ABSTRACT CLASS & INTERFACE

Topics

- Abstract class
- Interface

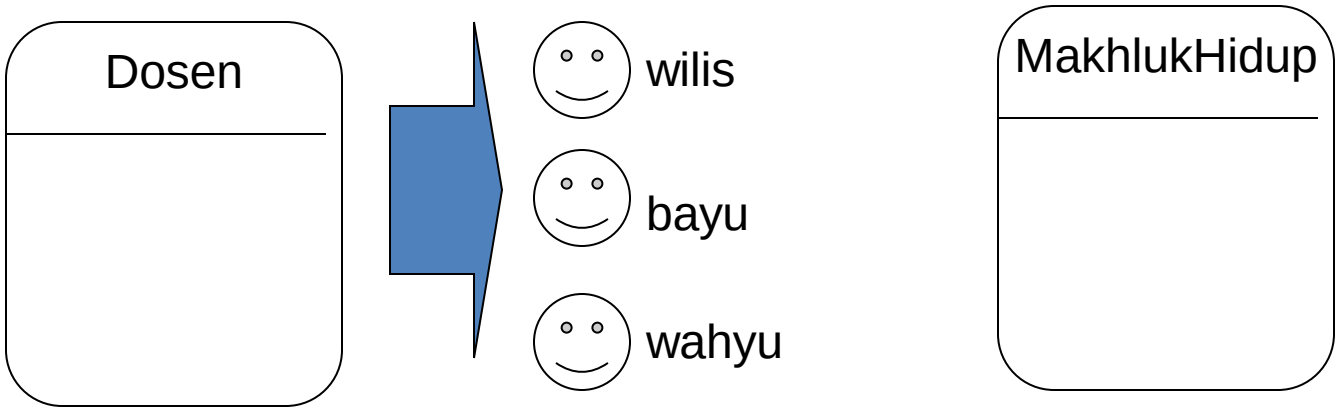
Apakah abstract class itu?

Abstract

- Saat kita membuat sebuah superclass, semua metode yang dimilikinya dapat digunakan pada class anaknya.
- metode yang dimiliki superclass dapat diperbarui pada subclassnya dgn cara **override pada metode tersebut.**
- **Case Studi**
  - dibuat sebuah superclass bernama **mahlukhidup**
  - salah satu methode yang dimiliki class **mahlukhidup** adalah bernafas
  - Class induk **mahlukhidup** tersebut akan diturunkan menjadi class manusia dan ikan, manusia bernafas dengan paru-paru, sedangkan ikan bernafas dengan insang
  - Bagaimana membuat methode bernafas tersebut (di class **mahlukhidup, manusia, ikan)**

Analogi Kelas yang Abstract

- Suatu kelas dapat diinstansiasi menjadi objek, misal kelas Dosen dapat diinstansiasi menjadi wilis, bayu, wahyu, namun sulit untuk menginstansiasi kelas MahlukHidup sebab kelas tersebut terlalu umum (abstract), kelas seperti inilah yang disebut kelas abstract. Dibutuhkan kelas turunan yang lebih khusus.



Analogi Method yang Abstract

- Bila kelas MahlukHidup mempunyai method bernafas, maka tidak dapat ditentukan cara suatu mahluk hidup tersebut bernafas (dengan paru-paru, insang, atau stomata), method seperti inilah yang disebut method abstract. Dibutuhkan kelas turunan yang khusus dan method override dari method yang abstract

Abstract method

- Method yang tidak memiliki implementasi (body)
- Untuk membuat abstract method, hanya tuliskan deklarasi method tanpa bodi, dan gunakan abstract keyword
  - tidak ada { }
- Sebagai contoh:
  - //ingat tidak ada { }
  - public abstract void someMethod();

Abstract Class

- Abstract class adalah class yang memiliki minimal satu abstract method
- Abstract class tidak bisa dibuat instance-nya (tidak bisa dibuat objectnya)
  - //kode di bawah ini menyebabkan compilation error

```
MyAbstractClass a1 = new MyAbstractClass();
```

- Untuk menggunakan abstract class, digunakan class lain (concrete class)

  -concrete class harus mengimplementasi semua abstract method

  -concrete class menggunakan extends keyword

Contoh abstract class

```
public abstract class MakhlukHidup {
        public void bernafas(){
            System.out.println("Makhluk hidup bernafas...");
        }
        public void makan(){
            System.out.println("Makhluk hidup makan...");
        }
        public abstract void berjalan();
}
```

Extending abstract class

- Ketika concrete class mengimplementasi abstract class MakhlukHidup, maka concrete class tersebut harus mengimplementasi method berjalan(). Jika tidak, maka class tersebut akan jadi abstract class juga, sehingga tidak dapat diinstansiasi / dibuat objectnya.
- Contohnya

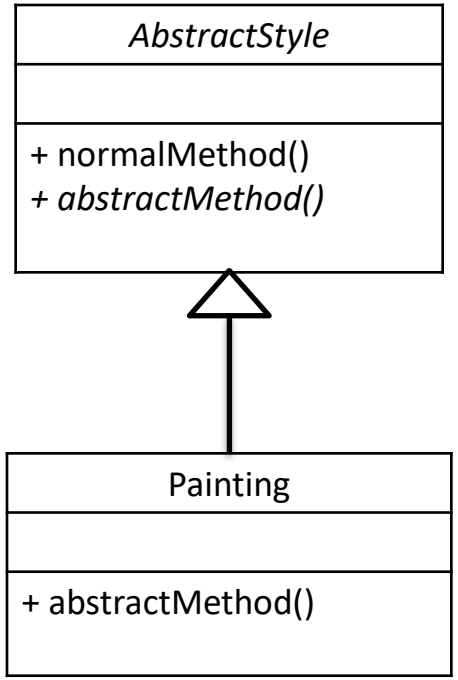
```
public class Manusia extends MakhlukHidup {
    public void berjalan(){
        System.out.println("Manusia berjalan...");
    }
}
```

Kapan menggunakan Abstract Method dan Abstract Class?

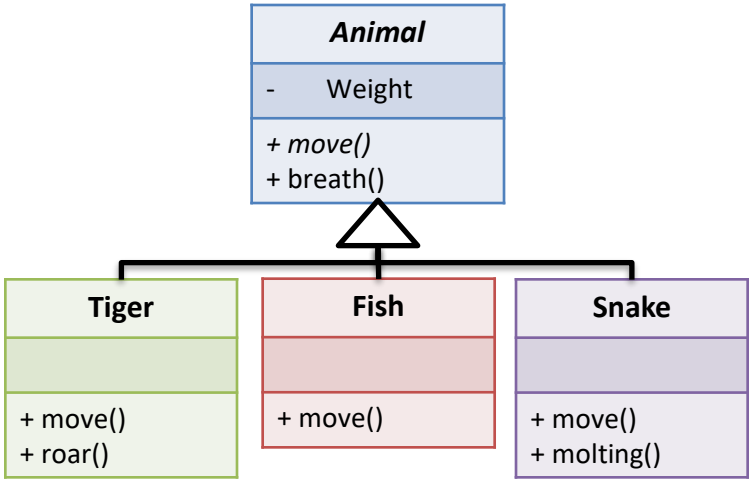
- Abstract method digunakan ketika dua atau lebih subclass dirancang untuk memenuhi aturan yang sama dengan implementasi yang berbeda
  - Subclass-subclass ini mengextend abstract class yang sama namun dengan implementasi yang berbeda pada abstract methodnya.
- Menggunakan abstract class sebagai konsep umum pada top level class hierarki, dan menggunakan banyak subclass untuk membuat detail implementasi abstract class

Abstract Class Diagram

- Class diagram of an abstract
  - Italic class name
  - Italic method name if abstract
- Child extends the abstract parent class
  - Child must implement all abstract method
  - Or child will be declared abstract too



Example: Animal Hierarchy



Both tigers, fish, and snakes are animals and they all can move, but their 'moving' behavior are really different

We don't need to specify the behavior in animal class, let the child classes define themselves

another view:
We make sure that every animal can move

Contoh Implementasi

```
public abstract class Printer {
    public abstract void printing();
    public void scanner(){
    }
}
```

```
public class Canon extends Printer{
    public void printing(){
        System.out.println("print pakai tinta");
    }
}
```

```
public class Samsung extends Printer {
    public void fotocopy(){
        System.out.println("Bisa buat fotocopy");
    }
    public void printing(){
        System.out.println("print pakai carbon");
    }
}
```

Latihan

Buatlah sebuah class abstract bernama Mahasiswa memiliki method sbb :

- isiBiodata(String nama, int nilai) sebagai method abstract
- registrasi(), menampilkan tulisan ke layar : “nama.....telah melakukan registrasi”
- testMasuk(), menampilkan tulisan ke layar : “nilai test anda .....”

- Tambahkan variabel nama dan nilai pada Mahasiswa
- Buatlah class MahasiswaBaru yang melakukan extending dari class Mahasiswa. Deklarasi ulang method isiBiodata, bodi method akan menampilkan ke layar biodata anda (“nama:....., nilai : .....)
- Tambahkan main method pada MahasiswaBaru sehingga output sbb :

```
run:
nama ahmad ,nilai : 80
nama ahmad telah melakukan registrasi
nilai test anda 80
BUILD SUCCESSFUL (total time: 1 second)
```

Apakah interface itu?

Interface


XBOX KINECT GAMES:
STAR WARS / MICHAEL PHELPS:
PUSH THE LIMIT
One asks you to dance DDR-style with Stormtroopers. The other has you mimic swim strokes while standing up.
SAMSUNG SMART TV
Remote-control haters can log on to the $3,500 web-enabled TV by showing their face, and change channels using their voice.
GOOFY

Interface

- Menggambarkan cara standard tentang behavior-behavior maupun atribut yang dimiliki oleh class.
- Memungkinkan bagi class, dimanapun dia berada, untuk mengimplementasi standar tingkah laku
- Untuk model multi inheritance yang mengijinkan sebuah class dapat memiliki lebih dari satu superclass.

Interface

- Diawali dengan keyword "**interface**". Ingat! Bukan keyword "class".
- Berisi method abstract (dapat juga dilengkapi dengan konstanta).
  - Mendefinisikan tingkah laku dari sekumpulan method, tanpa ada body (implementasi dari method)

Interface

- Like an abstract class, interface cannot be instantiated
- A class that implements interface(s) must implements all abstract methods
- An abstract class can also implements interface(s)
- An abstract class that implements interface(s) may not implements all abstract methods

Interface

- Practical use of interfaces enables the separation of the interface and implementation of an object
  - The interface is provided by the interface definition
  - The implementation is provided by the classes which implement the interface methods

Interface

- Interface usually(not always) named **"…… - able"**
  - Serializable
  - Runnable
  - Cloneable
- Meaning that the implementing class able to do anything as the interface told
  - Implement interface to be able of doing some predefined tasks

Interface

- Implementing an interface allows a class to become more formal about the behavior it promises to provide
- Interfaces form a contract between the class and the outside world,
  - this contract is enforced at build time by the compiler.

When to use/create Interface

- Define some rules or mechanisms
- Group classes with no inheritance relationship

Deklarasi Interface

- Syntax :

```
public interface [InterfaceName] {

        //beberapa method tanpa body

}
```

- Contoh :

```
public interface Facsimile{

}
```

Contoh Interface

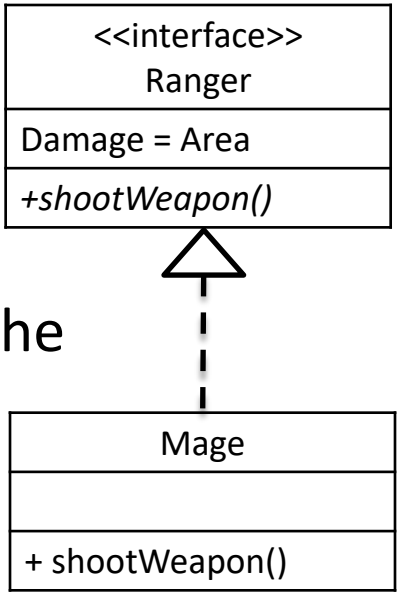
```
public interface Facsimile{
    int dotPerInches = 90;

    boolean dialPhone (String number);
    void getDocument(int nDoc);
    boolean sendDocument(int nDoc);
    void closePhone();
}
```

- Interface facsimile memiliki member : 1 konstanta, 4 method abstract.

Interface Class Diagram

- Class diagram of an interface
  - Tag <<interface>>
- When class A implements Interface, the diagram was denoted as

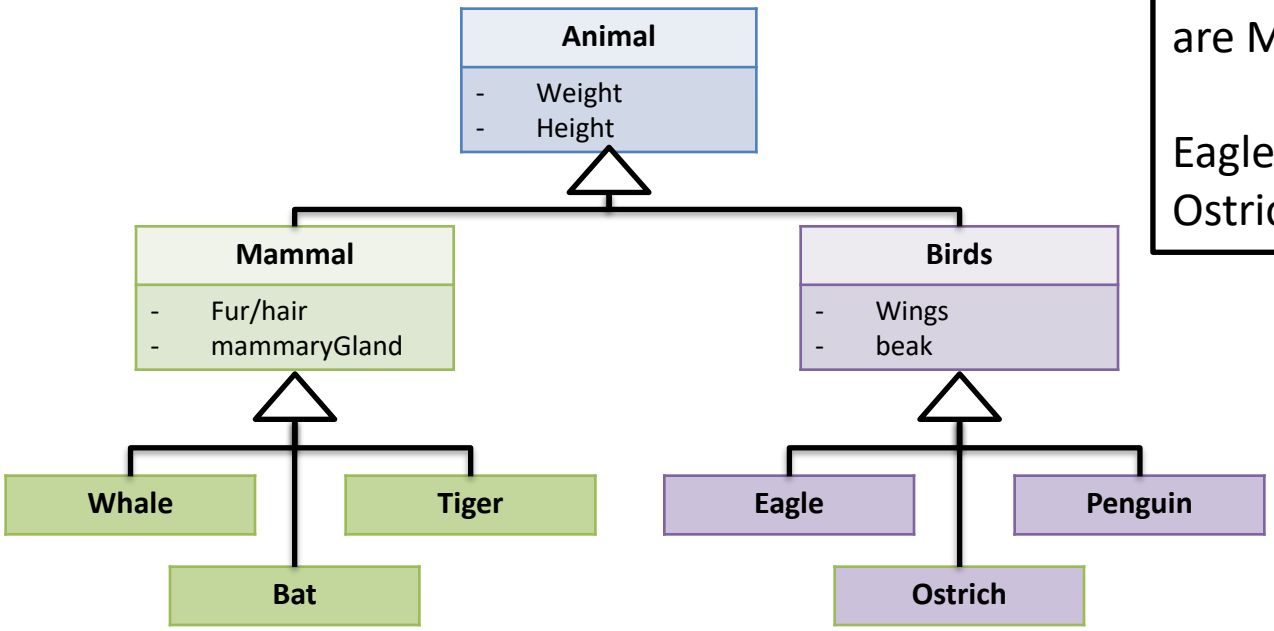


Example: Animal Hierarchy

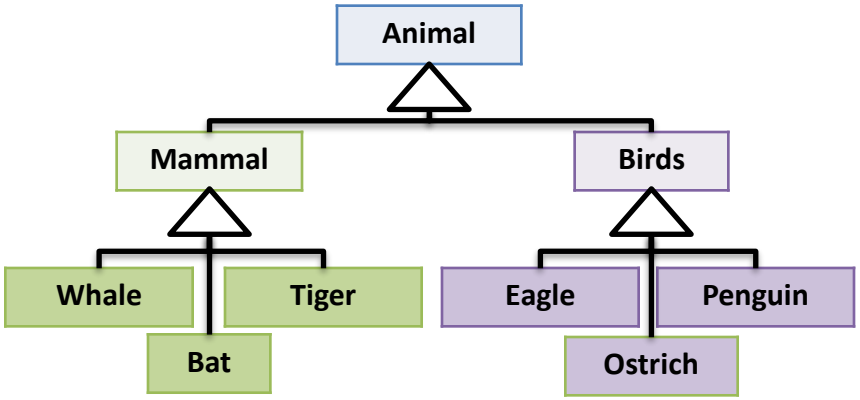
Birds and Mammals are animal

Whale, Tiger and Bat are Mammal

Eagle, Penguin, Ostrich are Bird



Example: Animal Hierarchy

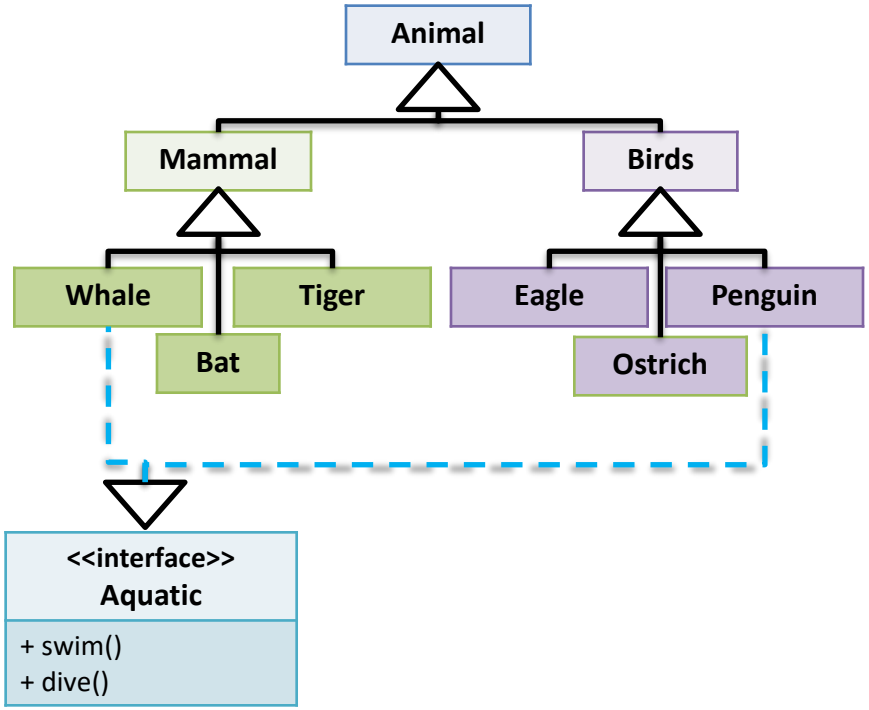


whales and penguins are aquatic animals and can swim, but bats, tigers, eagles and ostriches can not

we **can not** put a swimming behavior in mammal, bird or animal class as not all mammals, birds and animals can swim

We also **can not** put penguins and whales have the same parent class as they both are different

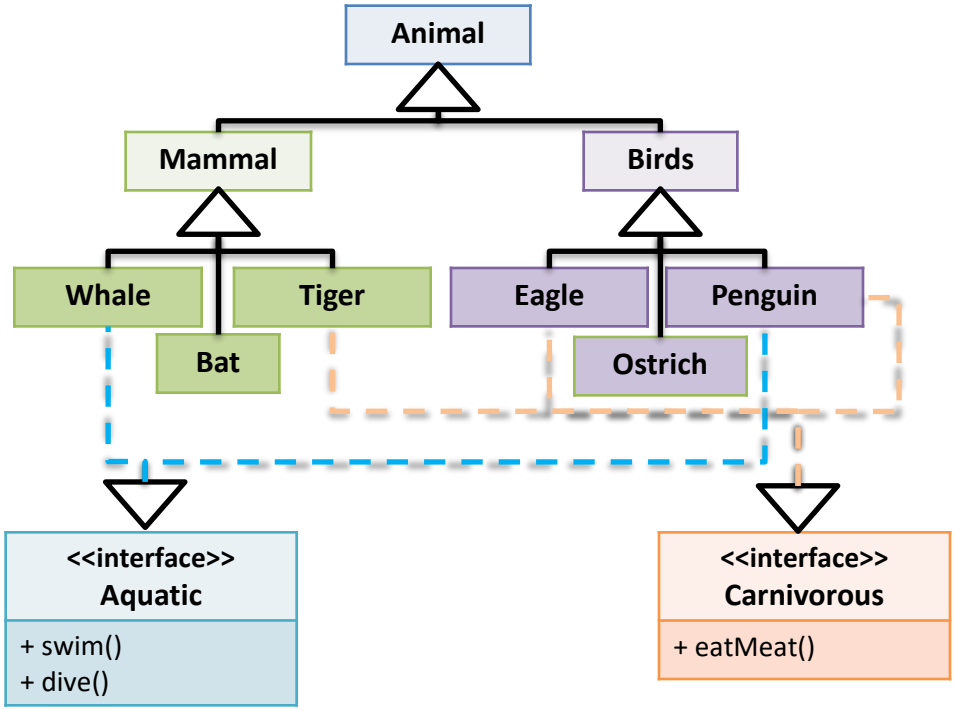
Example: Animal Hierarchy



Here we can create an Aquatic Animal Interface that defines that animals that implements this interface will be able to swim and dive

Then we can make the whales and penguins implement Aquatic interface behavior

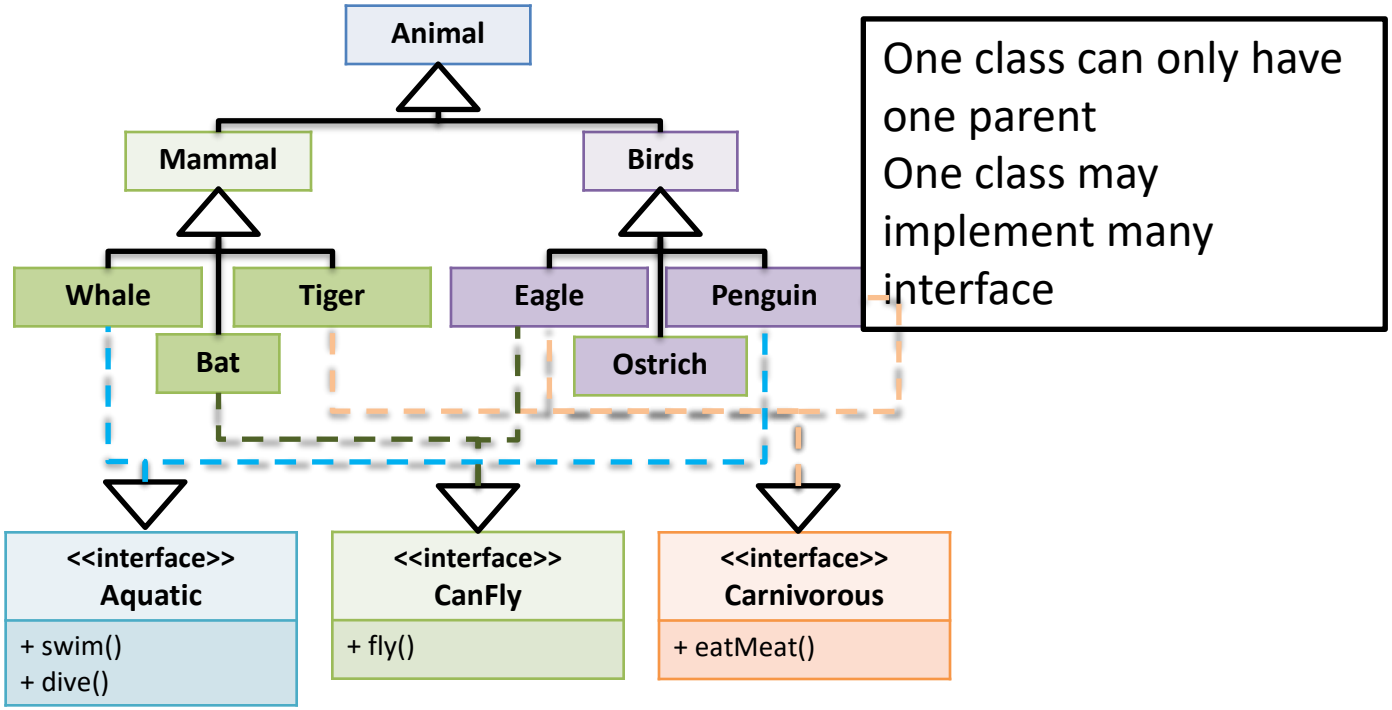
Example: Animal Hierarchy



Another example, tigers, eagles and penguins are carnivorous, while the rest are not

Create a Carnivorous Interface, and make the carnivorous animals implement the behaviors

Example: Animal Hierarchy



Mengimplementasikan Interface

Implementasi Interface

- Untuk membuat concrete class yang mengimplementasi interface, gunakan keyword implements
- Syarat : seluruh method abstract yang ada pada interface harus dideklarasi ulang didalam class yang mengimplementasikannya.
- Syntax :

```
class [className] implements [interfaceName]
{
        //overiding abstract method
        //dan beberapa kode yang lain
}
```

Contoh : Implementasi Interface

```
public interface Facsimile{
    int dotPerInches = 90;
    boolean dialPhone (String number);
    void getDocument(int nDoc);
    boolean sendDocument(int nDoc);
    void closePhone();
}
```

```
public class Printer implements Facsimile{

    boolean dialPhone (String number){
       return true; }
    void getDocument(int nDoc){   }
    boolean sendDocument(int nDoc){
       return true; }
    void closePhone(){   }

}
```

Multi-implements

- Sebuah class dapat melakukan implements lebih dari satu interface.
- Contoh :

```
class Person implements PersonInterface,LivingThing,
                        WhateverInterface {

   //beberapa kode disini
}
```

Class Person implements terhadap 3 interface sekaligus, yaitu : PersonInterface, LivingThing dan WhateverInterface.

Latihan

- Buatlah abstract class “MakhlukHidup”, memiliki method : bernafas() return type boolean, dan method berbicara() yang merupakan abstract method.
- Buatlah interface “Aves” memiliki method : terbang() memiliki return type boolean, sayap=true
- Buatlah interface “Kuda” memiliki atribut kaki=4, method : berjalan()
- Buatlah class “Elang” yang melakukan extending pada class MakhlukHidup dan interface Aves.
- Buatlah class “KudaTerbang” yang mengimplementasikan abstract class MakhlukHidup dan kedua interface diatas.

Inheritance pada Interface

- Interface dapat memiliki hubungan inheritance antar mereka sendiri.
- Menggunakan keyword **“extends”**.
- Contoh :

  interface PersonInterface merupakan superinterface dari student interface.

```
public interface PersonInterface {
      . . .
}

public interface StudentInterface extends PersonInterface
{
      . . .
}
```

Implements SubInterface

- Sebuah class yang mengimplementasikan subinterface wajib mendeklarasi ulang seluruh method yang ada pada subinterface tersebut dan juga method yang ada pada superinterface-nya.
- Contoh :

```
Class Mahasiswa implements StudentInterface {
      . . .
}
```

- Karena **StudentInterface** merupakan subclass dari PersonInterface maka class Mahasiswa harus mendeklarasi ulang semua method abstract yang dimiliki oleh kedua interface tersebut.

Contoh Multiple-Implements

```
public interface PersonInterface {
    public void setNama(String nama);
}
```

```
public interface StudentInterface extends
PersonInterface{
    public void setSekolah(String sekolah);
}
```

```
public abstract class Mahasiswa implements
                                        StudentInterface
{

    public abstract void isiBiodata(String nama, int nilai);

    public void setSekolah(String skul){

    }
    public void setNama (String nam){

    }
}
```

Latihan

- Terdapat sebuah interface dengan deklarasi berikut :

```
public interface MesinATM
{
    void keluarkanUang(double nominal);
}
```

- Berdasarkan interface di atas, buatlah : Kelas bernama ATMBankSukses yang mengimplementaskan interface MesinATM. Kelas ATMBankSukses memiliki :
  - Atribut (private) : kodeTransaksi (String) dan tanggalTransaksi (Date)
  - Konstruktor (kodeTransaksi, tanggalTransaksi)
  - Method get dan set untuk seluruh atribut
  - Implementasi method keluarkanUang adalah menampilkan ke layar : “Anda telah melakukan pengambilan uang sebesar [nominal]”.

Kesimpulan

- Untuk membuat interface keyword yang dipakai adalah **interface**.
- Sedangkan untuk menggunakannya keyword yang dipakai adalah **implements**.
- Interface dapat digunakan untuk mengatasi permasalah multiple inheritance pada java.