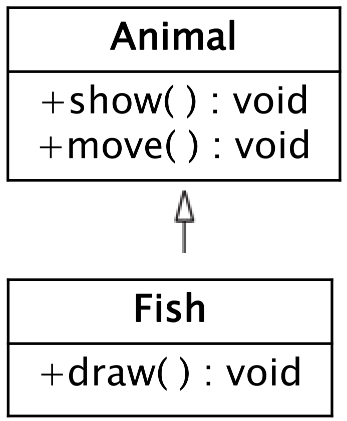
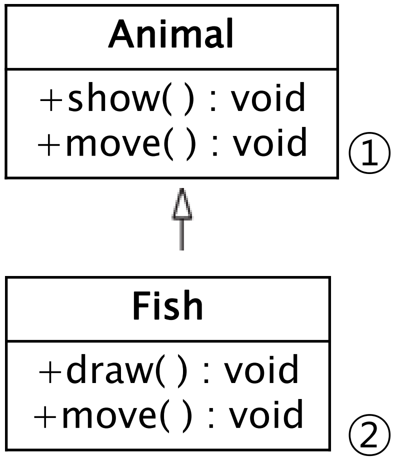
Practice on Polymorphism
- step by step -

1



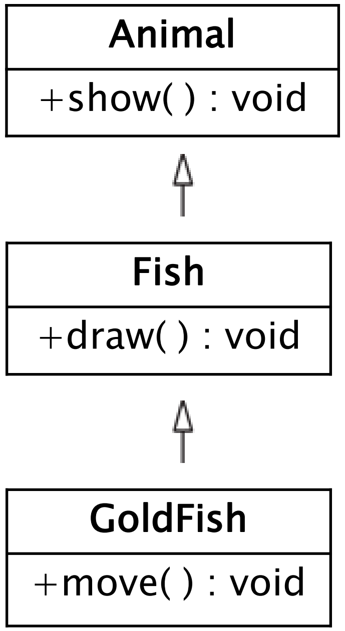
```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new Fish();
        fish.move();
    }
}
```

2



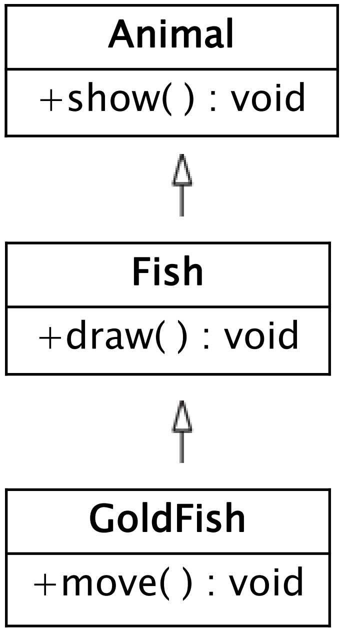
```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new Fish();
        fish.move();
    }
}
```

3



```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new Fish();
        fish.move();    // error
    }
}
```

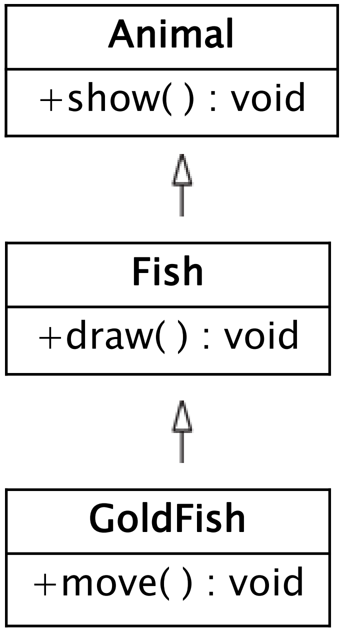
4



```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new Fish();
        ((GoldFish)fish).move();  // downcasting
                                  // ClassCastException
    }
}
```

```
Exception in thread "main" java.lang.ClassCastException: Fish cannot be cast to GoldFish
	at AnimalTest.main(AnimalTest.java:8)
```

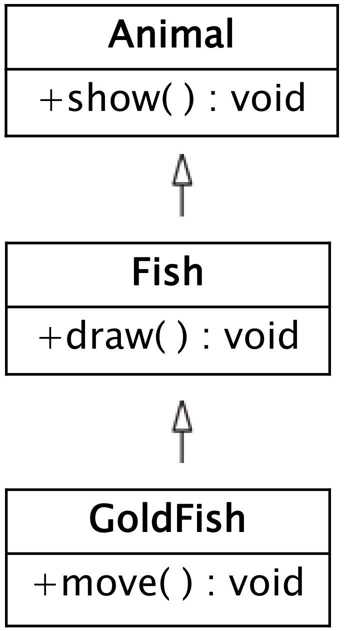
5



```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new GoldFish(); // upcasting
        ((GoldFish)fish).move();    // downcasting
    }
}
```

upcasting 된 것을 downcasting 하라!

6



```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new GoldFish();  // upcasting

        if ( fish instanceof GoldFish )
            ((GoldFish)fish).move(); // downcasting
    }
}
```

instanceof 연산자를 사용하여 적절한 upcasting 인지 확인한 후 downcasting 하라!
(즉, fish가 참조하는 실제 객체가 GoldFish 형의 객체인지 확인할 필요가 있음)

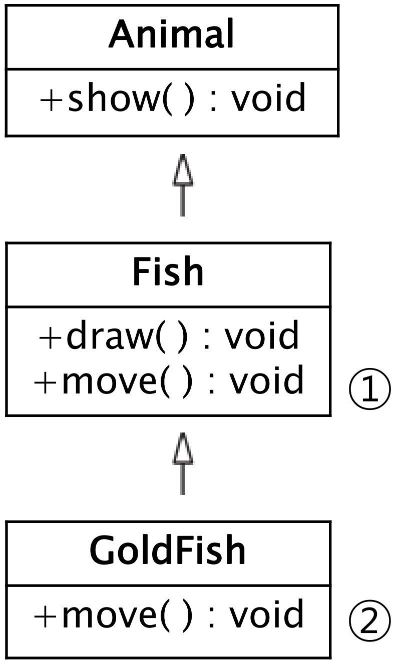
Upcasting and downcasting

- Casting a subclass type to a superclass type(upcasting)
  - A superclass reference to a subclass object
  - *Basically*,
    - you can use just the superclass type's members.

  - *For the overridden methods*, however,
    - the called method is determined *at execution time* polymorphically.
    - *Subclass's overridden method is called*.

- Casting a superclass type to a subclass type (downcasting)
  - needs *explicit type casting*.
  - enables a program to invoke subclass methods that are not in the superclass.

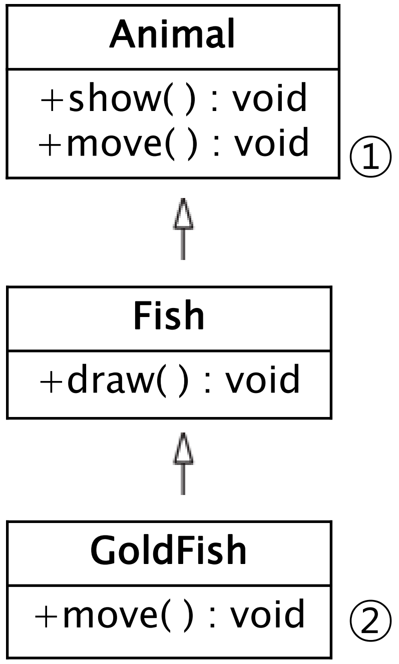
7.



overridden
methods

```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new GoldFish();  // upcasting
        fish.move();                 // ②
    }
}
```

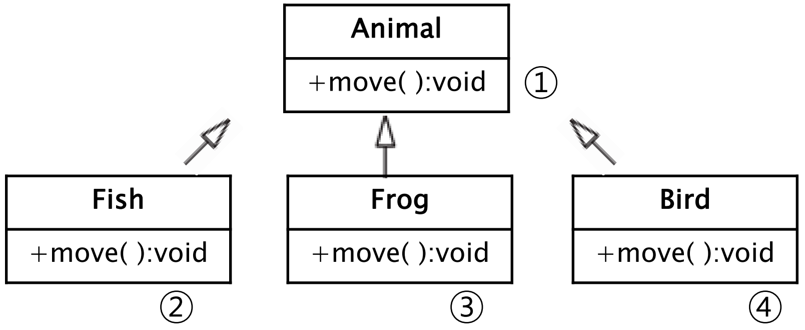
8.



overridden
methods

```
public class AnimalTest
{
    public static void main(String[] args)
    {
        Fish fish = new GoldFish();   // upcasting
        fish.move();                  // ②
    }
}
```

9. polymorphism

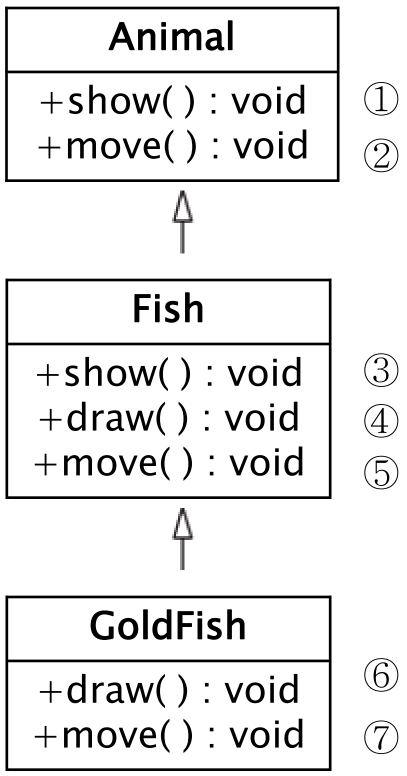


method overriding

```
public class AnimalTest
{
    public static void main(String[] args)
    {
        // each array element is a reference to Animal
        Animal[] animal = new Animal[ 3 ];
        animal[ 0 ] = new Fish();   // upcasting from Fish to Animal
        animal[ 1 ] = new Frog();   // upcasting from Frog to Animal
        animal[ 2 ] = new Bird();   // upcasting from Bird to Animal

        for ( int i = 0; i < animal.length ; i++ )
            animal[i].move();    // polymorphic behavior
    }                            // dynamic binding or late binding
}
```

10.



```
public class AnimalTest
{
    public static void main(String[] args)
    {
        // case 1.
        GoldFish goldFish = new GoldFish();
        goldFish.show();

        // case 2.
        Animal animal = new Fish();
        animal.draw();    // error

        // case 3.
        Animal animal2 = new Fish();
        animal2.show();
        animal2.move();

        // case 4.
        Animal animal3 = new GoldFish();
        ((Fish)animal3).draw();
    }
}
```