객체지향 프로그래밍

Object-oriented Programming

Chapter 07.
인터페이스와 특수 클래스

- 추상 클래스
- 인터페이스 기본
- 인터페이스 응용
- 인터페이스와 다형성
- 중첩 클래스와 중첩 인터페이스
- 익명 클래스

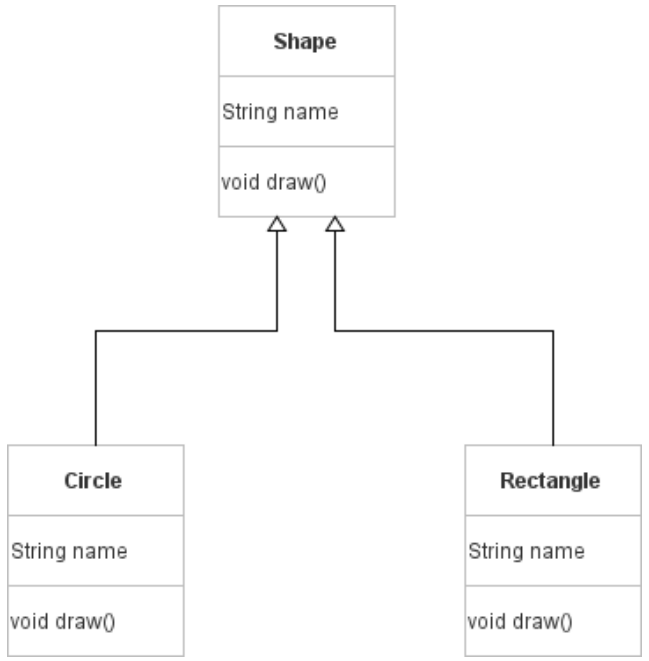
추상 클래스

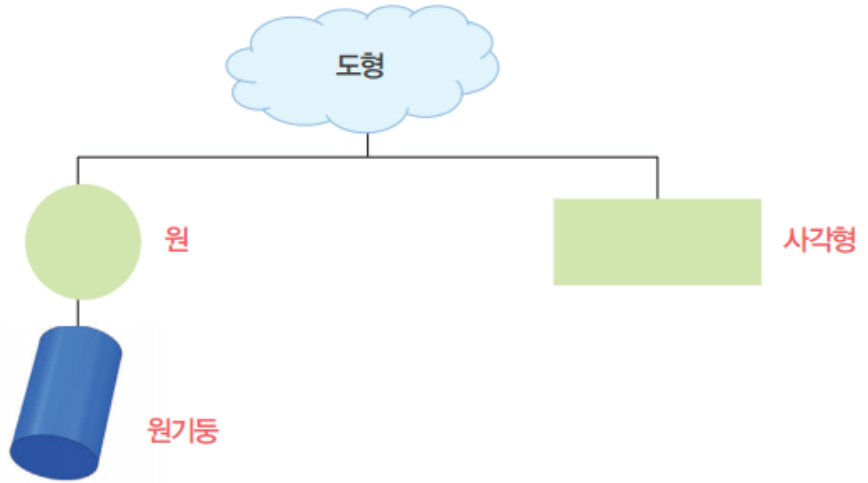
추상 클래스

- **추상 클래스**: 추상 메서드를 가질 수 있는 클래스

- **추상 메서드**: 메서드 본체를 완성하지 못한 메서드.
  - 무엇을 할지는 선언할 수 있지만, 어떻게 할지는 정의할 수 없음.
  - 추상 클래스를 상속 받는 클래스에서 추상 메서드 몸체를 정의.

Shape 클래스의 draw()?





추상 클래스

추상 클래스

- 보통 하나 이상의 추상 메서드를 포함하지만 없을 수도 있음

- 주로 상속 계층에서 자식 멤버의 이름을 통일하기 위하여 사용

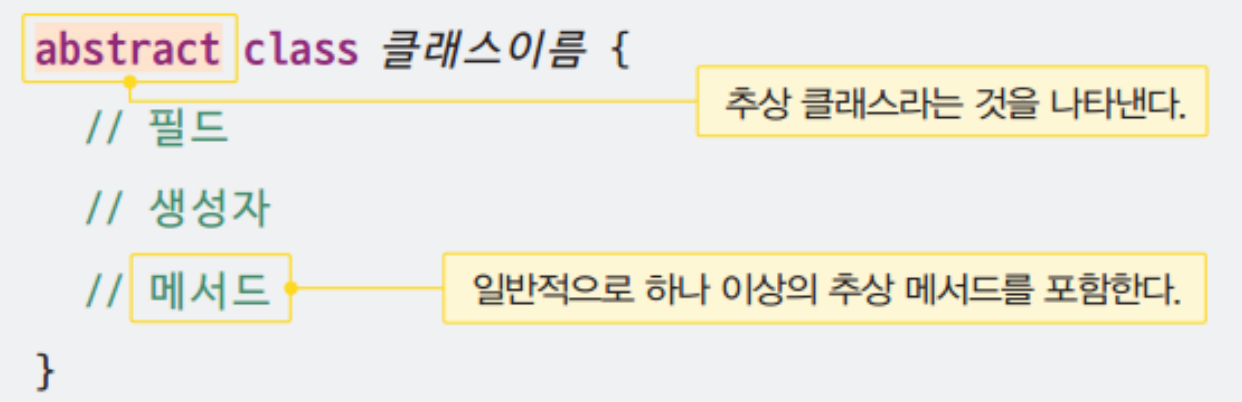
- 몸체가 정의되지 않은 추상 메서드를 포함하므로 추상 클래스의 인스턴스를 바로 생성 불가.

```
추상클래스 s = new 추상클래스();   // 추상 클래스는 인스턴스를 생성하지 못한다.
```

추상 클래스

추상 클래스의 선언

- 클래스 선언부에 abstract 키워드를 명시



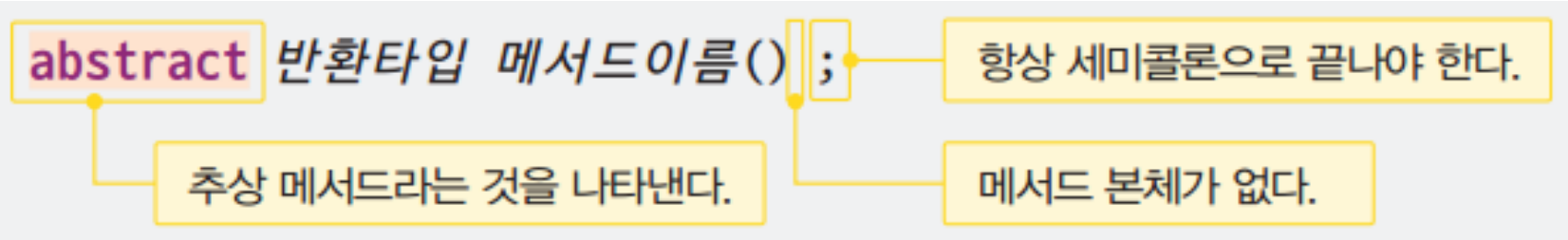

```
abstract class Shape {

}
```

추상 클래스

추상 메서드의 선언

- 메서드 선언부에 abstract 키워드를 명시




```
abstract class Shape {
    double pi = 3.14;

    abstract void draw();

    public double findArea() {
        return 0.0;
    }
}
```

Practice

추상 클래스 실습

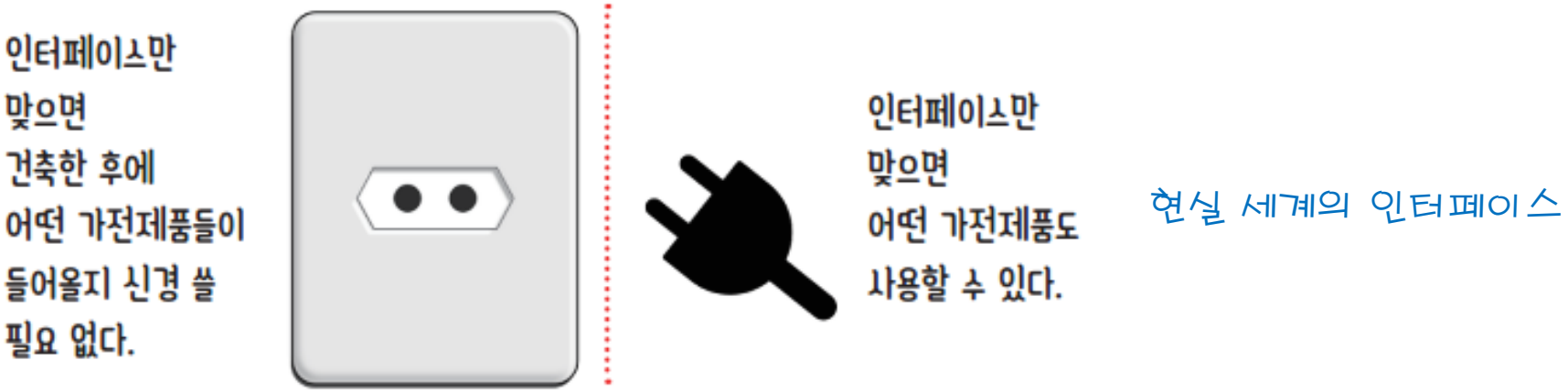
기초적 형태의 추상 클래스 선언 및 활용

- 예제 7-1 (p. 268), 7-2 (p. 268), 7-3 (p. 269)

인터페이스 기본

인터페이스

- 두 시스템, 장치, 소프트웨어 간에 정보를 주고받는 접점이나 경계면



현실 세계의 인터페이스

인터페이스 기본

자바의 인터페이스

- **인터페이스**: 클래스들이 구현해야 하는 동작을 지정하는데 사용되는 추상 자료형.

- 추상 클래스와 마찬가지로, 인터페이스에 여러 개의 **추상 메서드**를 선언 가능.
  하지만, 추상 메서드만 선언 가능.
  - 멤버 필드 선언 불가.
  - static 메서드, 필드는 선언 가능.

인터페이스 기본

인터페이스 선언

- 클래스 선언과 유사
- class 대신 interface 키워드를 사용

```
public interface Predator {

}
```

인터페이스 기본

인터페이스의 추상 메서드 선언

- Interface의 모든 추상 메서드는 public 메서드
- abstract, public 키워드를 생략하여 선언 가능

```
public interface Predator {
        boolean chasePrey(Prey p);
        void eatPrey(Prey p);
}
```

인터페이스 기본

인터페이스의 구현

- 추상 클래스는 **상속**의 대상이지만, 인터페이스는 **구현**의 대상
- 클래스는 **implements** 키워드를 통해 인터페이스를 구현함을 명시

```java
public class Lion implements Predator {

    @Override
    public boolean chasePrey(Prey p) {
        System.out.println("사자가 " + p + "을(를) 쫓는다!");
        return true;
    }
}
```

인터페이스 기본

디폴트 메서드

- interface의 추상 메서드 중 동작이 미리 정의된 메서드
- 추상 메서드 선언부에 **default** 키워드를 명시
- 디폴트 메서드 또한 interface의 구현 클래스에서 오버라이딩 가능

```
public interface Predator {
    boolean chasePrey(Prey p);

    default void eatPrey(Prey p) {
        System.out.println(p.toString() + "을 먹었다!!!")
    }
}
```

인터페이스 기본

인터페이스의 장점

- 인터페이스만 준수하면 통합에 신경 쓰지 않고 다양한 형태로 새로운 클래스를 개발할 수 있다.

- 클래스의 다중 상속을 지원하지 않지만, 인터페이스로 **다중 상속 효과**를 간접적으로 얻을 수 있다.
  - 하나의 클래스는 여러 종류의 인터페이스를 구현할 수 있다.

인터페이스 연관 개념 (참고):

- System Architecture
- Clean Architecture
- Microservices Architecture
- Dependency Injection

인터페이스 기본

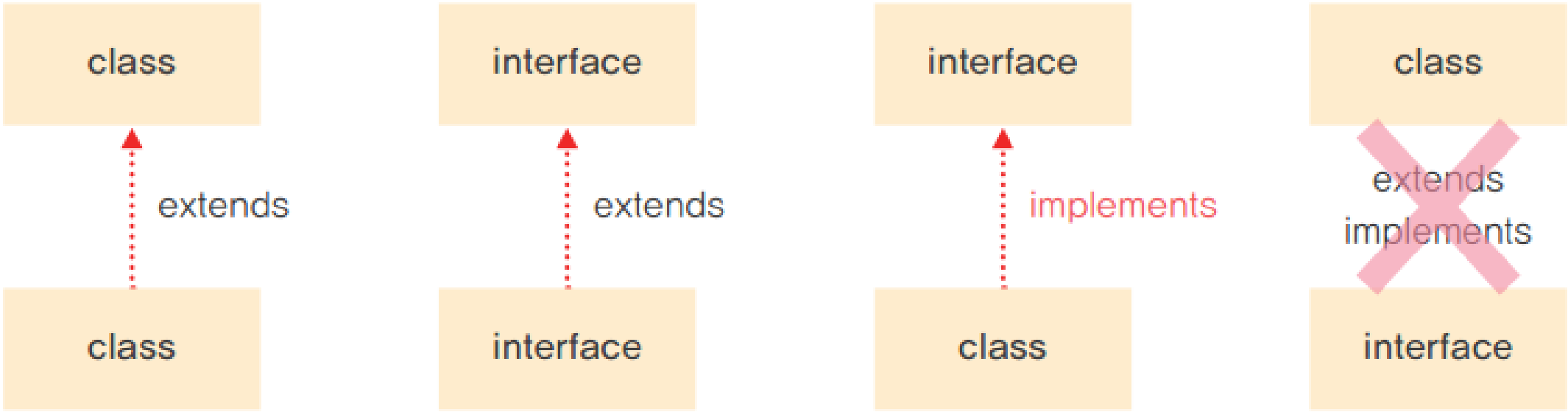
인터페이스 VS 추상 클래스

분류	인터페이스	추상 클래스
구현 메서드	포함 불가(단, 디폴트 메서드와 정적 메서드는 예외)	포함 가능
인스턴스 변수	포함 불가능	포함 가능
다중 상속	가능	불가능
디폴트 메서드	선언 가능	선언 불가능
생성자와 main( )	선언 불가능	선언 가능
상속에서의 부모	인터페이스	인터페이스, 추상 클래스
접근 범위	모든 멤버를 공개	추상 메서드를 최소한 자식에게 공개

인터페이스 기본

클래스와 인터페이스의 관계

- 인터페이스는 인터페이스를 상속 받을 수 있으나,
  클래스를 상속 받을 수는 없다.



인터페이스 기본

인터페이스 간 상속

- 하나의 인터페이스가 다른 인터페이스의 추상 메서드들을 포함하도록 상속 관계를 만들 수 있다.
- **extends** 키워드를 통해 인터페이스 간 상속 관계 명시.
- 클래스 간 상속과 달리, 여러 부모 인터페이스를 상속할 수 있음.

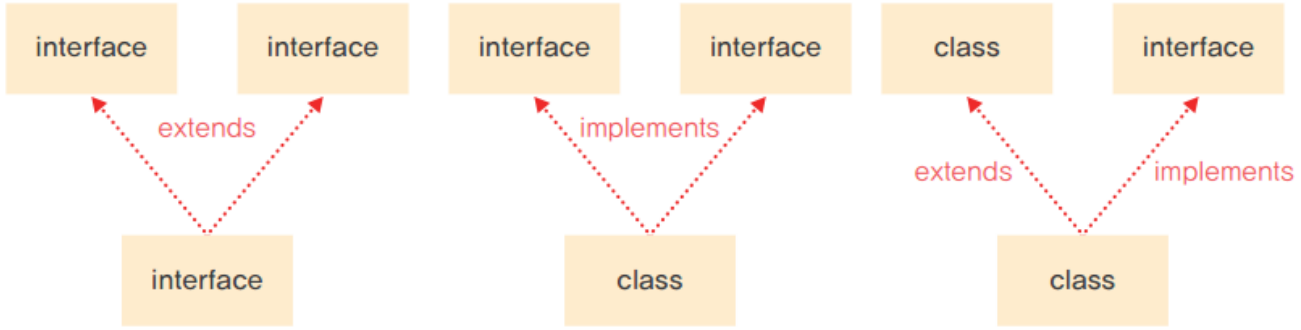
```
// 인터페이스를 상속하려면 extends 키워드를 사용한다.
interface 자식인터페이스 extends 부모인터페이스 {
}
```

```
// 상속할 인터페이스가 여러 개라면 쉼표(,)로 연결한다.
interface 자식인터페이스 extends 부모인터페이스1, 부모인터페이스2 {
}
```

인터페이스 기본

인터페이스를 이용한 다중 상속 효과

- 다음을 통해 다중 상속과 유사한 효과를 얻을 수 있다.
  - **인터페이스 간 상속** ——— `interface 자식인터페이스 extends 부모인터페이스1, 부모인터페이스2`
  - **클래스의 다중 인터페이스 구현** ——— `class 자식클래스 implements 부모인터페이스1, 부모인터페이스2`



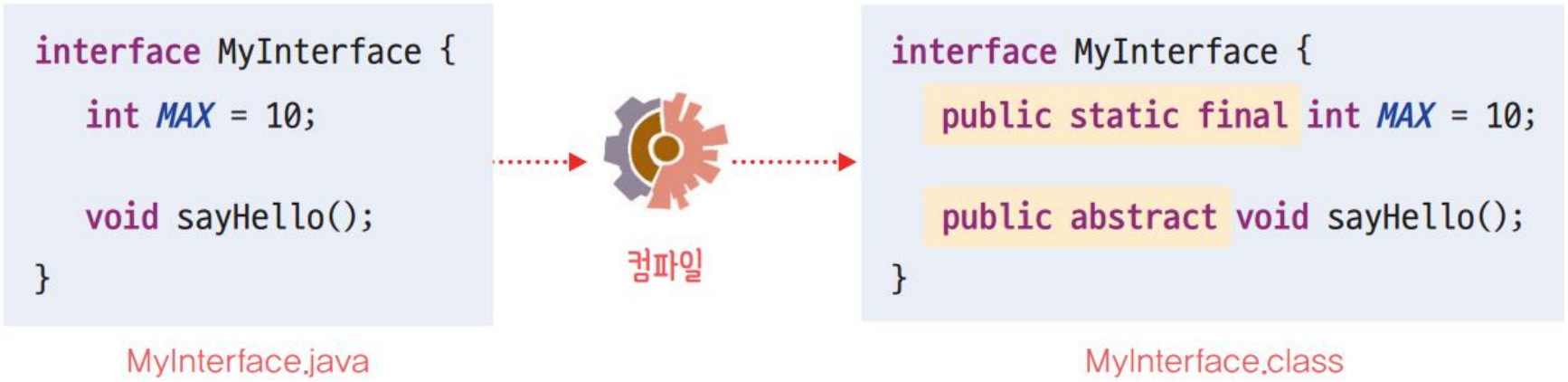
구분	추상 클래스	인터페이스
핵심 역할	내부 구조의 공통 구현 공유	외부와의 기능 규약 정의
공유 대상	"같은 종류의 클래스들"	"서로 다른 종류의 클래스들"
적용 범위	**보통 한 시스템 내, 한 팀 내 설계 통일**	**여러 팀·프로젝트·조직 간 표준화**
목표	코드 재사용, 부분 구현 제공	호환성, 표준화, 확장성
비유	"공통된 부모"	"계약서(Contract)"

- **추상 클래스**는 내부 구조의 공통점을 묶어서 **"코드 재사용 + 기본 구현 제공"**
- **인터페이스**는 서로 다른 시스템 간의 **"규약 통일 + 확장성 확보**

인터페이스 응용

인터페이스 상수

- 인터페이스에 선언된 필드들은 public static final 제어자를 취한다.
- 클래스의 전역 상수와 마찬가지로, 인스턴스와 별개로 접근 가능.



Practice

인터페이스 실습

인터페이스 선언 및 활용, 인터페이스 상수

예제:

- 7-4 (p. 277)
- 7-6 (p. 278)
- 7-7 (p. 278)
- 7-8 (p. 279)
- 7-9 (p. 279)
- 7-10 (p. 280)

인터페이스와 다형성

인터페이스 타입

- 인터페이스도 클래스처럼 하나의 타입이므로 변수를 인터페이스 타입으로 선언 가능

- 인터페이스의 구현 클래스는 그 인터페이스의 자식 타입

인터페이스와 다형성

인터페이스 → 구현 클래스 타입 변환

- 인터페이스 타입 변수가 구현 객체를 참조한다면 강제 타입 변환 가능
- 모든 TV는 Controllable이지만, 그 역은 항상 성립하지 않음 → 아래로의 강제 타입 변환 시 유의

```
Controllable controllable = new TV();
TV tv = (TV) controllable;
```

Practice

인터페이스와 다형성

인터페이스 타입 대상의 자동/강제 타입 변환 실습

예제:

- 7-11 (p. 282)
- 7-12 (p. 283)
- 7-13 (p. 284)