Fakultas Informatika
School of Computing
Telkom University

CII3B4
Pemrograman Berorientasi Objek

Abstract Class and Interface

Abstract

Abstract Method

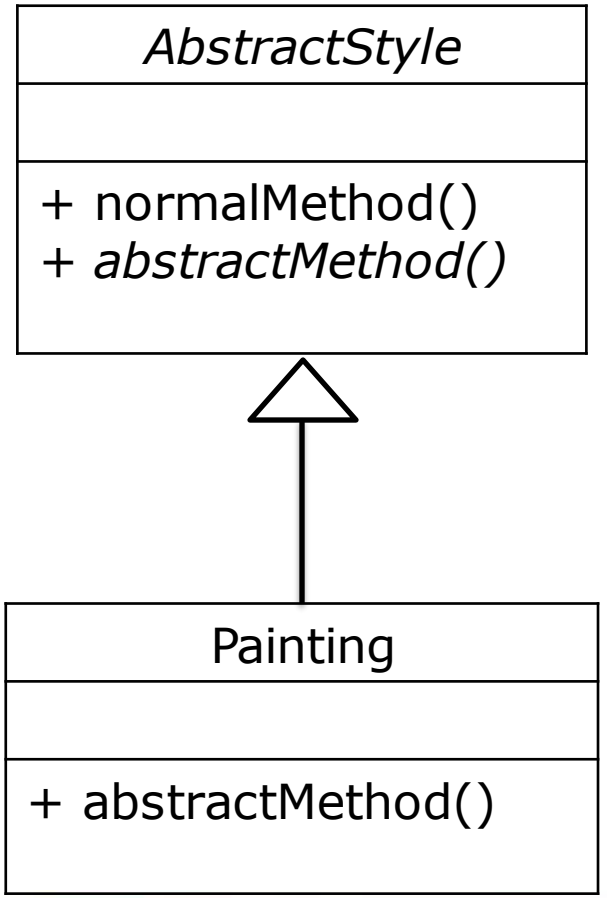
- Method with no implementation or specific behavior
- Handed over to the child class to implement its own
  - Parent class only declare the name of method
- Child class **must** implement or specify the method
  - Full overriding

Abstract Class

- A class with at least one abstract method must be declared abstract class
- An abstract class cannot be instantiated
  - Because abstract class has abstract method(s), the method which aren't implemented. An instance must have all methods already implemented
- Abstract class will ensure the child implements the declared abstract method
- Use "extends" like inheritance

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

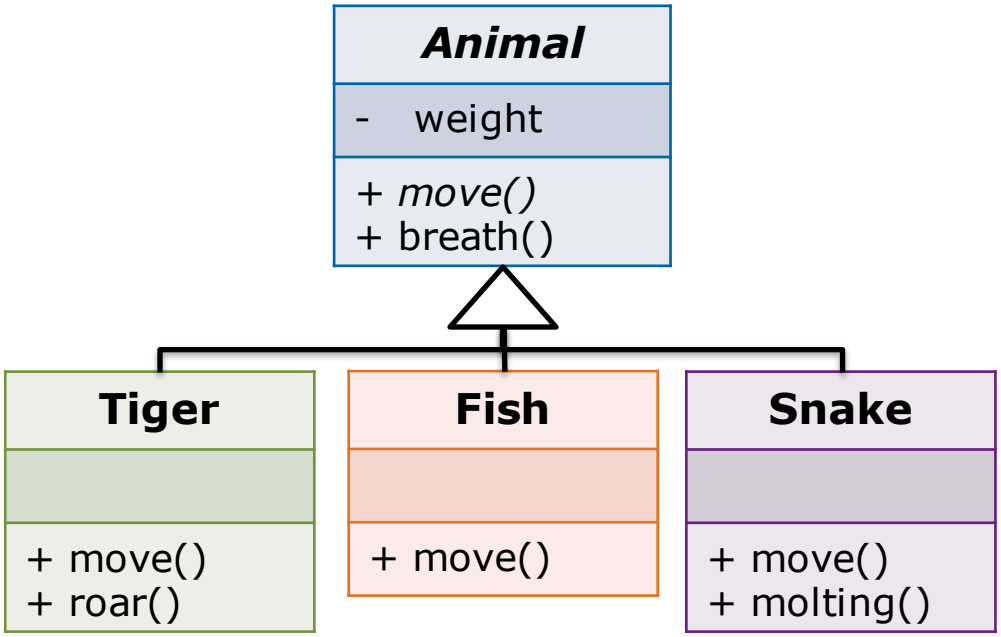
```
public abstract class AbstractParent{

   public String toString(){
      return "this is class Parent";
   }

   public abstract void abstractMethod();

}
```

```
public class Child extends AbstractParent{

   public void abstractMethod(){
      // implementing abstract method
   }

}
```

When to use/create Abstract class

- The implementation of some methods are too vary for each child
- No concrete/actual object for the parent class
  - No such object exists
  - Parent doesn't need to be instantiated
- Make a condition that the child class will have to implement some specific methods

Example: Animal Hierarchy

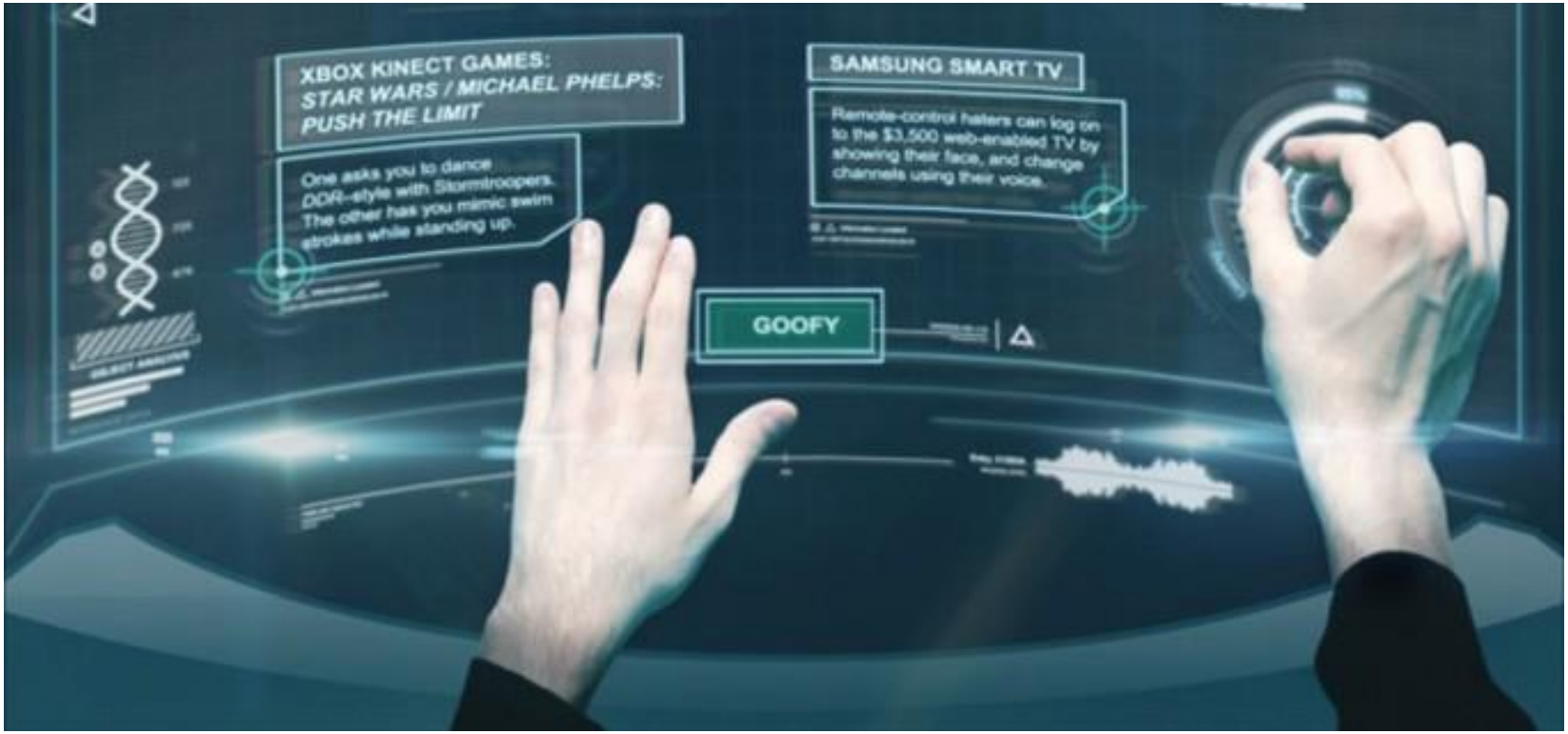


Both tigers, fish, and snakes are animals and they all can move, but their 'moving' behavior are really different

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

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

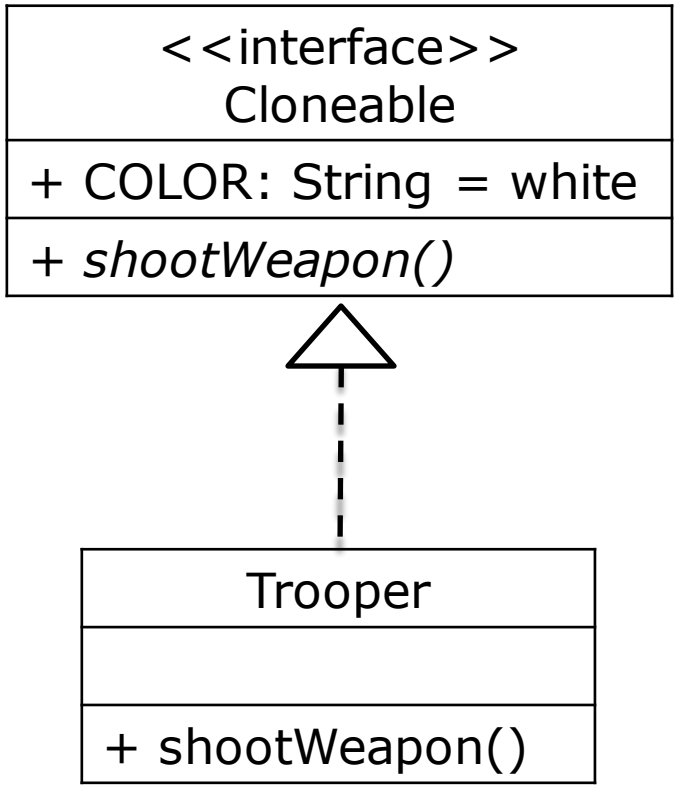
- A special type of class which define a set of method prototypes
- To deliver a rule or mechanism to it's child (class that implements)
- An interface can only contain
  - public, abstract method
  - public, static, final attributes
  - No constructor

Interface

- Use keyword "implements"
- A class
  - Can "extend" only one class i.e. only one superclass
  - Can "implements" multiple interfaces
- Interfaces don’t have instance fields
  - they are merely a set of methods that the object implementing the interface methods must define

Interface Class Diagram

- Class diagram of an interface
  - Tag <<interface>>



- When class A implements Interface I, the diagram was denoted as

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 (set of rules) is provided by the interface definition
  - The implementation is provided by the classes which implement the interface methods

Interface

- In Java, Interface usually named **"..... - able"** (but not always)
  - 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.

Example

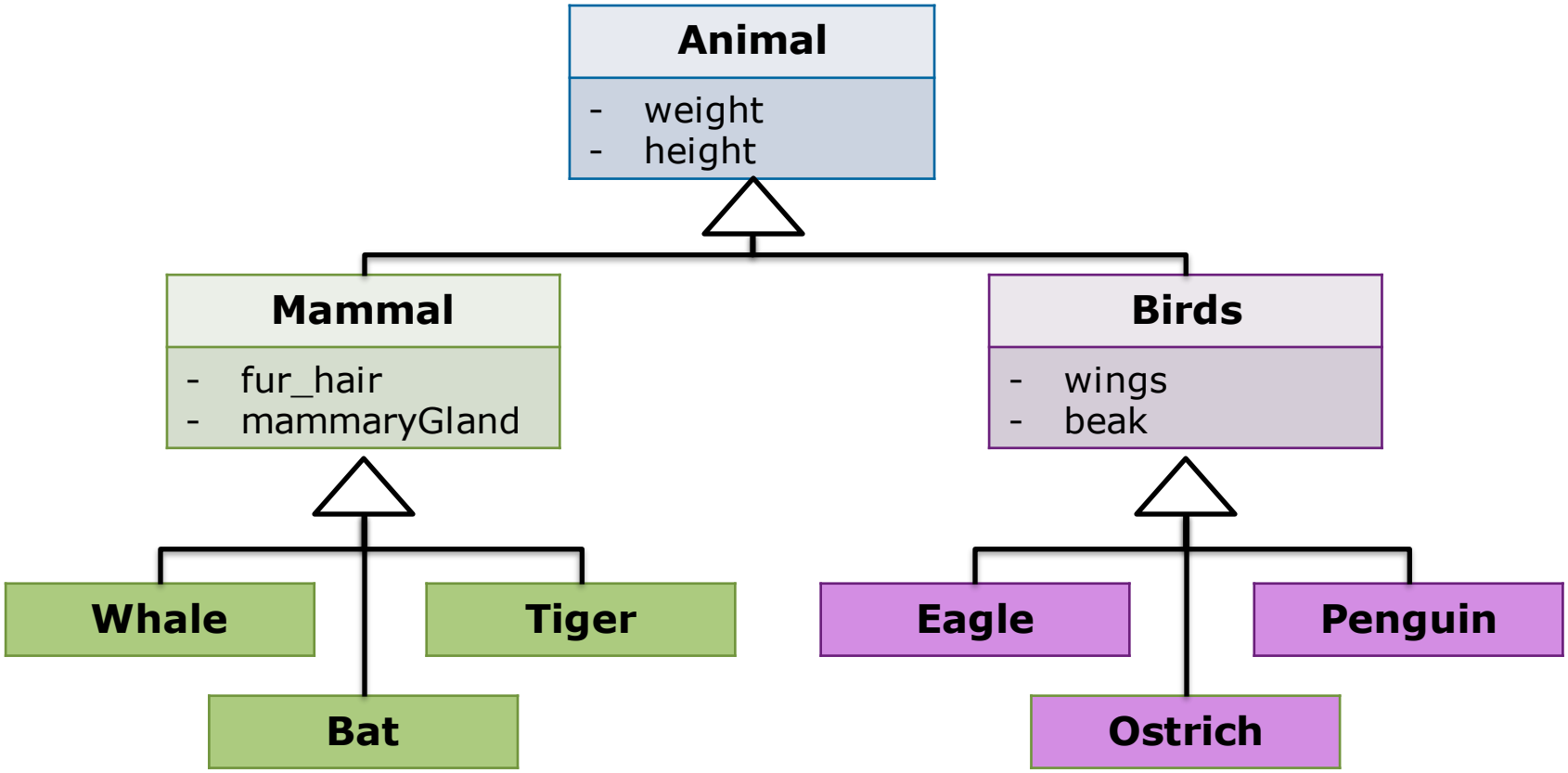
```
interface Instagramable{

   public abstract void takePicture();
   public abstract void setFilter();
   public abstract void sharePhoto();
}
```

```
public class Kitten implements Instagramable{

   public void takePicture(){
      // implementing abstract method
   }

   public void setFilter(){
      // implementing abstract method
   }

   public void sharePhoto(){
      // implementing abstract method
   }
}
```

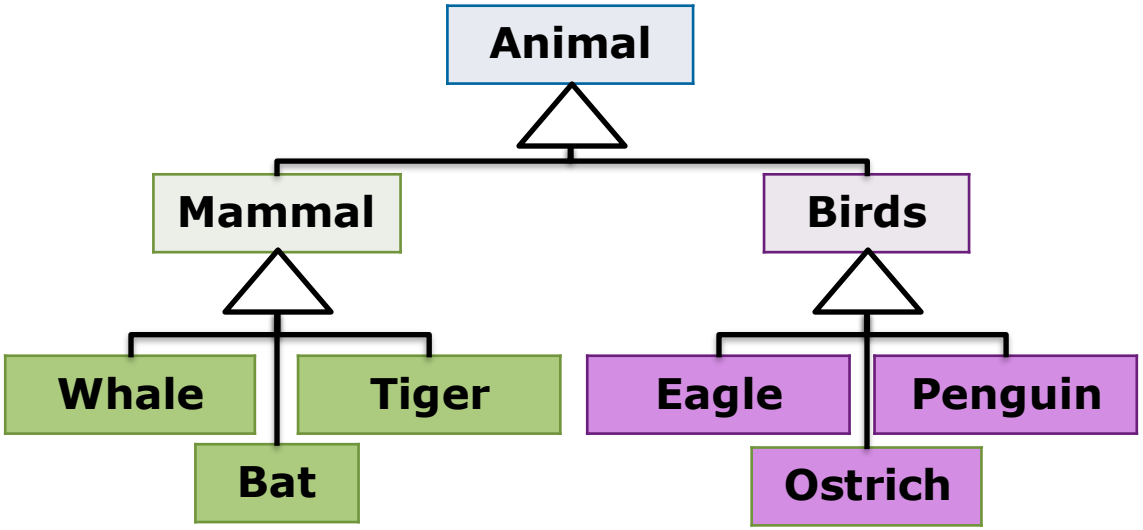
When to use/create Interface

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

Example: Animal Hierarchy



Example: Animal Hierarchy

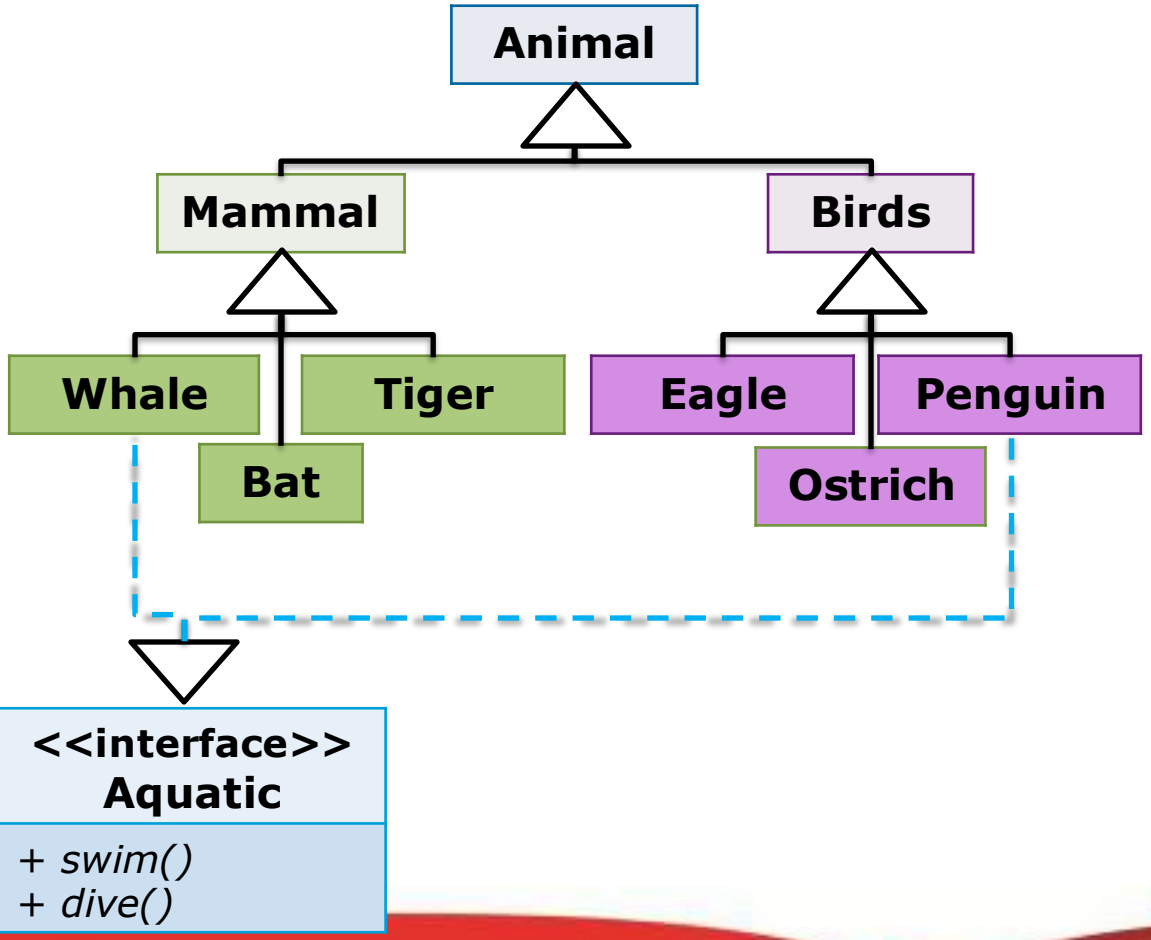


whales and penguins are aquatic animals and can swim, but bats, tigers, eagles and ostriches can not

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

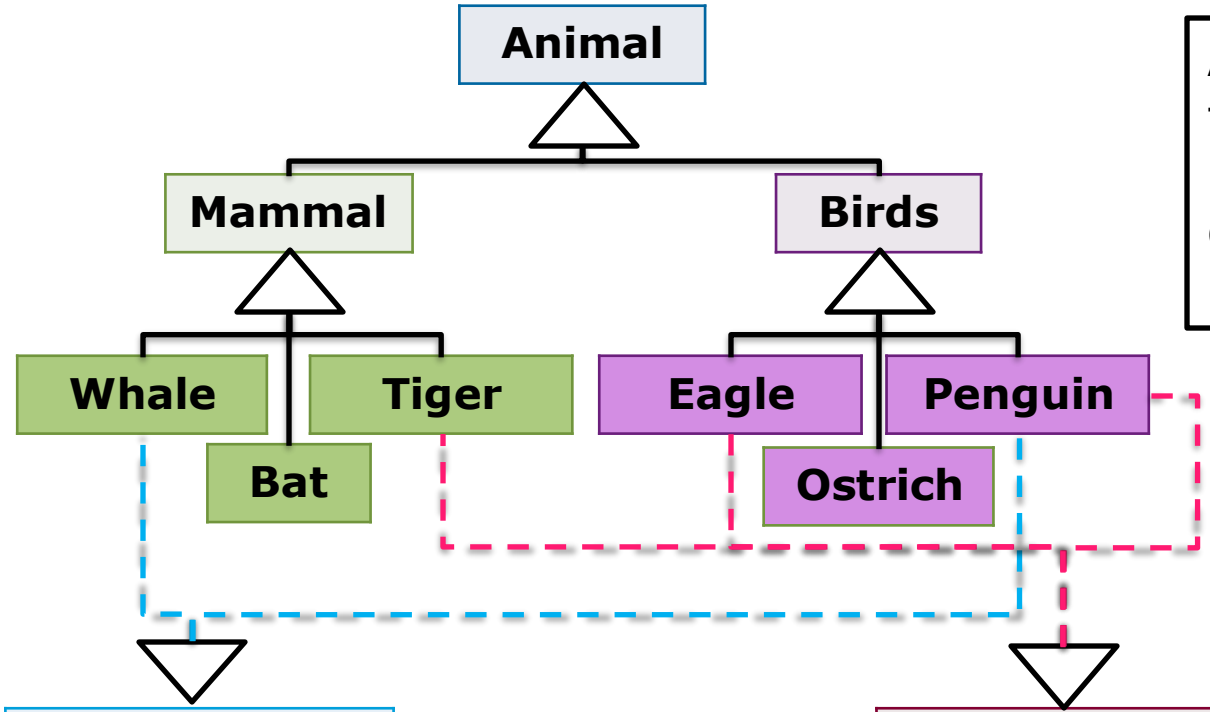
Example: Animal Hierarchy



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

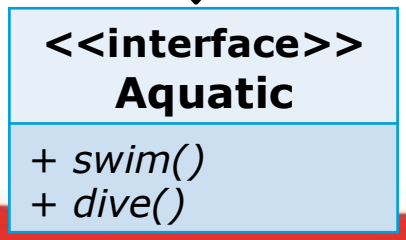
Then we can make the whales and penguins implement Aquatic interface behavior

Example: Animal Hierarchy

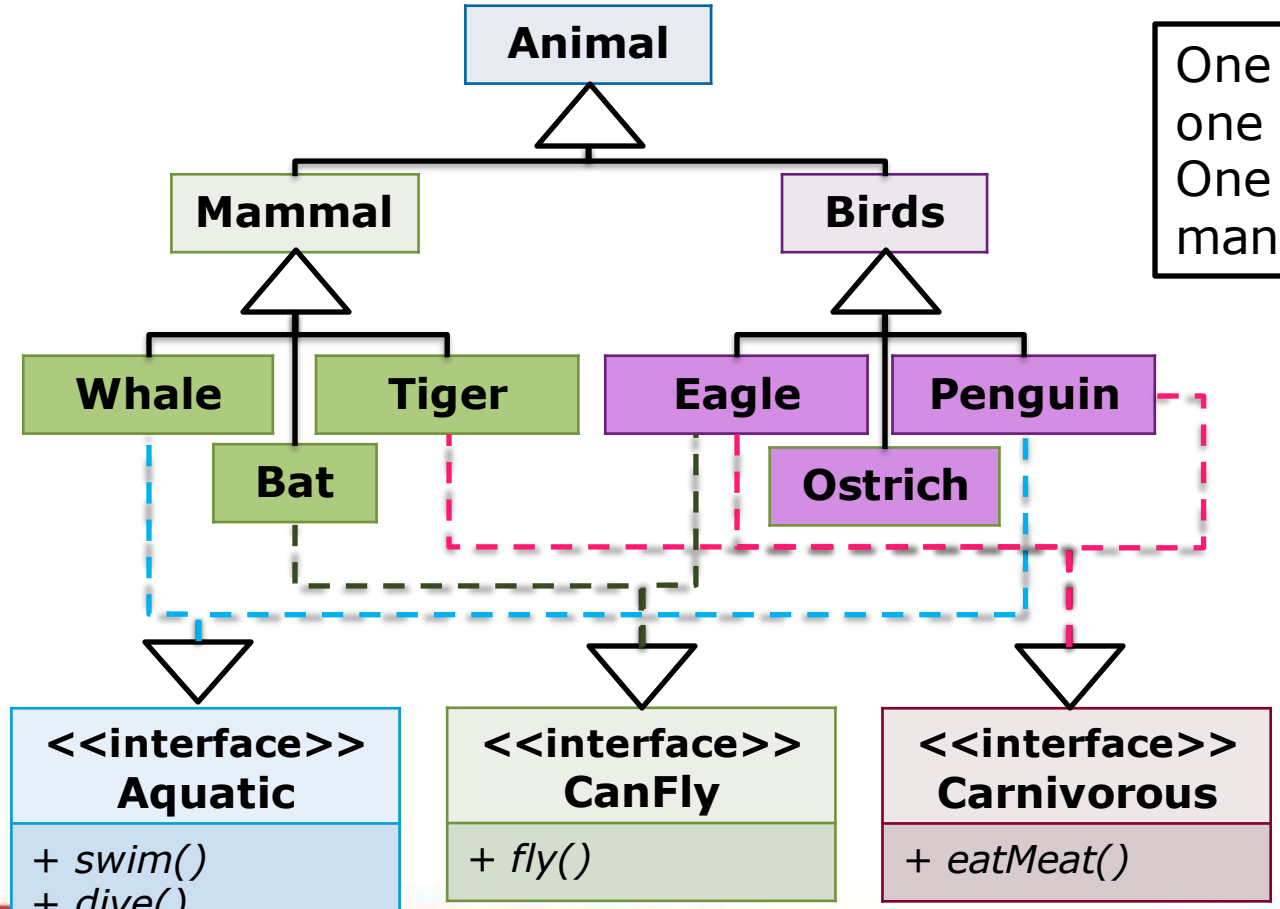


Another example, tigers, eagles and penguins are carnivorous, while the rest are not

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

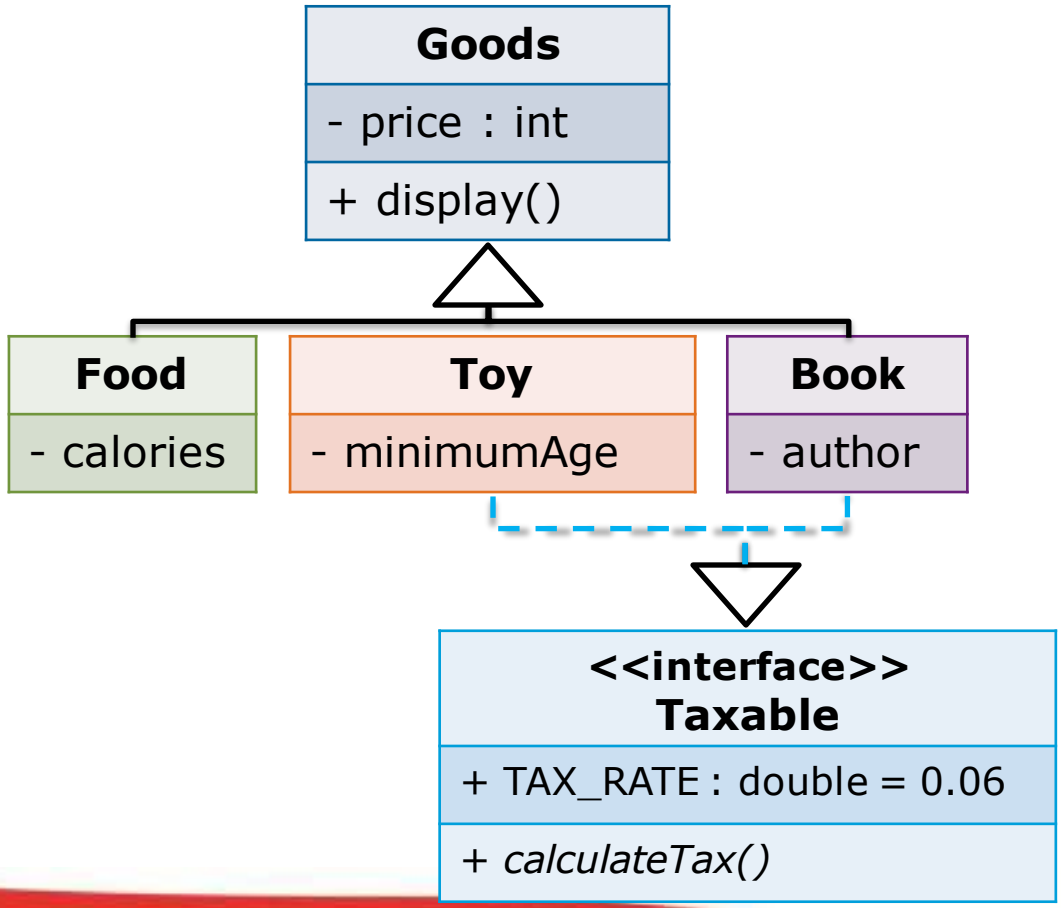


Example: Animal Hierarchy



One class can only have one parent
One class may implement many interface

Example: Taxable Goods



Toys and Books are classified as taxable goods, while foods are not

Question?

Fakultas Informatika
School of Computing
Telkom University

THANK YOU