

CISC 3115

Interface

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

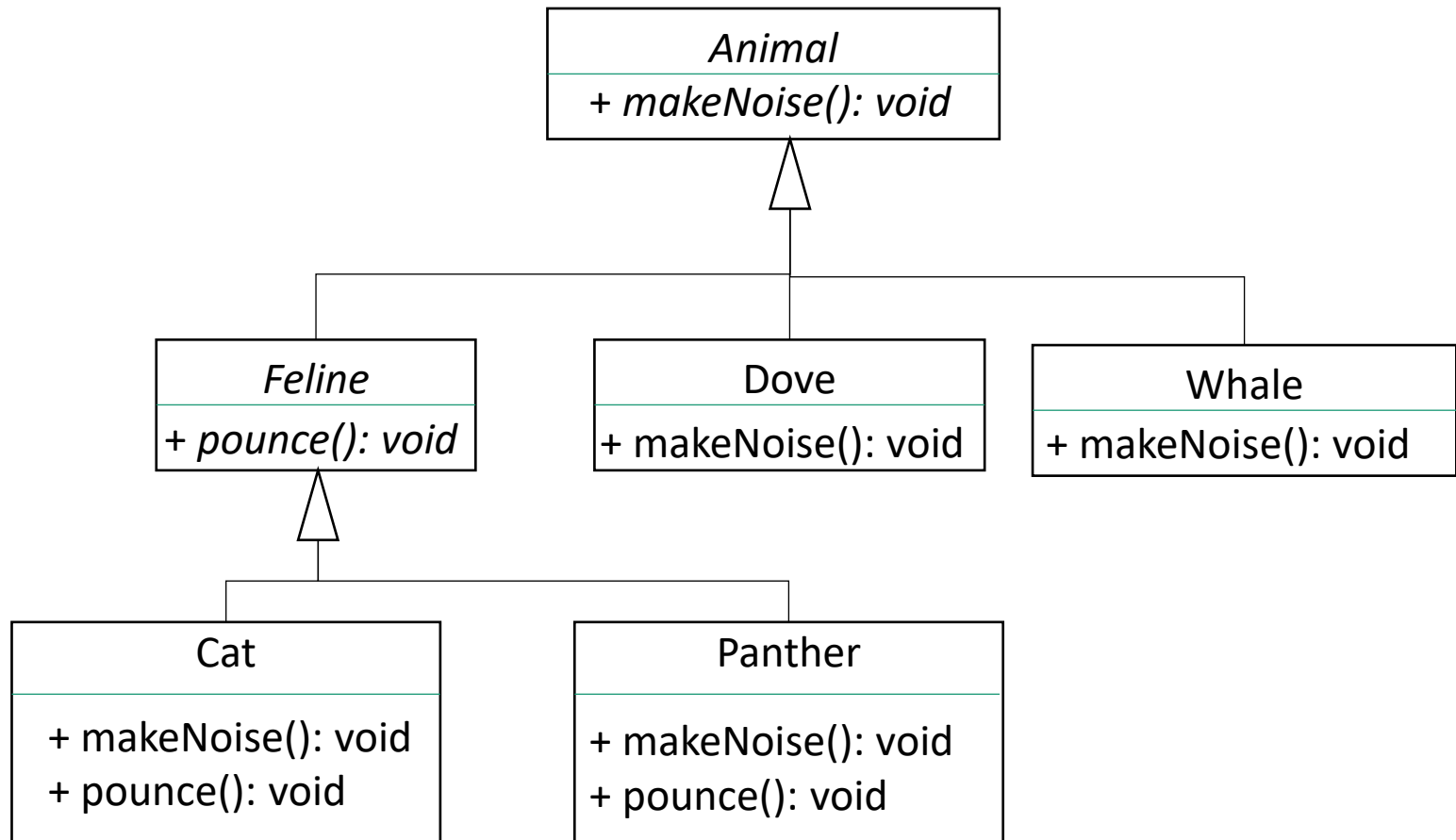
Outline

- Recap
 - Inheritance and polymorphism
 - Abstract method and class
- Interface
 - Motivation
 - Define interface
 - Extend interface
 - Implement interface
 - Use interface as data type
- Exercises

Different Classes, Same Behaviors

- Different classes, although vastly different, may exhibit similar behavior
 - Any communication devices can “transmit” and “receive”
 - Any vehicles can “move”
 - Any objects can be “compared” to each other
 - Any objects may be cloned
 -
- Using subclasses (inheritance via subclass) may be too rigid for this kind of flexibility in real life.

Let's consider an Animal class hierarchy (for a computer game)



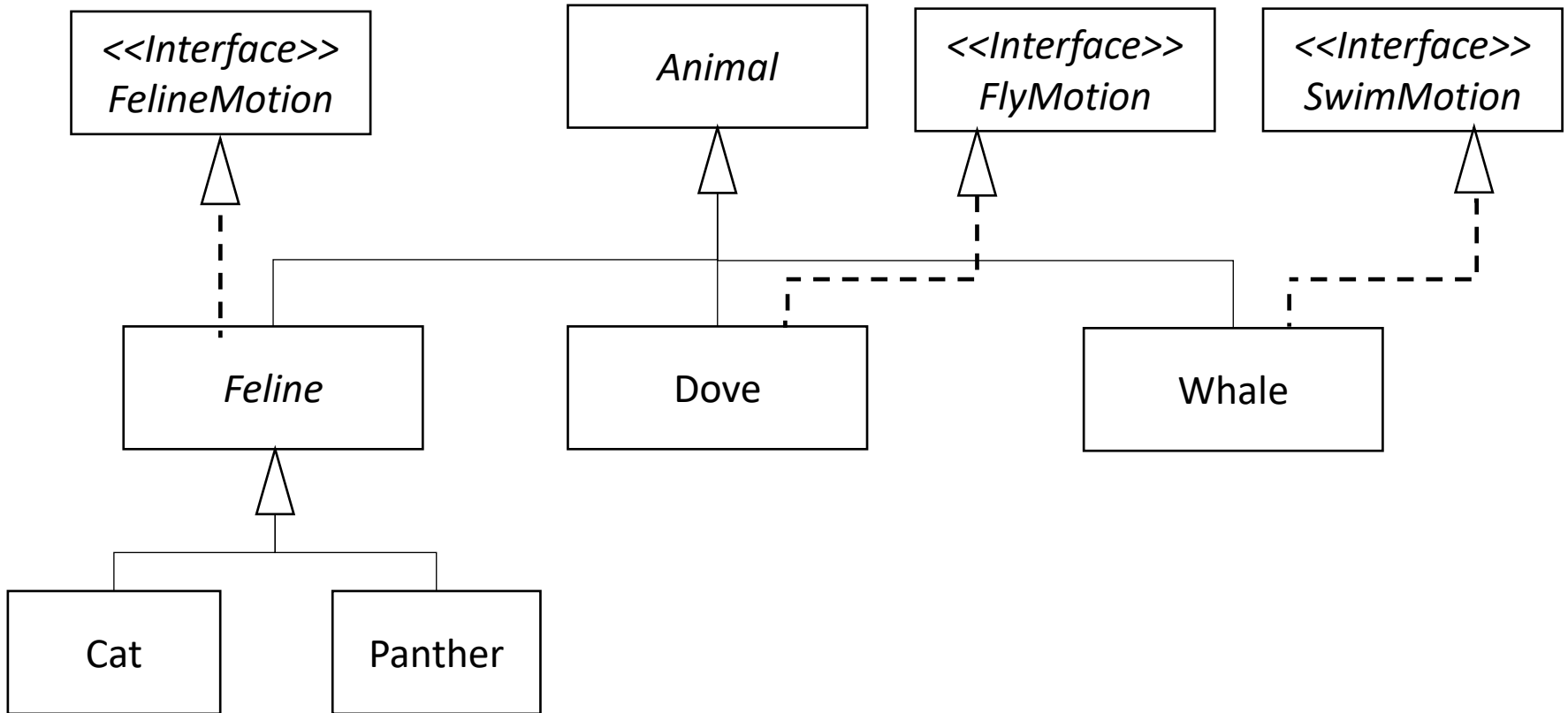
Improving the Animal Class Hierarchy?

- Different animals have different motions
 - Birds fly
 - Whales swim
 - Cats pounce
- But,
 - Can a bat fly? Can an insect fly?
 - Does every bird fly? Does every insect fly?
 - Can a dog swim? Can a bird also swim?
- ...Using subclasses (inheritance via subclass) may be too rigid for this kind of flexibility in real life.

Introducing Java Interface

- Not the “interface” in “Graphical User Interface”
- Java has a reference type, called interface
 - Contain abstract methods only.
 - Java 8 introduces the concept of default methods and permits static methods (abstract methods with default implementation)
 - At this moment, pretend this does not exist (so that we aren’t distracted from the discussion that follows).
 - Define only one or more behaviors

The Improved Animal Class Hierarchy



How to Defining Interfaces: Birds Fly, Whales Swim, ...

```
public interface FlyMotion {  
    public void fly(Direction direction, double speed, double distance);  
}
```

```
public interface SwimMotion {  
    public void swim(Direction direction, double speed, double distance);  
}
```

```
public interface FelineMotion {  
    public void jump(Direction direction, double speed, double distance);  
    public void pounce(Animal prey);  
}
```


Implementing Interfaces

```
abstract class Feline implements FelineMotion {
```

```
...
```

```
    public void jump(Direction direction, double speed, double distance) { ... }
```

```
    public void pounce(Animal prey) { ... }
```

```
...
```

```
}
```

```
class Dove extends Animal implements FlyMotion { ...
```

```
    public void fly(Direction direction, double speed, double distance) { ... }
```

```
}
```

Questions?

- Interface
 - What is it?
 - Why?
 - How?
 - Implement interface
- Examples

Interface: Remark

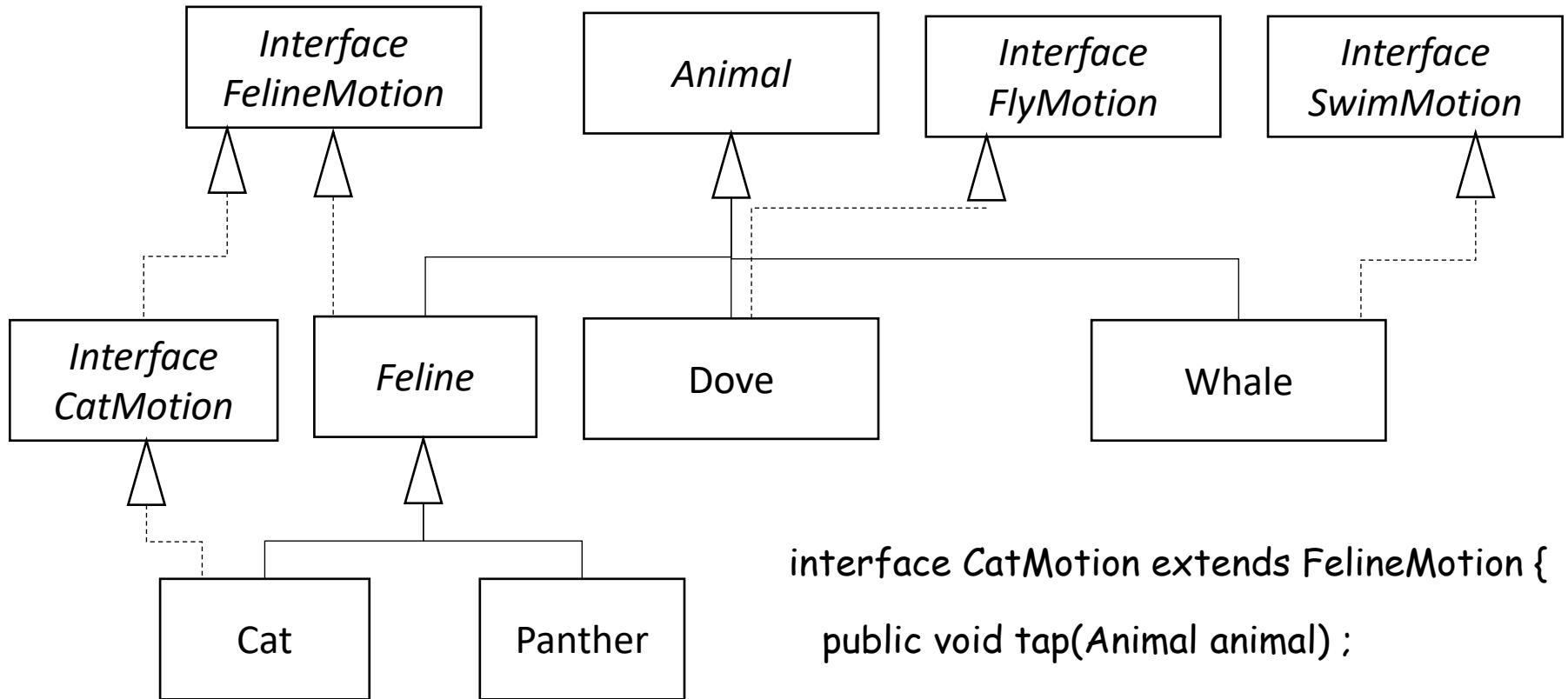
- Interfaces are like abstract classes, cannot be instantiated
 - can only be implemented by classes or extended by other interfaces
- “implement” and “extend” are two distinct Java terms, such as,
 - A class (the class) “implements” an interface (the class provides an implementation of the interface)
 - A class (the class) “extends” another class (the class becomes a subclass of the other)
 - An interface (the interface) “extends” another interface (define new behavior in the interface)

Evolving Interfaces

- Interfaces can be extended (like classes)

```
interface CatMotion extends FelineMotion {  
    public void tap(Animal animal) ;  
}
```

Extending FelineMotion

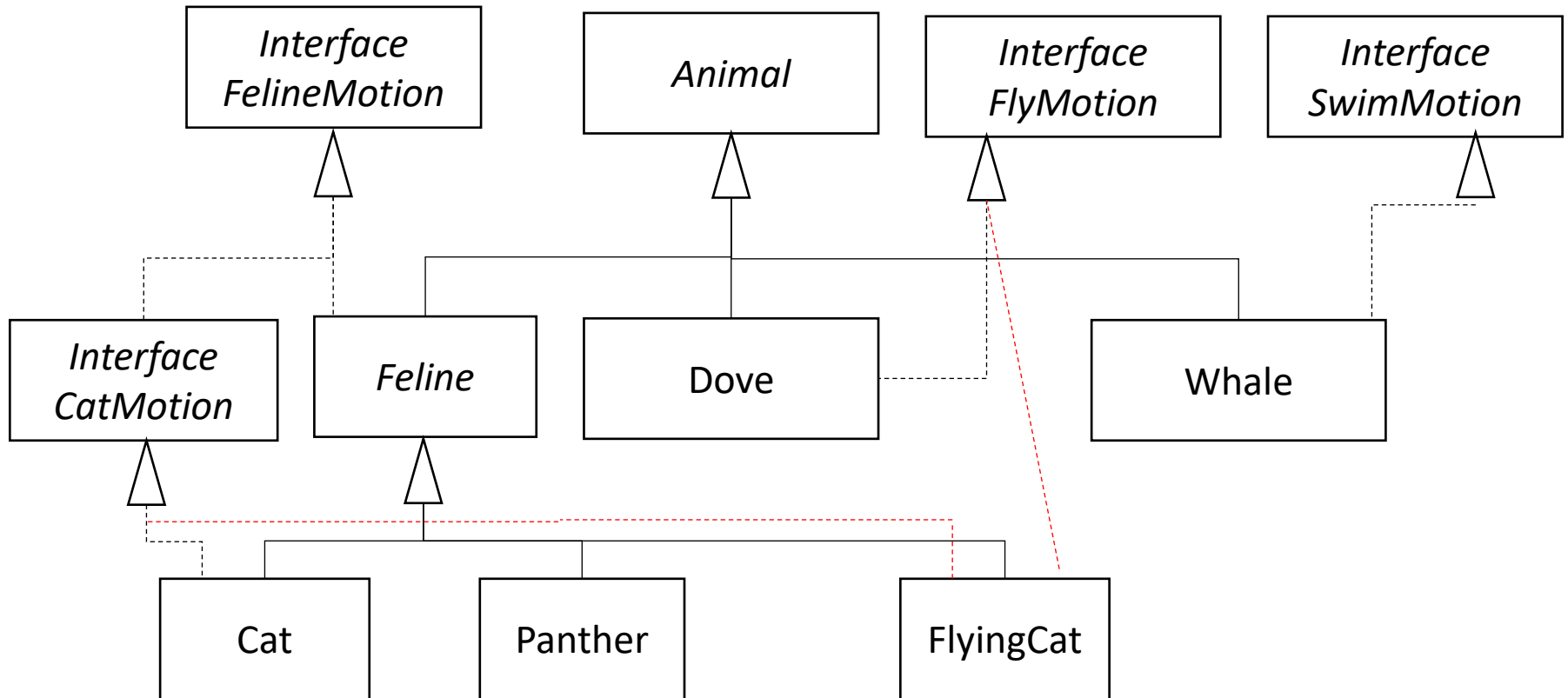


```
interface CatMotion extends FelineMotion {  
    public void tap(Animal animal) ;  
}
```

Questions?

- Extending interface
- Examples

Flying Cat



Implementing Multiple Interfaces

- A class **can** implement multiple interfaces
- But a class **cannot** extend multiple classes
- Which one of the following are is allowed in Java?

```
class FlyingCat extends  
Cat, Dove {  
  
...  
  
}
```

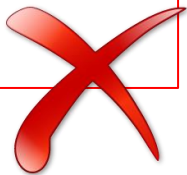
```
class FlyingCat implements  
FlyMotion, CatMotion {  
  
...  
  
}
```

```
class FlyingCat extends  
Feline implements  
FlyMotion, CatMotion {  
  
...  
  
}
```


Implementing Multiple Interfaces

- A class **can** implement multiple interfaces
- But a class **cannot** extend multiple classes

```
class FlyingCat extends  
Cat, Dove {  
...  
}
```



```
class FlyingCat implements  
FlyMotion, CatMotion {  
...  
}
```



```
class FlyingCat extends  
Feline implements  
FlyMotion, CatMotion {  
...  
}
```



Questions

- Interfaces
 - Model common behaviors
 - Have only abstract methods
 - Since Java 8, can have default methods and static methods (virtual/abstract functions/methods with default implementations)
 - Can be extended
 - Must be implemented
- Assemble (or aggregate) behaviors
 - Examples

Using Interface as Type

- Interfaces are data types

```
void flyAll(ArrayList<BirdMotion> flyingAnimals) {
```

```
...
```

```
}
```

```
Void moveFlyable(FlyMotion flyable) {
```

```
...
```

```
}
```

Questions

- Interfaces are data types
- Write generic method with interface
- Examples