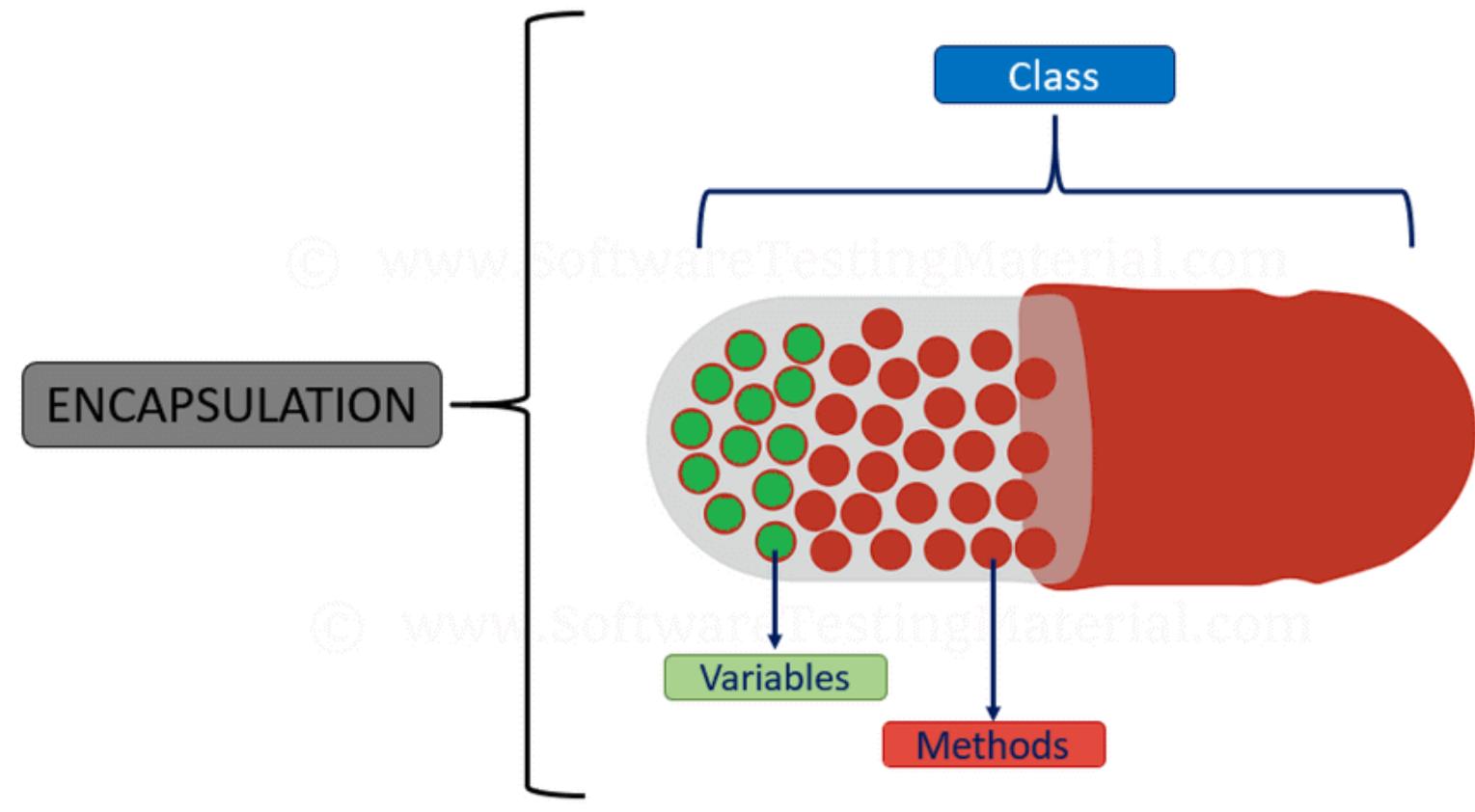
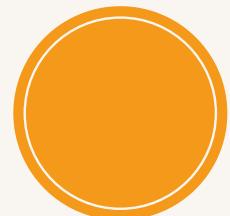
```
...
public class Codernity {
    public static void main(String[] args) {
        Car car = new Car();
        car.allInfo();
    }
}

class Vehicle {
    public void info() {
        System.out.println("This is a vehicle");
    }
}

class Car extends Vehicle {
    public void info() {
        System.out.println("This is a car");
    }

    public void allInfo() {
        super.info(); // This is a vehicle
        this.info(); // This is a car
    }
}
```

Encapsulation



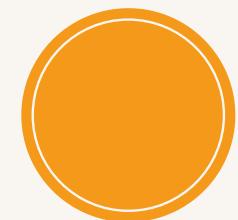
Enkapsulasi (encapsulation) adalah pembungkusan data atau penyembunyian data-data privat dari suatu obyek sehingga tidak dapat diakses dari obyek lain.

Untuk mengatur hak akses dari masing-masing method dan atribut digunakan modifier akses.

Method Setter & Getter

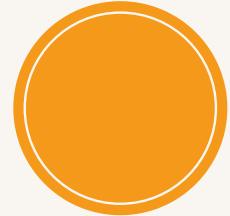
```
class Vehicle {  
    private int speed;  
  
    //setter method to set the value of speed  
    public void setSpeed(int speed) {  
        this.speed = speed;  
    }  
  
    //getter method to set the value of speed  
    public int getSpeed() {  
        return speed;  
    }  
}
```

Latihan Enkapsulasi



1. Salin Class **Kendaraan**, **Mobil** & **Motor** ke package **oop.latihanenkapsulasi**
2. Enkapsulasi attribut-attribut dari class **Kendaraan**, lalu buatlah method getter setternya untuk setiap attribut di class **Kendaraan**.
3. Buatlah Class **KendaraanEnkapsulasiMain** lalu buatlah obyek berdasarkan kelas **Kendaraan**, pada objek tersebut buatlah:
 - a. Isikan attribut-attribut kendaraan dengan **method setter**
 - b. Cetak Isi dari atribut-atribut kendaraan dengan **method getter**

Polymorphism



Polymorphism merupakan kemampuan **objek**, **variabel**, atau **fungsi** yang dapat memiliki berbagai bentuk.

1. Compile Time Polymorphism (**Overloading**)
 - a. Overloading -> Fungsi dengan nama yang sama tetapi memiliki parameter yang berbeda.
2. Runtime Polymorphism (**Overriding**)
 - a. Overriding -> Sebuah Class mempunya fungsi dengan nama yang sama yang ada di kelas induknya.

Overloading



Overloading adalah deklarasi method maupun konstruktor lebih dari satu kali pada kelas yang sama.

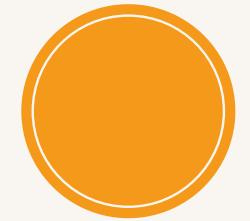
```
public class Codernity {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle();
        vehicle.run();
        vehicle.run(100);
        vehicle.run("fast");
    }
}

class Vehicle {
    public void run() {
        System.out.println("Vehicle is running");
    }

    public void run(int speed) {
        System.out.println("Vehicle is running at " + speed + " km/h");
    }

    public void run (String speed) {
        System.out.println("Vehicle is running " + speed);
    }
}
```

Overriding



Method **overriding** adalah pendeklarasian ulang sebuah method pada subclassnya.

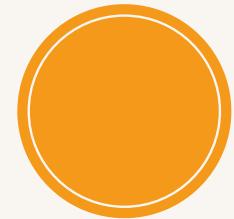
```
public class Codernity {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle();
        Car car = new Car();

        vehicle.info();
        car.info();
    }
}

class Vehicle {
    public void info() {
        System.out.println("This is a vehicle");
    }
}

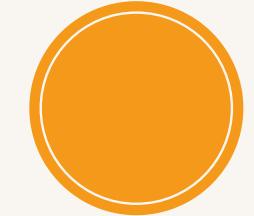
class Car extends Vehicle {
    public void info() {
        System.out.println("This is a car");
    }
}
```

Latihan Polymorfisme



1. Salin Class **Kendaraan**, **Mobil** & **Motor** ke package **oop.latihanpolymorfisme**
2. Buatlah method Overloading Method di kelas **Mobil**, yaitu:
 - a. **isiBensin()** - Cetak "Isi Bensin sebanyak-banyaknya ahh~"
 - b. **isiBensin(int liter)** - "Isi Bensin sebanyak {liter} Liter"
3. Buatlah method Overriding Method di kelas **Motor**, yaitu:
 - a. **gasPol()** - cetak "Gas Pol Pake Motor!!!"
4. Buatlah Class **KendaraanPolymorfismeMain** lalu buatlah obyek berdasarkan kelas **Mobil** dan **Motor**, pada setiap objek panggilah fungsi berikut:
 - a. Mobil -> Panggil Method **isiBensin()** dan **isiBensin(int liter)**
 - b. Motor -> Panggil Method **gasPol()** dan **super.gasPol()**

Kelas Abstract



Kelas abstrak (abstract class) adalah kelas yang dideklarasikan menggunakan modifier abstract yang mana kelas abstrak tidak dapat dibuat obyek namun hanya dapat diturunkan.

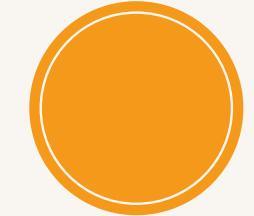
```
public class Codernity {
    public static void main(String args[]) {
        //Vehicle vehicle = new Vehicle(); not legal because Vehicle is an abstract class
        Car car = new Car(); //Legal
        Train train = new Train(); //Legal
    }
}

abstract class Vehicle {
    //Methods & attributes of Vehicle
}

class Car extends Vehicle {
    //Methods & attributes of Car
}

class Train extends Vehicle {
    //Methods & attributes of Train
}
```

Method Abstrak



Method abstrak (abstract method) adalah method yang dideklarasikan kosong tanpa adanya tubuh method dan statement sama sekali.

```
public class Codernity {
    public static void main(String args[]) {
        Car car = new Car();
        Train train = new Train();

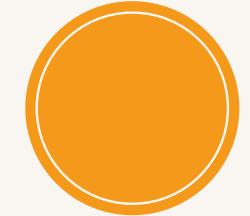
        car.horn();
        train.horn();
    }
}

abstract class Vehicle {
    abstract void horn();
}

class Car extends Vehicle {
    void horn() {
        System.out.println("Beep beep...");
    }
}
class Train extends Vehicle {
    void horn() {
        System.out.println("Choo choo...");
    }
}

//CombatVehicle is declared as abstract because it doesn't override horn
abstract class CombatVehicle extends Vehicle { }
```

Interface



Interface adalah bentuk mirip kelas dimana didalamnya terdiri dari kumpulan method kosong dan konstanta. Interface tidak dapat dibuat obyek namun hanya dapat diimplementasi.

```
...
public class Codernity {
    public static void main(String args[]) {
        Car car = new Car();
        car.accelerate();
        car.turnLeft();
        car.turnRight();
    }
}

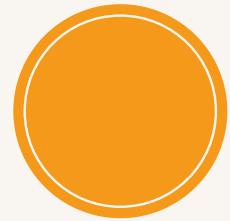
interface Drivable {
    void accelerate();
    void turnLeft();
    void turnRight();
}

class Car implements Drivable {
    @Override
    public void accelerate() {
        System.out.println("Car is accelerate");
    }

    @Override
    public void turnLeft() {
        System.out.println("Car is turn left");
    }

    @Override
    public void turnRight() {
        System.out.println("Car is turn right");
    }
}
```

Multiple Inheritance

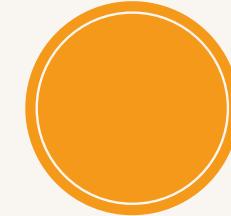


Multiple inheritance adalah pewarisan sifat terhadap dua atau lebih kelas.

Pada dasarnya Java **tidak mengizinkan** sebuah kelas melakukan extends terhadap dua kelas sekaligus.

Multiple inheritance pada Java dilakukan dengan **meng-extends sebuah kelas** ditambah **implementasi** satu atau lebih **interface**.

Praktek Pada Class Kendaraan

- 
1. Salin Class **Kendaraan**, **Mobil** & **Motor** ke package **oop.latihanabstraksi**
 2. Ubahlah Class **Kendaraan** menjadi **abstract Class**, lalu ubahlah method **gasPol()** menjadi method **abstract**
 3. Buatlah **Interface** dengan nama **Navigasi** dengan isi-isi method abstract seperti berikut:
 - a. **maju()**
 - b. **belokKiri()**
 - c. **belokKanan()**
 4. Implementasikan Interface **Navigasi** ke kelas **Mobil** & **Motor** lalu implementasi method dengan mencetak
 - a. **maju()** - Cetak "Maju dengan menggunakan {Mobil/Motor}"
 - b. **belokKiri()** - Cetak "Belok kiri dengan menggunakan {Mobil/Motor}"
 - c. **belokKanan()** - Cetak "Belok kanan dengan menggunakan {Mobil/Motor}"
 5. Buatlah Class **KendaraanAbstraksiMain** lalu buatlah obyek berdasarkan kelas **Mobil** dan **Motor**, pada setiap objek panggilah fungsi berikut:
 - a. Mobil -> Panggil Method **maju()**, **belokKiri()** dan **belokKanan()**
 - b. Motor -> Panggil Method **maju()**, **belokKiri()** dan **belokKanan()**



Object Oriented Programming (OOP)

The End