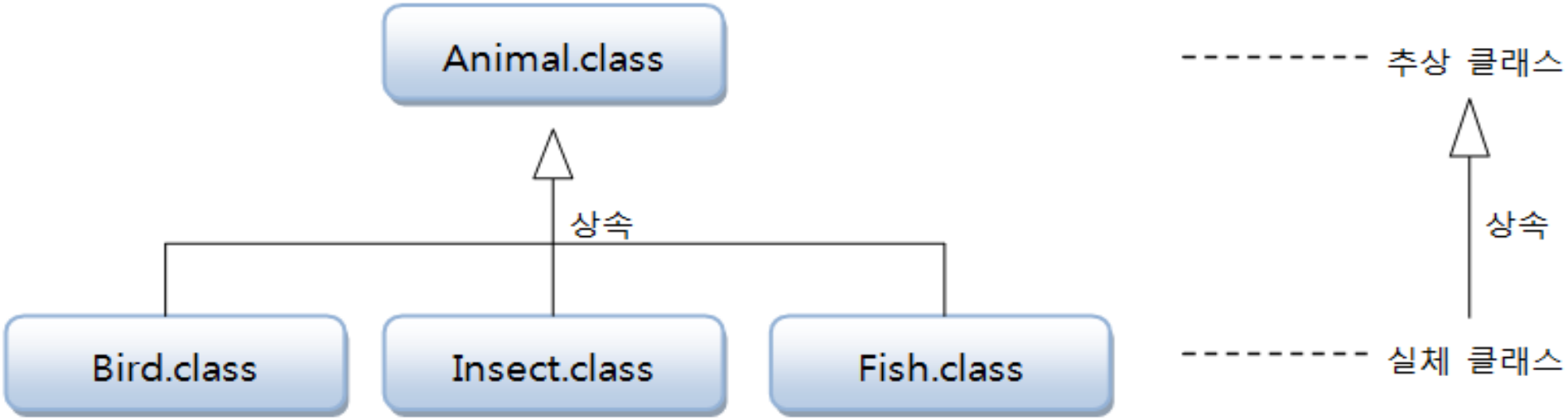
추상 클래스

추상 클래스(Abstract Class)

❖ 추상 클래스 개념

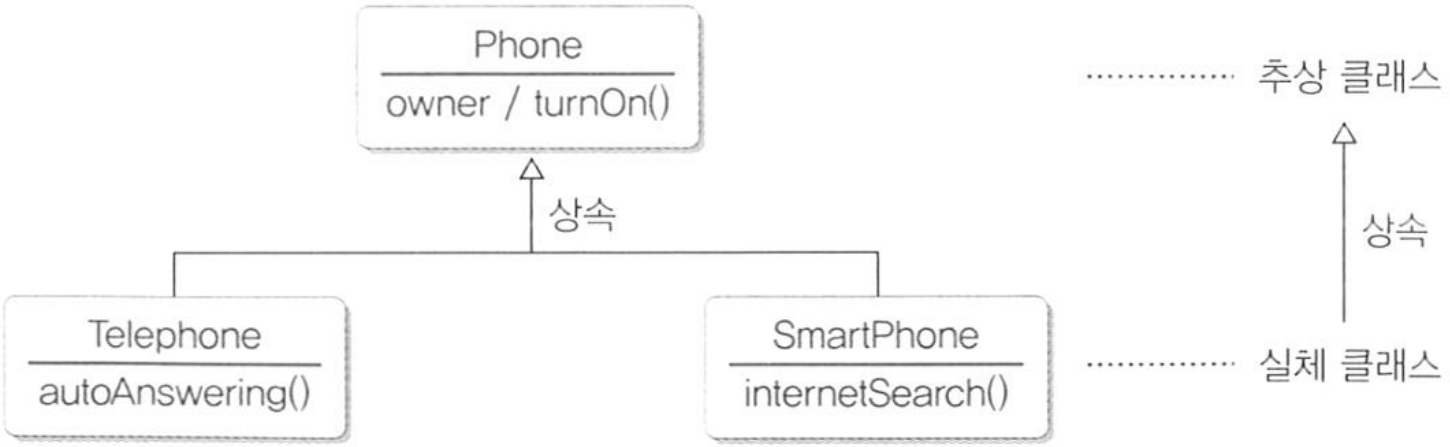
- 추상(abstract)
  - 실체들 간에 공통되는 특성을 추출한 것
    - 예1: 새, 곤충, 물고기 → 동물 (추상)
    - 예2: 삼성, 현대, LG → 회사 (추상)
- 추상 클래스(abstract class)
  - 실체 클래스들의 공통되는 필드와 메소드 정의한 클래스
  - 추상 클래스는 실체 클래스의 부모 클래스 역할 (단독 객체 X)



추상 클래스(Abstract Class)

❖ 추상 클래스의 용도

- ㅇ 실체 클래스의 공통된 필드와 메소드의 이름 통일할 목적
  - 실체 클래스를 설계자가 여러 사람일 경우,
  - 실체 클래스마다 필드와 메소드가 제각기 다른 이름을 가질 수 있음
- ㅇ 실체 클래스를 작성할 때 시간 절약
  - 실체 클래스는 추가적인 필드와 메소드만 선언



- ㅇ 실체 클래스 설계 규격을 만들고자 할 때
  - 실체 클래스가 가져야 할 필드와 메소드를 추상 클래스에 미리 정의
  - 실체 클래스는 추상 클래스를 무조건 상속 받아 작성

추상 클래스(Abstract Class)

❖ 추상 클래스 선언

- ㅇ 클래스 선언에 abstract 키워드 사용
  - New 연산자로 객체 생성하지 못하고 상속 통해 자식 클래스만 생성 가능

```
public abstract class 클래스 {
   //필드
   //생성자
   //메소드
}
```

추상 클래스(Abstract Class)

❖ 추상 클래스 : Phone.java

```
public abstract class Phone {
    // 필드
    public String owner;

    // 생성자
    public Phone(String owner) {
        this.owner = owner;
    }

    // 메소드
    public void turnOn() {
        System.out.println("폰 전원을 켭니다.");
    }

    public void turnOff() {
        System.out.println("폰 전원을 끕니다.");
    }
}
```

추상 클래스(Abstract Class)

❖ 실제 클래스 : SmartPhone.java

```
public class SmartPhone extends Phone {
    // 생성자
    public SmartPhone(String owner) {
        super(owner);
    }

    // 메소드
    public void internetSearch() {
        System.out.println("인터넷 검색을 합니다.");
    }
}
```

추상 클래스(Abstract Class)

❖ 추상 클래스 : PhoneExample.java

```
public class PhoneExample {
    public static void main(String[] args) {
        // Phone phone = new Phone(); (x)

        SmartPhone smartPhone = new SmartPhone("홍길동");

        smartPhone.turnOn();       // Phone의 메서드
        smartPhone.internetSearch();
        smartPhone.turnOff();      // Phone의 메서드

        Phone phone = new SmartPhone("홍길동");

        phone.turnOn();     // Phone의 메서드
        ((SmartPhone)phone).internetSearch();
        phone.turnOff();    // Phone의 메서드


    }
}
```

추상 클래스(Abstract Class)

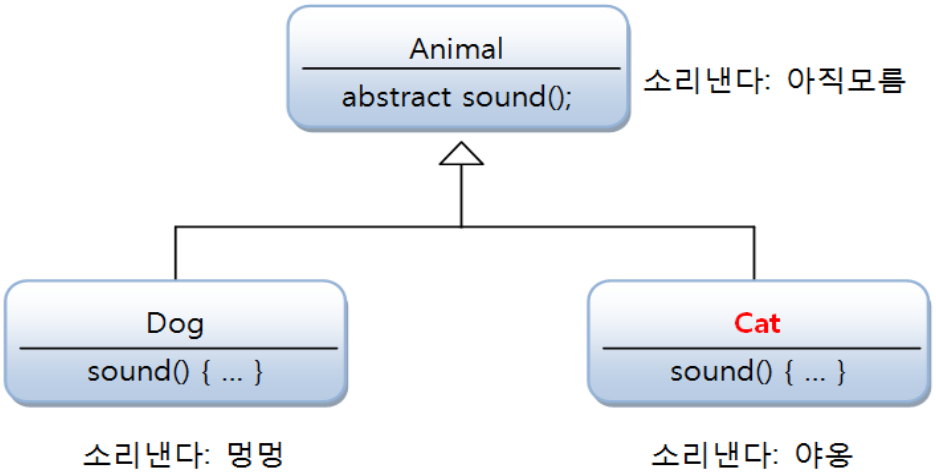
❖ 추상 메소드와 오버라이딩(재정의)

- 메소드 이름 동일하지만, 실행 내용이 실체 클래스마다 다른 메소드
  - 예: 동물은 소리를 낸다. 하지만 실체 동물들의 소리는 제각기 다르다.

- 구현 방법
  - 추상 클래스에는 메소드의 선언부만 작성 (추상 메소드)
  - 실체 클래스에서 메소드의 실행 내용 작성(오버라이딩(Overriding))

추상 클래스(Abstract Class)

❖ 추상 메소드와 오버라이딩(재정의)




```
public abstract class Animal {
    public abstract void sound();
}
```

```
public class Dog extends Animal {
    @Override
    public void sound() {
        System.out.println("멍멍");
    }
}
```

```
public class Cat extends Animal {
    @Override
    public void sound() {
        System.out.println("야옹");
    }
}
```

추상 클래스(Abstract Class)

❖ 추상 메서드 선언 : Animal.java

```
public abstract class Animal {
    public String kind;    // 추상 클래스

    public void breathe() {
        System.out.println("숨을 쉽니다.");
    }

    public abstract void sound();    // 추상 메서드
}
```

추상 클래스(Abstract Class)

❖ 추상 메서드 오버라이딩: Cat.java

```
Public class Cat extends Animal {
    public Cat() {
        this.kind = “포유류”;
    }

    @Override
    public void sound() {  // 추상 메서드 재정의
        System.out.println("야옹");
    }
}
```

❖ 추상 메서드 오버라이딩: Dog.java

```
public class Dog extends Animal {
    public Dog() {
        this.kind = "포유류";
    }

    @Override
    public void sound() {  // 추상 메서드 재정의
        System.out.println("멍멍");
    }
}
```

추상 클래스(Abstract Class)

❖ 실행 클래스: AnimalExample.java

```
public class AnimalExample {
    public static void main(String[] args) {
        Dog dog = new Dog();
        Cat cat = new Cat();
        dog.sound();
        cat.sound();
        System.out.println("-----");

        // 변수의 자동 타입 변환
        Animal animal = null;
        animal = new Dog();
        animal.sound();
        animal = new Cat();
        animal.sound();
        System.out.println("-----");

        // 매개변수의 자동 타입 변환
        animalSound(new Dog());
        animalSound(new Cat());
    }

    public static void animalSound(Animal animal) {
        animal.sound();
    }
}
```