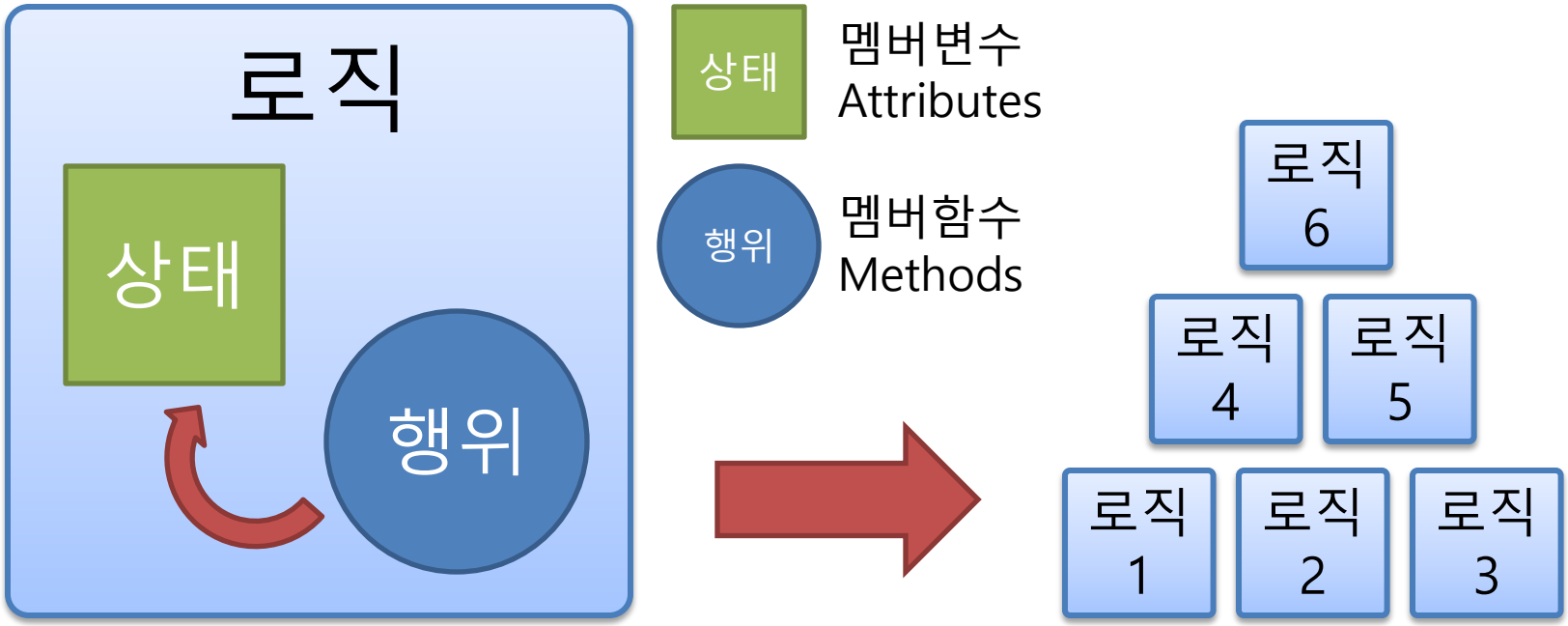
NCS - Java 007 abstract, interface

1. 객체지향 프로그래밍 다시보기

- 객체지향 프로그래밍은 로직을 상태(state)와 행위(behave)로 이루어진 객체로 만든 것이다. 그리고 이 객체들을 레고 블록처럼 조립해서 하나의 프로그램으로 만드는 것이다



행위는 상태를 변화시킨다

2. OOP 설계 5대원칙 S.O.L.I.D.

- 객체지향 프로그래밍은 로직을 상태(state)와 행위(behave)로 이루어진 객체로 만든 것이다. 그리고 이 객체들을 레고 블록처럼 조립해서 하나의 프로그램으로 만드는 것이다

SRP	Single Responsibility Principle 단일 책임의 원칙
OCP	Open Closed Principle 개방-폐쇄 원칙
LSP	Liskov Substitution Principle 리스코프 교체 원칙
ISP	Interface Segregation Principle 인터페이스 격리 원칙
DIP	Dependency Inversion Principle 의존 관계 역전 원칙

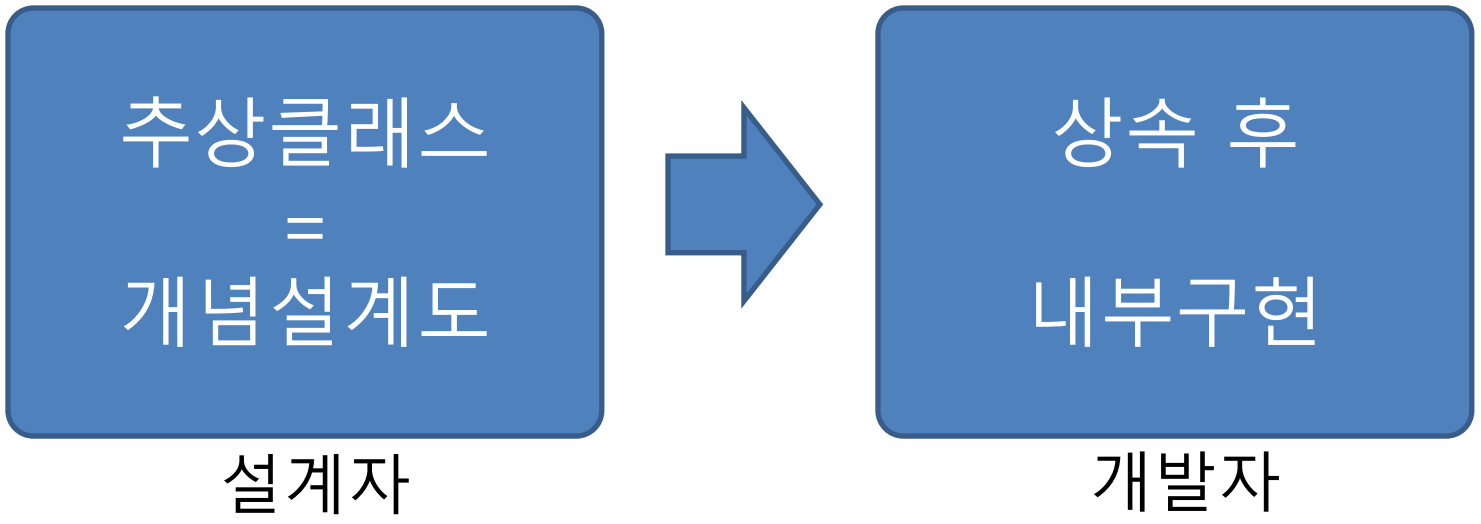
3. DIP - 의존 관계 역전으로 보는 추상화

- 클라이언트는 클래스가 아닌 추상화(인터페이스, 추상 클래스) 레이어에 의존해야 한다는 원칙으로, 확장 이슈가 있는 부분은 추상화를 해야 된다는 내용이다.

abstract

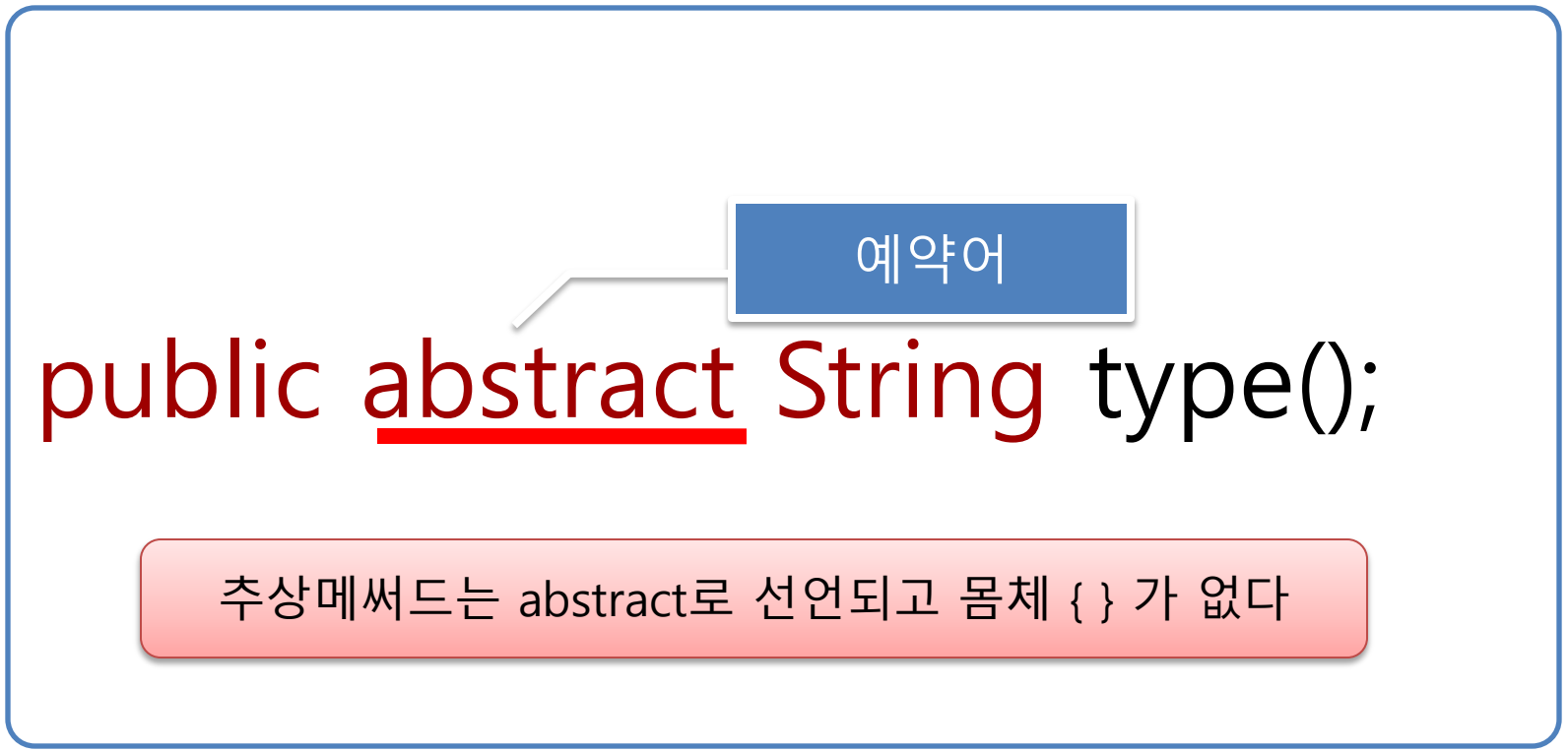
인터페이스 역할을 하면서 구현체도 가지고 있는 클래스. 일종의 개념 설계도와 같은 역할을한다

- 포괄적 의미의 객체를 지칭할 때 사용한다.
  예) 동물 > 종류에 따라서 구체적인 이동방식, 사는곳 등이 다르다



4. abstract method 추상함수 만들기

- 추상함수(abstract method)는 abstract 예약어로 선언되고 몸체에 해당하는 { } 블럭이 없다



5. abstract class 추상 클래스 만들기

- 추상 클래스(abstract class)는 abstract 예약어로 선언되고 하나 이상의 추상함수를 갖는다

예약어

```
abstract class Animal {

    public abstract String type();

}
```

추상클래스는 abstract로 선언되고 추상메써드를 갖는다

6. abstract 추상 클래스 구현

- 선언된 추상 클래스는 상속을 통해 자식 클래스에서 구현할 수 있다

```
abstract class Animal {
    public abstract String type();
}

class Dog extends Animal {
    String type = "포유류";
    public String type() { return type; }
}

class Bird extends Animal {
    String type = "조류";
    public String type() { return type; }
}
```

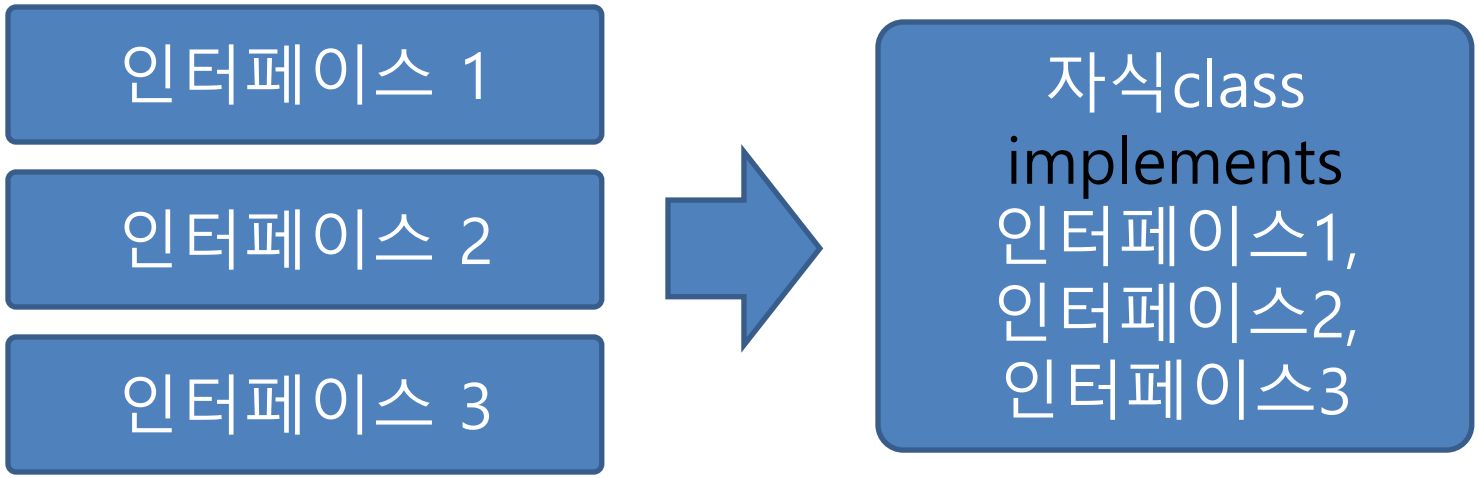

7. ISP - 인터페이스 분리의 법칙

- 하나의 범용인터페이스 보다 특화된 여러개의 인터페이스가 낫다

interface

몸체가 없는 추상함수(abstract method)들로 이루어져 있으며 interface를 상속받은 자식 객체는 반드시 메소드들을 구현해야만 한다

- 자바에서 상속은 한개만 가능하지만 인터페이스는 컴마로 구분해서 하나 이상을 구현 할 수 있다



8. interface method 인터페이스함수 만들기

- 인터페이스에서 사용되는 함수는 몸체가 없는 추상함수(abstract method) 이지만 abstract 예약어를 생략한다

9. interface class 인터페이스 클래스 만들기

- 추상 클래스(abstract class)는 abstract 예약어로 선언되고 하나 이상의 추상함수를 갖는다.
- **몸체가 있는 일반 함수를 만들 수 없다**
- **class 예약어를 사용하지 않는다**

```
interface City {

    public String name();

}
```

예약어

인터페이스 클래스는 interface로 선언되고 추상함수를 갖는다

10. interface class 인터페이스 클래스 구현

- 선언된 인터페이스 클래스는 구현(implements)을 통해 자식 클래스에서 구현할 수 있고, **자식 클래스는 반드시 구현**해야 한다

```
interface City {
    public String name();
}

class seoul implements City {
    String name = "서울";
    public String name() { return name; }
}

class busan implements City {
    String name = "부산";
    public String name() { return name; }
}
```

