Chapter 17-1

인터페이스

상수, 추상메서드만 가지고 있는 인터페이스

interface_/ISomething.java

```
1:  package interface_;
2:
3:  public interface ISomething {
4:      public static final int A = 11;
5:      int My_INT = 22;
6:      void run();
7:  }
```

인터페이스는 클래스가 아니기 때문에
일반 변수, 일반 메서드를 사용 할 수 없다!

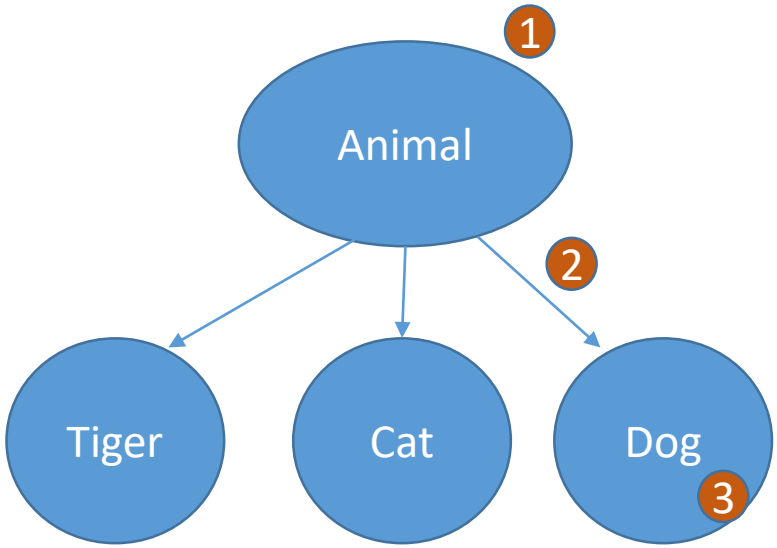
```
//자동으로 public static final이 됩니다.
//public abstract void run();과 동일합니다.
```

- 인터페이스 선언된 변수는 **public static final**을 생략하더라도 컴파일 과정에서 자동으로 붙게 됩니다. (**상수**)
- 인터페이스의 메서드를 추상메서드 형식으로 선언하면 **abstract**를 붙이지 않더라도 자동으로 컴파일 과정에서 붙게 됩니다. (**추상메서드**)
- 추가적으로 static메서드의 선언 또한 가능 합니다. (자바 1.8버전 이후)

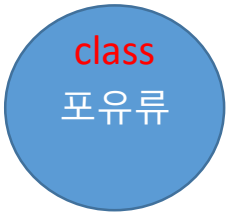
인터페이스 왜 생겼을까?

클래스는 다중 상속을 지원하지 않는다!

부모클래스를 추상클래스로 정의



만약 Dog에게 포유류만의 기능을 추가 하고 싶다면?
어디로 넣어줘야 할까?



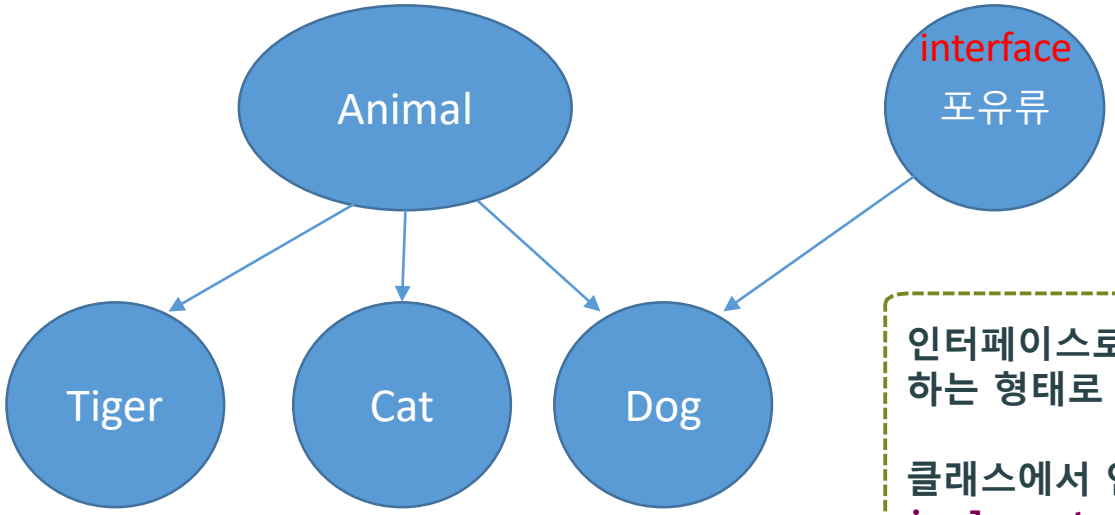
문제점

1. Animal 하위의 모든 클래스가 포유류의 기능을 갖게 된다
2. 포유류는 Animal의 기능을 갖게 된다.
3. 재활용성이 떨어진다(단순히 클래스에 추가하는 격)

인터페이스 왜 생겼을까?

부모클래스를 추상클래스로 정의

자 그렇다면 포유류를 interface구조로 정의 하고
추상메서드로 그 기능만 정의해 놓을게!



인터페이스로 정의하고 포유류의 기능을 구현(상속)
하는 형태로 사용가능하다.

클래스에서 인터페이스를 구현할 때는 클래스 이름 뒤에
implements 키워드를 사용합니다.

```
class Dog extends Animal implements 포유류 {...}
```

인터페이스 기본 예제

클래스에서 인터페이스를 구현할 때는 클래스 이름 뒤에 **implements** 키워드를 사용합니다

interface_/Shape.java

```
1:  package interface_;
2:
3:  public interface Shape{
4:     public abstract double getArea();
5:  }
```

interface_/Rectangle.java

```
1:  package interface_;
2:
3:  public class Rectangle implements Shape {
4:     int width;
5:     int height;
6:     public Rectangle(int width, int height) {
7:        this.width = width;
8:        this.height = height;
9:     }
10:
11:    public double getArea() {
12:       return    width * height;
13:    }
14: }
```

인터페이스의 추상 메서드는 반드시 오버라이딩 되어야 한다!

interface_/Triangle.java

```
1:  package interface_;
2:
3:  public class Triangle implements Shape {
4:     int width;
5:     int height;
6:
7:     public Triangle(int width, int height) {
8:        this.width = width;
9:        this.height = height;
10:    }
11:
12:    public double getArea() {
13:       return    width * height / 2;
14:    }
15: }
```

interface_/InterfaceExample.java

```
1:  package interface_;
2:
3:  public class InterfaceExample {
4:     public static void main(String[] args) {
5:        Rectangle rect = new Rectangle(20,34);
6:        Triangle tri = new Triangle(20,34);
7:
8:        System.out.println("rect's Area = " + rect.getArea());
9:        System.out.println("tri's Area = " + tri.getArea());
10:    }
11: }
```

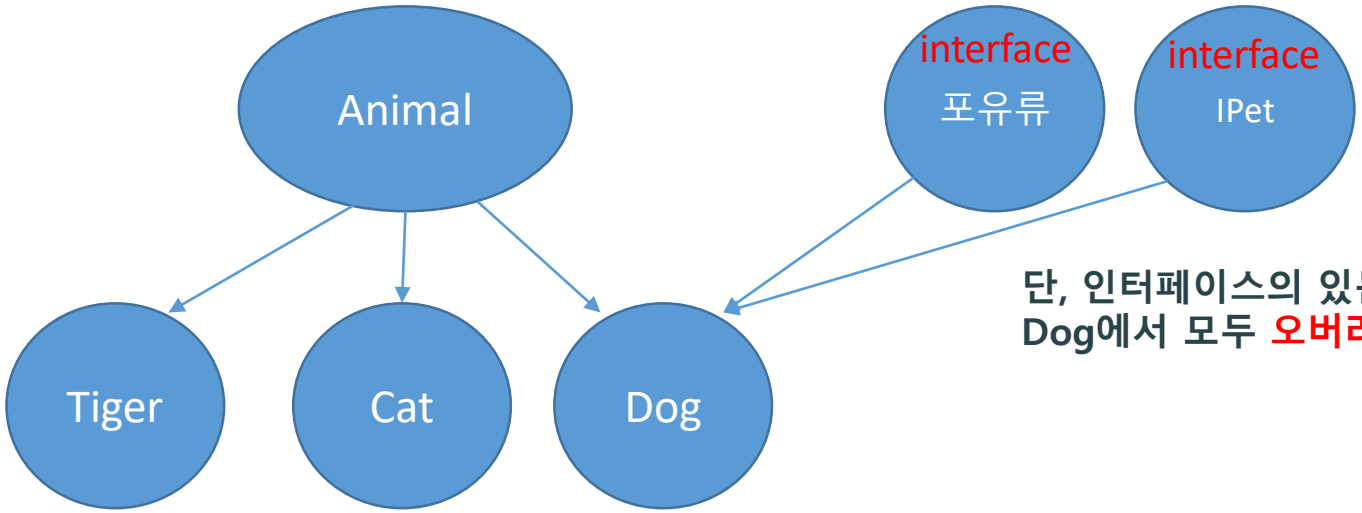
Console

<terminated> InterfaceExample [Ja

```
rect's Area = 680.0
tri's Area = 340.0
```

인터페이스의 기능1 (다중 상속을 지원한다)

- 인터페이스는 다중 상속도 표현할 수 있습니다.
- 여러 인터페이스를 동시 구현 할 수 있습니다



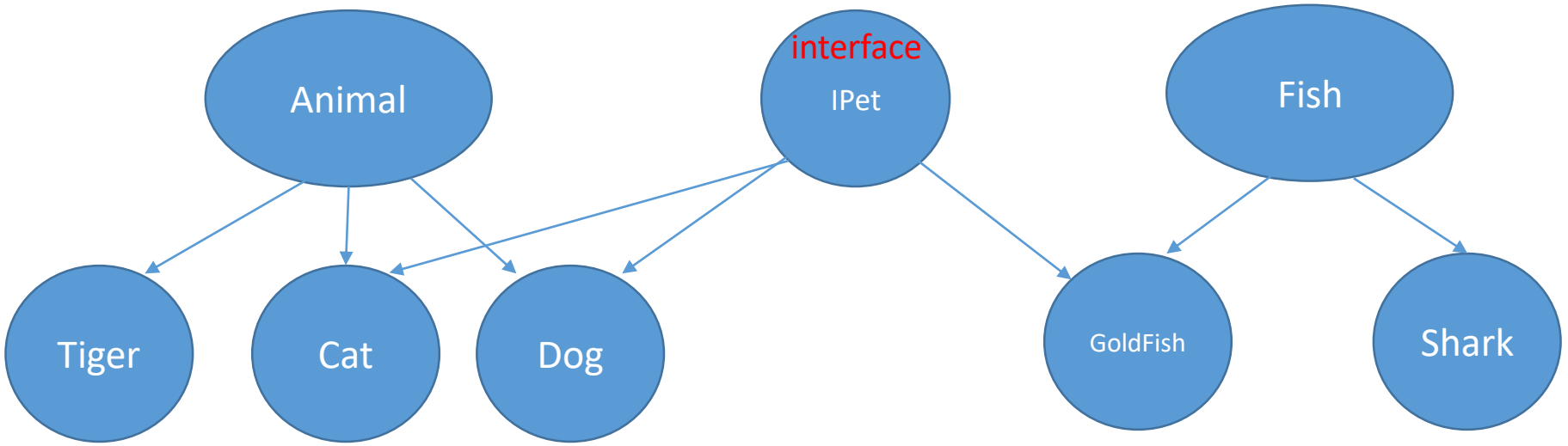
단, 인터페이스의 있는 **모든 추상 메서드**는
Dog에서 모두 **오버라이딩** 되야 한다

```
class Dog extends Animal implements 포유류, IPet {...}
```

둘 이상의 인터페이스 구현 가능! (,)로 연결

인터페이스 기능2 (메서드 명세서)

- 자바의 인터페이스는 객체의 사용 방법을 정의한 타입(메서드 명세서)으로 객체의 교환성을 높여주기 때문에 다형성을 구현하는 매우 중요한 역할을 합니다.



Dog클래스는 3가지 형태가 될 수 있다

```
Animal a = new Dog();
IPet I = new Dog();
Dog d = new Dog();
```

IPet을 부모타입으로 비슷한 기능을 묶어서 사용할 수 있다

```
IPet pet1 = new Dog();
IPet pet2 = new Cat();
IPet pet3 = new GoldFish();
```

인터페이스도 데이터 타입(부모타입)이 될 수 있다!!

인터페이스 기능3

- 사용방법이 동일한 클래스를 만드는 기술입니다.
- 프로그램에서 실제 사용되는 방법입니다.

마이크로소프트에서 정의한 소프트웨어 사용법

```
public interface Printed {

	public void print(String document);
	public void colorPrint(String document, String color);
	public int copy(int n);
}
```

인터페이스를 사용하면
각자의 나름대로 형식에 맞추어 사용방법을
정의 하는게 가능하다!!

LG 프로그램 개발자

```
public class LG implements Printed{

@Override
public void print(String document) {…}
@Override
public void colorPrint(String document, String color) {…}
@Override
public int copy(int n) {…}
```

삼성 프로그램 개발자

```
public class Samsung implements Printed{

@Override
public void print(String document) {…}
@Override
public void colorPrint(String document, String color) {…}
@Override
public int copy(int n) {…}
```

다중 상속 예제

interface_/IToDo1.java

```
1:  package interface_;
2:
3:  public interface IToDo1 {
4:     void m1();
5:  }
```

interface_/IToDo2.java

```
1:  package interface_;
2:
3:  public interface IToDo2 {
4:     void m2();
5:  }
```

interface_/IToDo3.java

```
1:  package interface_;
2:
3:  public interface IToDo3 extends IToDo1, IToDo2 {
4:     void m3();
5:  }
```

interface_/IToDo4.java

```
1:  package interface_;
2:
3:  public interface IToDo4 {
4:     void m4();
5:  }
```

interface간에도 상속이 가능하다
interface간에 상속은 **extends**를 이용한다

interface_/ToDo.java

```
1:  package interface_;
2:
3:  public class ToDo implements IToDo3, IToDo4 {
4:
5:     public void m1() {
6:        System.out.println("m1() 구현");
7:     }
8:
9:     public void m2() {
10:       System.out.println("m2() 구현");
11:    }
12:
13:    public void m3() {
14:       System.out.println("m3() 구현");
15:    }
16:
17:    public void m4() {
18:       System.out.println("m4() 구현");
19:    }
20: }
```

인터페이스 정리

- 인터페이스는 상수와 추상메서드 만을 구성멤버로 가집니다.

1. 인터페이스는 기본적으로 다중상속을 지원합니다.
2. 자바의 인터페이스는 객체의 사용 방법을 정의한 타입(메서드 명세서)으로 다형성을 구현하는 매우 중요한 역할을 합니다. 인터페이스도 데이터 타입(부모타입)이 될 수 있다!
3. 사용방법이 동일한 클래스를 만드는 기술입니다.

- 인터페이스의 구현 키워드는 `implements`
- 인터페이스도 `extends` 키워드를 사용하여 인터페이스 간의 상속을 구현할 수 있습니다.

인터페이스는 추상클래스의 확장된 형태 인가요?

- 네. 클래스가 가지는 한계점을 보완하면서 발전된 형태라고 생각해도 무방합니다.

부모클래스의 변화

일반 클래스 ➡ 오버라이딩 강제(추상클래스) ➡ 오버라이딩 강제 + 다중 상속(인터페이스)

Chapter 17

수고하셨습니다