Fakultas Ilmu Terapan
School of Applied Science
Diploma of Informatics Management

PEWARISAN

Disusun Oleh:
Reza Budiawan

Untuk:
Tim Dosen Algoritma & Pemrograman Lanjut

Inheritance/Pewarisan

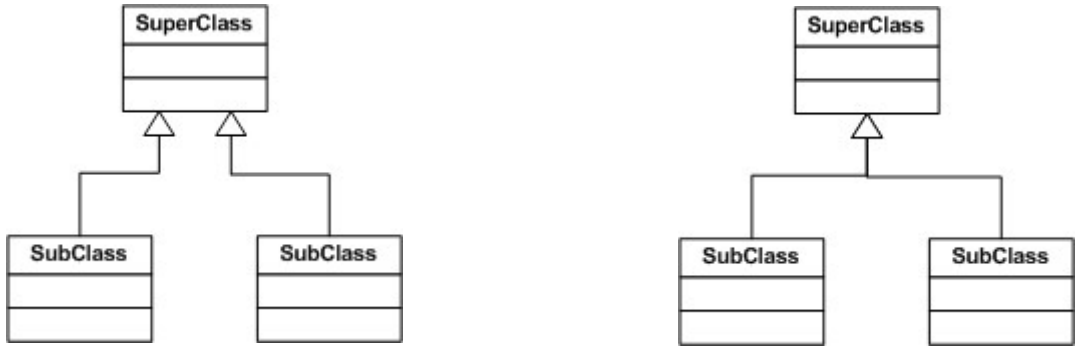
- Konsep pewarisan merupakan proses penciptaan class baru dengan mewarisi karakteristik class yang telah ada/dibuat, juga ditambah karakteristik unik dari class baru tersebut.
- Konsep ini memungkinkan class baru mewarisi fungsionalitas class yang sudah ada.
- Untuk menciptakan class baru, kita hanya perlu menspesifikasikan cara class baru itu berbeda dari class yang sudah ada.

Inheritance/Pewarisan

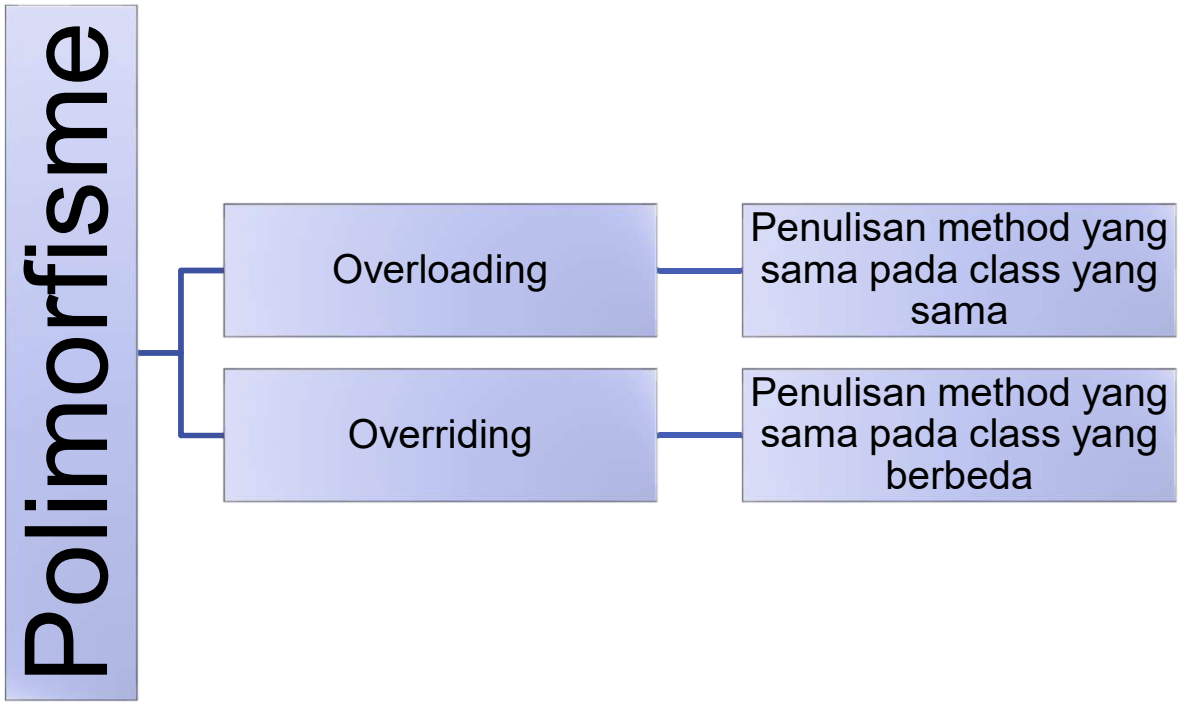
- Class yang sudah ada kita sebut dengan class induk/super class, dan class yang mewarisi class induk kita sebut dengan class turunan/sub class.
- Java hanya memungkinkan pewarisan tunggal (single inheritance) pada class, dan bisa pewarisan majemuk/jamak/multiple dengan menerapkan interface.
- Keyword yang dipakai pada konsep pewarisan, yakni: extends, super, override, dan implements.

Inheritance/Pewarisan

- Simbol: Panah segitiga di akhir class
- Contoh Diagram:



Polimorfisme



Polimorfisme

Apa tujuan Polimorfisme?

Overloading

- Biasa terjadi pada konstruktor
- Syarat:
  - Nama method sama,
  - parameternya berbeda baik dari segi tipe data yang dipakai, jumlah parameternya, serta susunan parameternya.

Contoh Overloading

```
public class Mahasiswa{
    private String nama, nim;
    public mahasiswa(String nama){
        this.nama = nama;
    }
 public mahasiswa(String nama, String nim){
        this.nama = nama;
        this.nim = nim;
    }
}
```

Overloading karna?

Contoh Overloading

```
public class Mahasiswa{
    private String nama, nim;
    public mahasiswa(String nim, String nama){
        this.nim = nim;
        this.nama = nama;
    }
 public mahasiswa(String nama, String nim){
        this.nama = nama;
        this.nim = nim;
    }
}
```

BUKAN
Overloading karna?

Overriding

- Jenis polimorfisme ini bisa diterapkan pada class turunannya.
- Overriding method terkait pada konsep pemrograman berorientasi object Pewarisan (inheritance).
- Maksudnya, method di class induk bisa diperluas di class turunannya.
- Class turunannya juga bisa mengubah isi method yakni operasi atau aksi yang dilakukan di class induk

Overriding

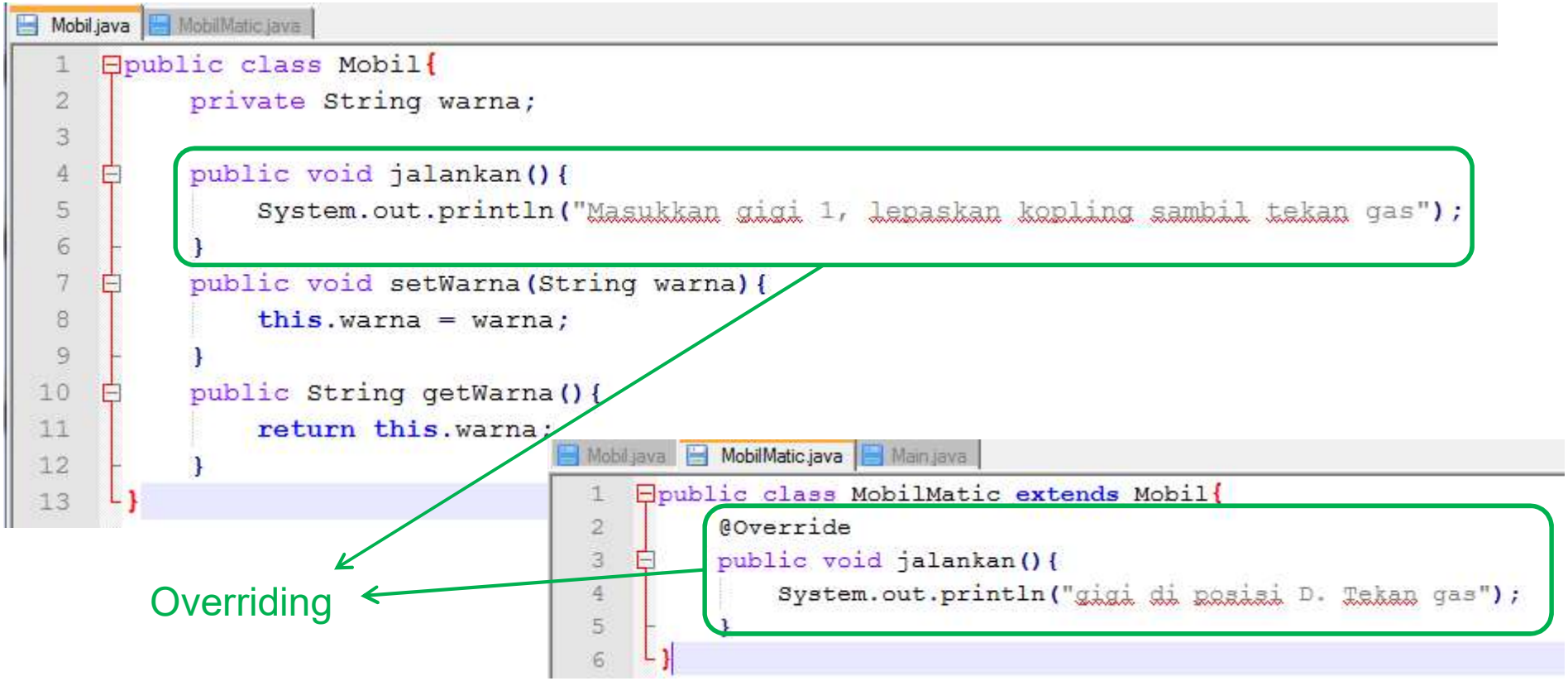
- Karakteristik dari overriding method:
  - Method di class induk yang bisa di-overriding yakni semua method dengan modifier default, public, protected.
  - Method constructor di class induk tidak pernah bisa di-overriding.
  - Jenis tipe data, jumlah, dan susunan parameter di method class induk tidak berubah.

Keyword Overriding

extends

@Override

Contoh Overriding



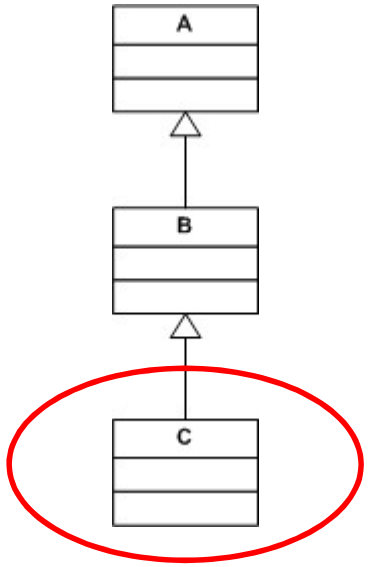
Overriding

Mobil.java | MobilMatic.java | Main.java

```
1  public class Main{
2      public static void main(String args[]){
3          MobilMatic m = new MobilMatic();
4          m.jalankan();
5      }
6  }
```

Instansiasi pada Inheritance

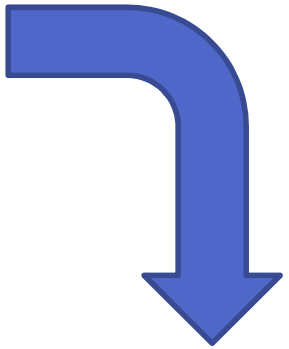
```
// Demonstrate when constructors are called.

// Create a super class.
class A {
  A() {
    System.out.println("Inside A's constructor.");
  }
}
// Create a subclass by extending class A.
class B extends A {
  B() {
    System.out.println("Inside B's constructor.");
  }
}

// Create another subclass by extending B.
class C extends B {
  C() {
    System.out.println("Inside C's constructor.");
  }
}
```



Buat Objek Class C

Instansiasi pada Inheritance

```
class CallingCons {
  public static void main(String args[]) {
    C c = new C();
  }
}
```



The output from this program is shown here:

```
Inside A's constructor
Inside B's constructor
Inside C's constructor
```

Keyword Super

- Keyword super digunakan untuk me-refer superclass dari suatu class, yaitu untuk merefer member dari suatu superclass, baik atribut maupun method.
- Super digunakan untuk memanggil konstruktor dari super class

Super: Konstruktor Pada Inheritance

Mobil.java | MobilMatic.java | Main.java

```
1  public class Mobil{
2      private String warna, merek;
3
4      public Mobil(String warna, String merek){
5          this.merek = merek;
6          this.warna = warna;
7      }
8      public Mobil(String merek){
9          this(merek, "Hitam");
10     }
11     public void jalankan(){
12         System.out.println("Masukkan gigi 1, lepaskan kopling sambil tekan gas");
13     }
14     public void setWarna(String warna){
15         this.warna = warna;
16     }
17     public String getWarna(){
18         return this.warna;
19     }
20 }
```

2 konstruktor

Super: Konstruktor Pada Inheritance

Mobil.java | MobilMatic.java | Main.java

```
1  public class MobilMatic extends Mobil{
2      @Override
3      public void jalankan(){
4          System.out.println("gigi di posisi D. Tekan gas");
5      }
6  }
```

→ Tanpa Konstruktor

Apa yang terjadi??

```
E:\Demo Slide\Slide 8>javac MobilMatic.java
MobilMatic.java:1: error: no suitable constructor found for Mobil()
public class MobilMatic extends Mobil{
       ^
    constructor Mobil.Mobil(String) is not applicable
      (actual and formal argument lists differ in length)
    constructor Mobil.Mobil(String,String) is not applicable
      (actual and formal argument lists differ in length)
1 error
```

Super: Konstruktor Pada Inheritance

Mobil.java | MobilMatic.java | Main.java

```
1  public class MobilMatic extends Mobil{
2      public MobilMatic(String warna){
3          super("Yaris");
4      }
5      @Override
6      public void jalankan(){
7          System.out.println("gigi di posisi D. Tekan gas");
8      }
9  }
```

Ambil paling
tidak 1 konstruktor
dari super class

Super: Memanggil Method

```
// Method overriding.
class A {
  int i, j;
  A(int a, int b) {
    i = a;
    j = b;
  }

  // display i and j
  void show() {
    System.out.println("i and j: " + i + " " + j);
  }
}
```

```
class B extends A {
  int k;

  B(int a, int b, int c) {
    super(a, b);
    k = c;
  }

  void show() {
    super.show(); // this calls A's show()
    System.out.println("k: " + k);
  }
}
```

Keyword super di B memanggil method show() di A

Super: Memanggil Method

```
class Override {
  public static void main(String args[]) {
    B subOb = new B(1, 2, 3);

   subOb.show(); // this calls show() in B
  }
}
```

```
i and j: 1 2
k: 3
```

Virtual method invocation

- Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan Overriding.
- Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap Overriding method pada subclass, sedangkan yang seharusnya dipanggil adalah overridden method.
- Pada VMI, atribut parent akan menutupi (hide) atribut dari child, sedangkan untuk method yang dijalankan tetap berstatus sama dengan overriding method.

Virtual method invocation

```java
public class Induk {

    private String attrib = "I";

    public void info() {
        System.out.println("Ini kelas Induk");
    }

    public String getAttrib() {
        return attrib;
    }
}
```

```java
public class Anak extends Induk{
    private String x = "A";

    @Override
    public void info() {
        System.out.println("Ini Kelas Anak");
    }

    public String getX() {
        return x;
    }

}
```

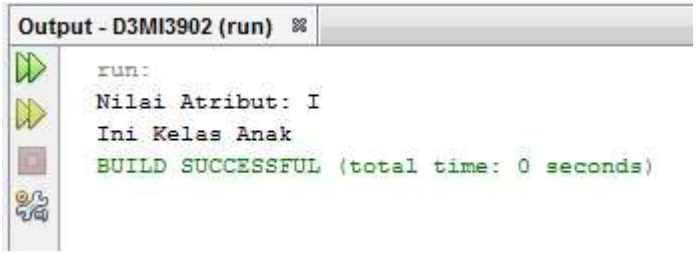
Virtual method invocation

```
public class Main {
    public static void main(String[] args) {
        Induk o = new Anak();
        System.out.println("Nilai Atribut: "+o.getAttrib());
        o.info();
    }
}
```

Perhatikan kode untuk membuat objek “o”

Virtual method invocation

Hasilnya apa??



END OF SLIDE…