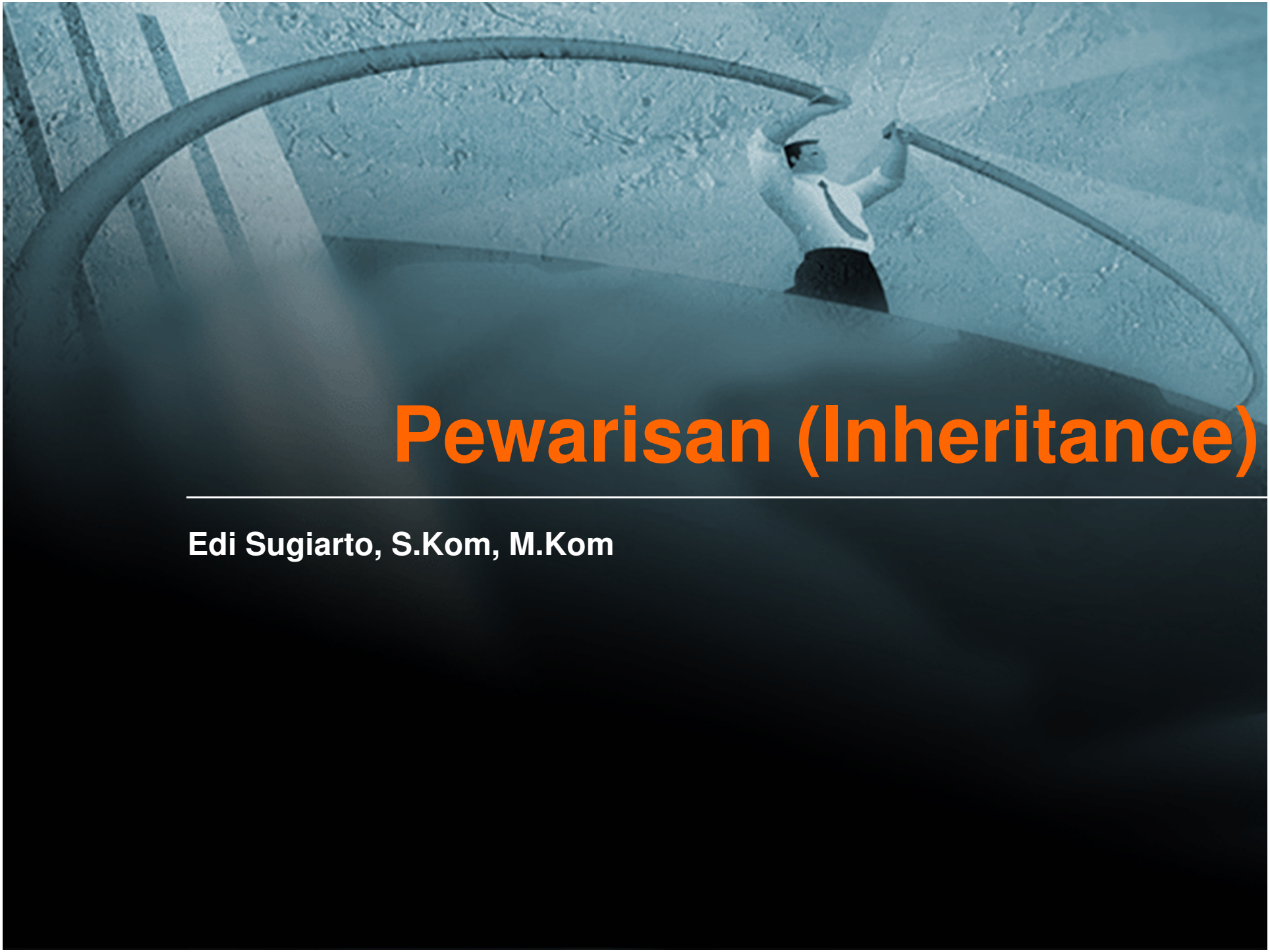


Pewarisan (Inheritance)

Edi Sugiarto, S.Kom, M.Kom

Pendahuluan

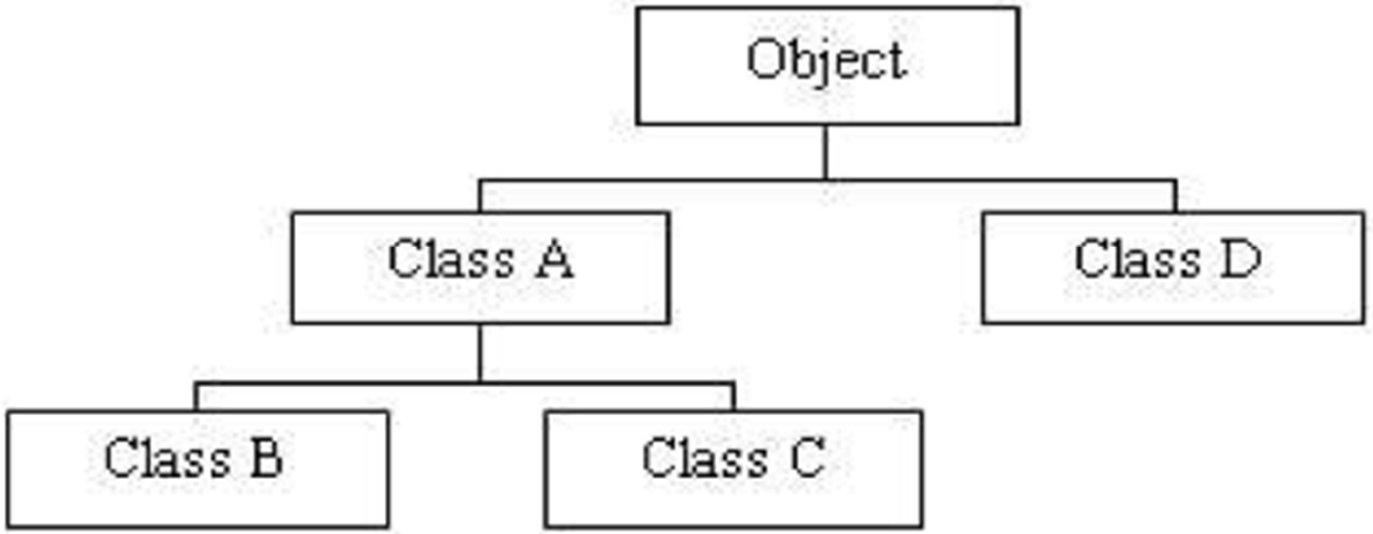
- Hampir selalu perangkat lunak baru memperluas pengembangan dari perangkat lunak yg sebelumnya.
- Cara terbaik untuk menciptakan adalah meniru, memperbaiki, dan mengkombinasikan, pada prinsipnya teknologi objek telah memperhatikan hal tersebut.
- Pewarisan merupakan sarana untuk meniru dan memperbaiki yang dilakukan secara jelas dan bersih, sementara komposisi merupakan sarana untuk mengkombinasikan hasil-hasil sebelumnya.

Inheritance

- Merupakan konsep bahwa suatu kelas dapat mewariskan atribut dan method yang dimilikinya (supperclass) kepada class lain (subclass) serta membentuk kelas yang hirarki.
- *Pewarisan merupakan mekanisme mengekspresikan keserupaan di antara kelas, menyederhanakan pendefinisian kelas yg serupa dengan kelas yg didefinisikan sebelumnya* (Bambang Hariyanto, Ir, 2004).

Inheritance (Lanjutan)

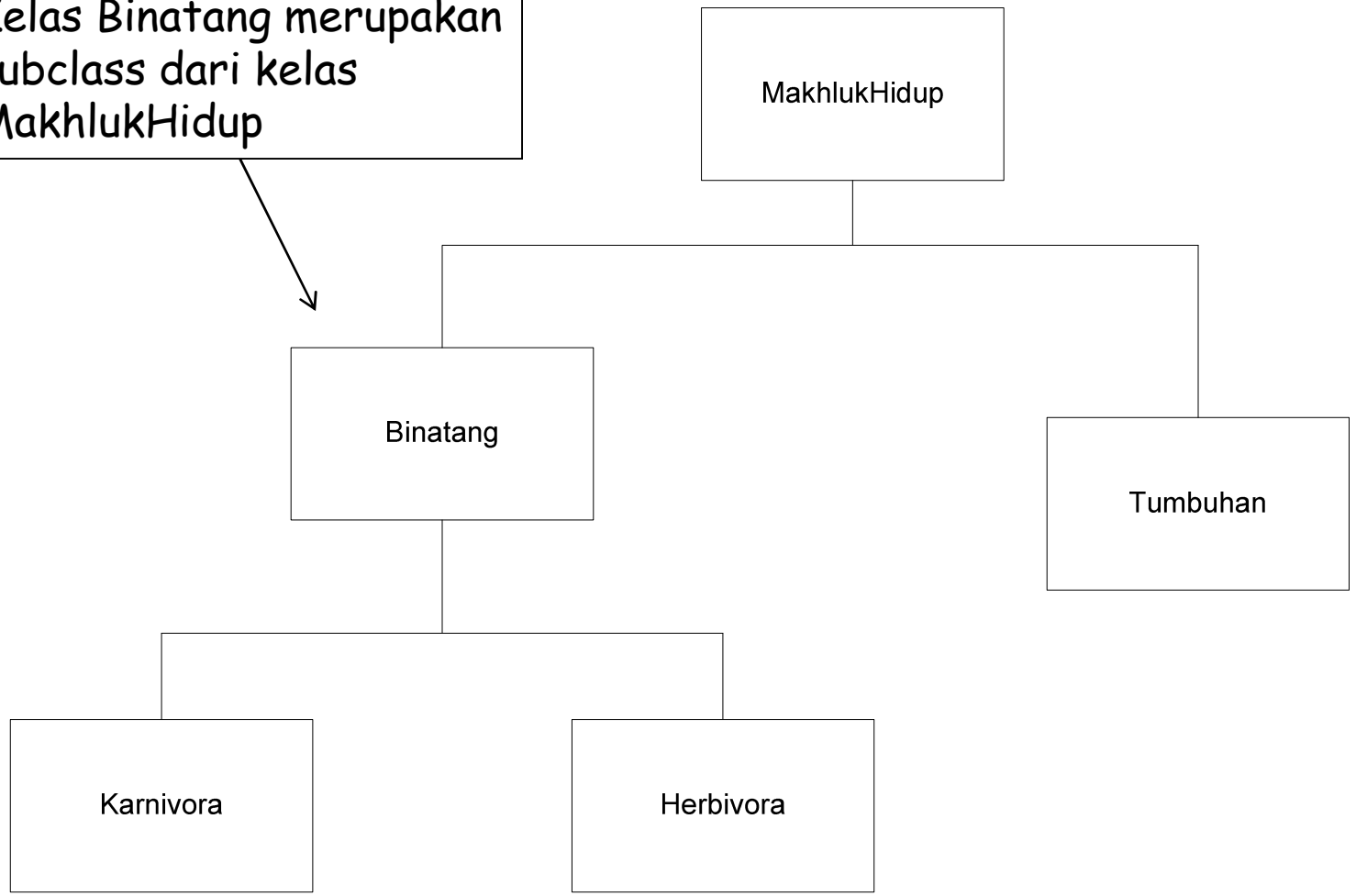
Kelas dalam Java, termasuk kelas yang membangun Java API, merupakan kumpulan subclass dari superclass Object.



Inheritance (Lanjutan)

- **Superkelas (Superclass)**
  - kelas yang letaknya di atas kelas tertentu di dalam hierarki.
  - Superkelas bertindak sebagai kelas yang mewariskan attribut dan method pada kelas dibawahnya (subkelas)
- **Subkelas (Subclass)**
  - kelas yang letaknya di bawah kelas tertentu di dalam hierarki, serta kelas yang mewarisi attribut dan method dari kelas di atasnya (superkelas)

Kelas Binatang merupakan subclass dari kelas MakhlukHidup



edi.sugiarto@yahoo.com

Inheritance (Lanjutan)

- Pewarisan merupakan sarana untuk menghilangkan penulisan ulang terhadap kode yang dapat digunakan berulang kali, dalam hal ini bersumber pada konteks antara superkelas dan subkelas.
- Pewarisan memberi fasilitas pemodelan untuk menstrukturkan kelas-kelas menjadi lebih ringkas, dan menangkap apa yang serupa dan apa yang berbeda antara kelas-kelas

Manfaat Inheritance

- **Ekstensi dan Batasan**
  - Subkelas boleh melakukan penimpaan fitur di superkelas dengan mendefinisikan fitur dengan nama yg sama.
  - Subkelas boleh menambah fitur baru sebagai perluasan, disebut ekstensi, dan subkelas boleh memberikan batasan attribut, disebut pembatasan.
- **Redefinisi operasi**
  - Penimpaan operasi sebagai perluasan
  - Penimpaan operasi sebagai batasan
  - Penimpaan operasi sebagai optimasi
  - Penimpaan operasi sebagai kenyamanan

Keuntungan Inheritance

Reusability

- Ketika behaviour(method) dideklarasikan dalam superclass, behaviour tersebut otomatis diwariskan ke seluruh subclass
- Jadi, Anda dapat meng-enkode method hanya sekali dan method tersebut dapat digunakan oleh seluruh subclass
- Sebuah subclass hanya perlu mengimplementasikan perbedaan antara dirinya sendiri dan parent-nya

Inheritance (Lanjutan)

- Untuk menjadikan sebuah kelas menjadi subclass dari objek lain dalam *java* maka dapat menggunakan keyword extends.

- Perhatikan contoh berikut :

```
public class kelasAnak extends kelasInduk
{

}
```

Keyword “super”

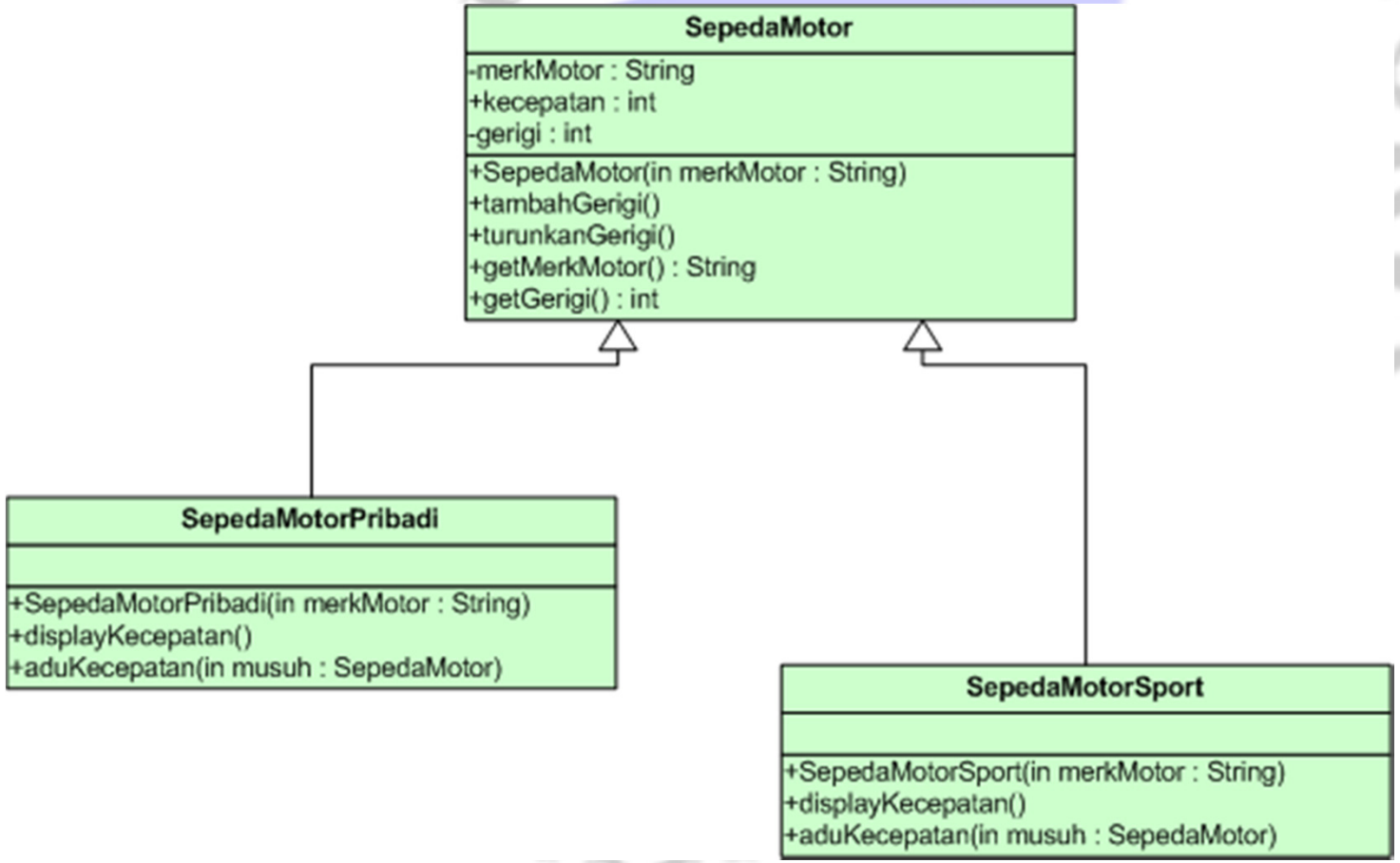
- Sebuah subclass dapat juga memanggil secara *eksplisit* sebuah constructor superclass yang ada diatasnya.
- Dapat dilakukan dengan menggunakan pemanggilan keyword ***super***.
- Pemanggilan keyword *super* didalam konstruktor subclass akan mengakibatkan eksekusi dalam konstruktor yang relevan dari superclass, berdasarkan *passing argument*-nya.

Keyword “super”

Beberapa hal untuk diingat ketika menggunakan pemanggilan constructor super :

- Pemanggilan *super()* HARUS DIJALANKAN SESUAI DENGAN STATEMENT PERTAMA DALAM SEBUAH KONSTRUKTOR.
- Pemanggilan *super()* hanya dapat digunakan di dalam definisi konstruktor.
- Hal ini menjelaskan bahwa konstruktor this() dan pemanggilan *super()* TIDAK DAPAT DIJALANKAN SECARA BERSAMAAN DI DALAM CONSTRUCTOR YANG SAMA.

Contoh Inheritance



```java
public class SepedaMotor {

    protected int kecepatan=10;
    private int gerigi=0;
    private String merkMotor="";

    public SepedaMotor(String merkMotor, int kecepatan)
    {
        this.merkMotor=merkMotor;
        this.kecepatan=kecepatan;
    }

    public void tambahGerigi()
    {
        gerigi++;
        System.out.println("Motor "+this.merkMotor+" Pindah ke Gerigi "+gerigi);
    }

    public void turunkanGerigi()
    {
        gerigi--;
        System.out.println("Motor "+this.merkMotor+" Pindah ke Gerigi "+gerigi);
    }

    public String getMerkMotor()
    {
        return this.merkMotor;
    }

    public int getGerigi()
    {
       return this.gerigi;
    }
}
```

```
public class SepedaMotorPribadi extends SepedaMotor {

    public SepedaMotorPribadi(String merkMotor)
    {
        super(merkMotor, 20);
    }

    public void aduKecepatan(SepedaMotor motorLawan)
    {
        if((this.kecepatan*this.getGerigi())>(motorLawan.kecepatan*motorLawan.getGerigi()))
        {
            System.out.println(this.getMerkMotor()+" Menang dengan kecepatan "
                    +(this.kecepatan*this.getGerigi())+" Km/Jam");
        }
        else if((this.kecepatan*this.getGerigi())<(motorLawan.kecepatan*motorLawan.getGerigi()))
        {
            System.out.println(motorLawan.getMerkMotor()+ " Menang dengan kecepatan "
                    +(motorLawan.kecepatan*motorLawan.getGerigi())+" Km/Jam");
        }
        else
        {
            System.out.println("Kecepatan "+this.getMerkMotor()+" dengan "+
                    motorLawan.getMerkMotor()+" Sama ");
        }

    }
    public void displayKecepatan()
    {
        System.out.println("Kecepatan "+getMerkMotor()+" " +getGerigi()*kecepatan+" Km/Jam");
        System.out.println("--------------------------------");
    }

}
```

```
public class SepedaMotorSport extends SepedaMotor {

    public SepedaMotorSport(String merkMotor)
    {
        super(merkMotor, 50);
    }

    public void aduKecepatan(SepedaMotor motorLawan)
    {
        if((this.kecepatan*this.getGerigi())>(motorLawan.kecepatan*motorLawan.getGerigi()))
        {
            System.out.println(this.getMerkMotor()+" Menang dengan kecepatan "
                    +(this.kecepatan*this.getGerigi())+" Km/Jam");
        }
        else if((this.kecepatan*this.getGerigi())<(motorLawan.kecepatan*motorLawan.getGerigi()))
        {
            System.out.println(motorLawan.getMerkMotor()+ " Menang dengan kecepatan "
                    +(motorLawan.kecepatan*motorLawan.getGerigi())+" Km/Jam");
        }
        else
        {
            System.out.println("Kecepatan "+this.getMerkMotor()+" dengan "+
                    motorLawan.getMerkMotor()+" Sama ");
        }

    }

    public void displayKecepatan()
    {
        System.out.println("Kecepatan "+getMerkMotor()+" "+getGerigi()*kecepatan+" Km/Jam");
        System.out.println("--------------------------------");
    }

}
```

```
public class SepedaMotor_Main {

    public static void main(String[] args)
    {
        SepedaMotorSport kawasaki = new SepedaMotorSport("Kawasaki");
        kawasaki.tambahGerigi();
        kawasaki.tambahGerigi();

        SepedaMotorPribadi mio = new SepedaMotorPribadi("Yamaha Mio");
        mio.tambahGerigi();
        mio.tambahGerigi();
        mio.tambahGerigi();
        mio.tambahGerigi();
        mio.tambahGerigi();

        kawasaki.aduKecepatan(mio);

        kawasaki.tambahGerigi();

        mio.aduKecepatan(kawasaki);

    }

}
```

Prinsip Inheritance

- Atribut maupun method pada kelas diatasnya (superclass), dapat diakses oleh kelas dibawahnya (subclass). Namun tidak sebaliknya.
- Menggunakan keyword **extends** untuk mengubah suatu class menjadi subclass dari objek lain
- Menggunakan keyword **super()** untuk memanggil secara eksplisit konstruktor kelas diatasnya.

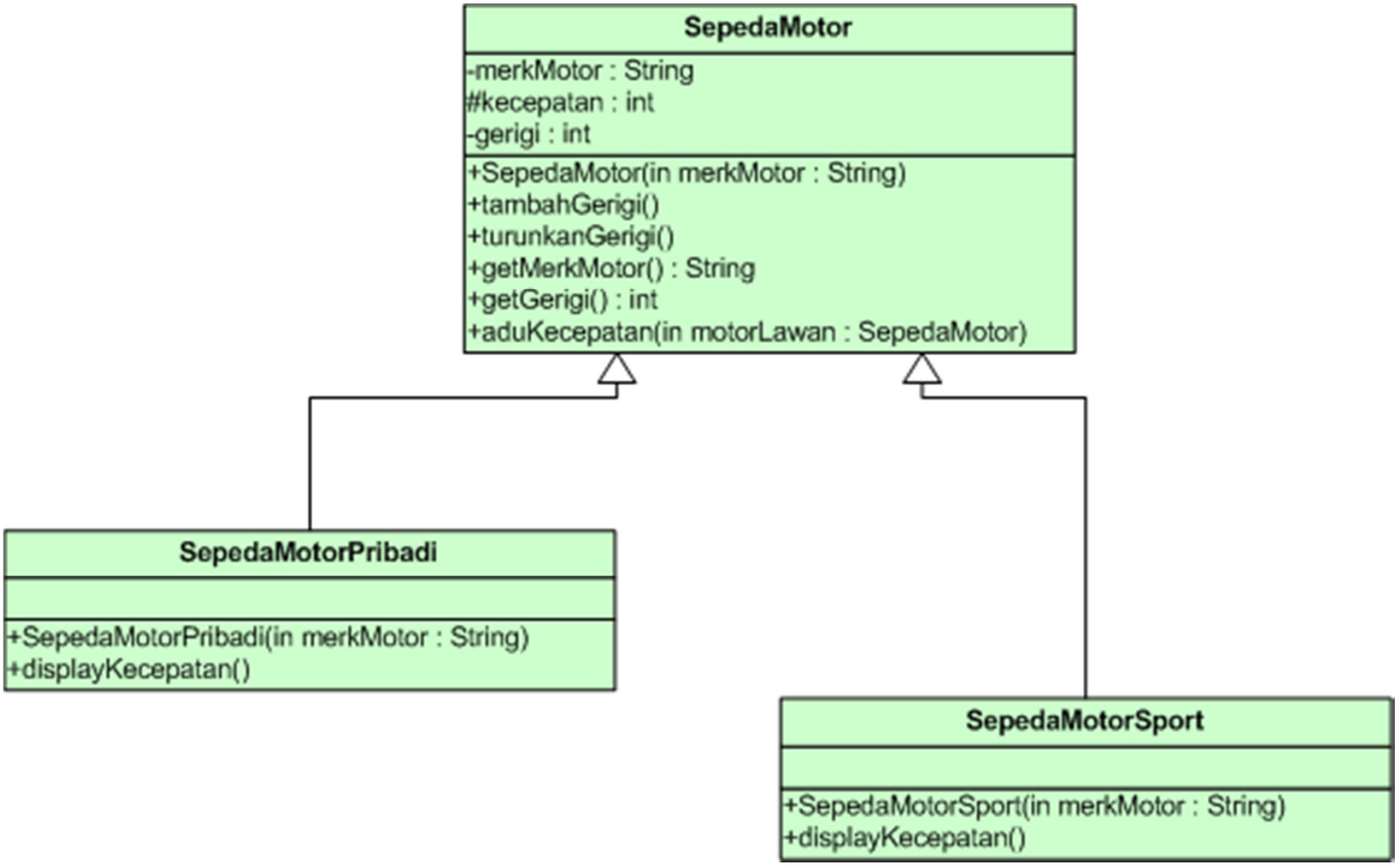
What???

Ada Pertanyaan ?

Why ???

edi.sugiarto@yahoo.com

Latihan 6.3


SepedaMotor
-merkMotor : String
#kecepatan : int
-gerigi : int
+SepedaMotor(in merkMotor : String)
+tambahGerigi()
+turunkanGerigi()
+getMerkMotor() : String
+getGerigi() : int
+aduKecepatan(in motorLawan : SepedaMotor)
SepedaMotorPribadi
+SepedaMotorPribadi(in merkMotor : String)
+displayKecepatan()
SepedaMotorSport
+SepedaMotorSport(in merkMotor : String)
+displayKecepatan()

Latihan 6.4

Implementasikan class diagram disamping kedalam bahasa pemrograman.

Selanjutnya
Representasikan ke dunia nyata bahwa binatang karnivora hanya memakan makhluk dari kelas binatang, sedangkan herbivora hanya memakan makhluk dari kelas tumbuhan

- Selanjutnya implementasikan ke dalam main program sbb:

```
public class MakhlukHidup_Main {

    public static void main(String[] args)
    {
        Tumbuhan rumput = new Tumbuhan("Rumput Teki");
        Tumbuhan beringan = new Tumbuhan("Pohon Beringin");

        Herbivora kambing = new Herbivora("Kambing");
        Karnivora harimau = new Karnivora("Harimau");

        kambing.Memakan(rumput);
        harimau.Memakan(kambing);

        harimau.profile();

        harimau.berTumbuh();
        harimau.berTumbuh();
        harimau.profile();
    }

}
```