

---

Chapter 16-1

**final**





# 금지의 규제 final 키워드

---

final (변경 금지의 규제)

- final 키워드는 클래스, 메서드, 변수에 적용되며 abstract와 동시에 사용될 수 없습니다.
- final 클래스의 경우에는 상속이 안됩니다. 즉 서브클래스를 가질 수 없습니다.
- final 메서드는 재정의할 수 없습니다.
- final 변수는 값을 변경할 수 없습니다.

# final 클래스 – final 메서드

## final class

- 클래스 선언 시 final을 사용하면 그 클래스는 **상속이 불가능**해집니다.
- final 클래스는 자식 클래스를 가질 수 없고, 오직 외부에서 객체 생성을 통해서만 사용할 수 있습니다.
- final 클래스의 대표적인 예가 String 클래스입니다. 사용자가 임의로 String 클래스를 상속받아 메서드를 재정의하는 것을 방지하기 위한 것입니다.

## final method

- final 메서드는 자식 클래스에서 부모 클래스의 **메서드를 재정의하지 못하게 합니다.**
- 하지만 클래스에 final이 붙지 않는다면 상속은 가능하므로 자식 클래스에서 final 메서드의 참조는 가능합니다.
- 자식 클래스에서 반드시 부모의 메서드를 기능의 변경없이 사용하도록 강요할 경우에 final 메서드를 선언합니다.

```
public final class Parent{  
    public final void method(){  
        System.out.println("Parent - method()");  
    }  
}
```

final 클래스는 상속 불가  
final 메서드는 오버라이딩 불가

```
public class Child extends Parent {  
    public void method(){  
        System.out.println("Parent - method()");  
    }  
}
```

# final 변수

## final 변수

- final 변수는 한번 값을 할당하면 그 값을 **변경할 수 없습니다**.
- final 변수는 선언시에 초기화하는 방법과 생성자를 통하여 초기화하는 방법이 있는데 만약 초기화하지 않고 남겨두면 컴파일 에러가 발생합니다.

```
public class Person {  
  
    public final String nation = "한국";  
    public final String ssn;  
    public String name;  
  
    public Person(String ssn, String name) {  
        this.ssn = ssn;  
        this.name = name;  
    }  
}
```

final 필드는 직접 초기화,  
또는 생성자로 초기화 해야한다

```
public class MainClass {  
  
    public static void main(String[] args) {  
  
        Person kim = new Person("11111-1111", "김한국");  
  
        kim.name = "김마이클";  
        kim.nation = "미국"; //에러  
        kim.ssn = "12345"; //에러 final 변수는 변경할 수 없다  
    }  
}
```



# static과 final이 동시에 붙으면 상수!

상수(static final)

- 자바에서는 불변의 값을 저장하는 필드를 **상수**(constant)라고 부릅니다.
- 상수는 객체마다 저장할 필요가 없는 공용성을 가져야 하며, 여러가지 값으로 초기화될 수 없기 때문에 static과 final 제한자를 동시에 붙여 선언합니다.
- 상수 이름은 모두 대문자로 작성하는 것이 관례입니다. 연결된 단어라면 (\_)로 단어들을 연결해줍니다.

ex)

```
public static final long VERSION = 1L;
```

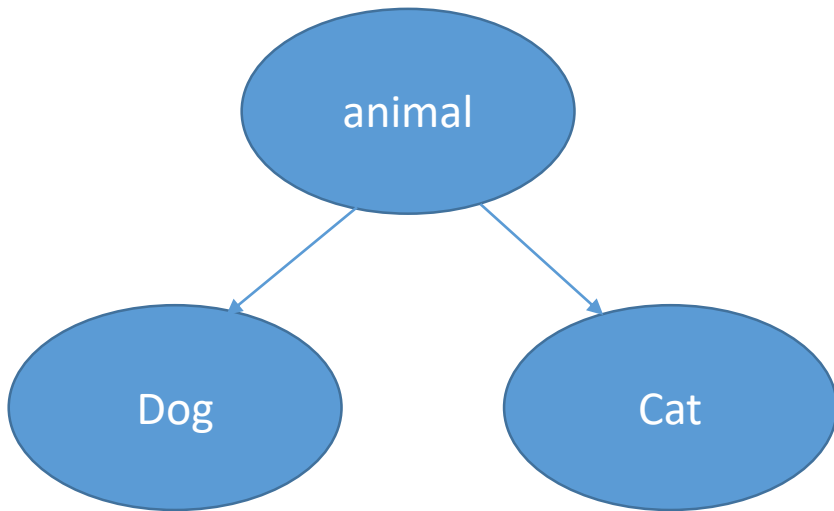
# **abstract**



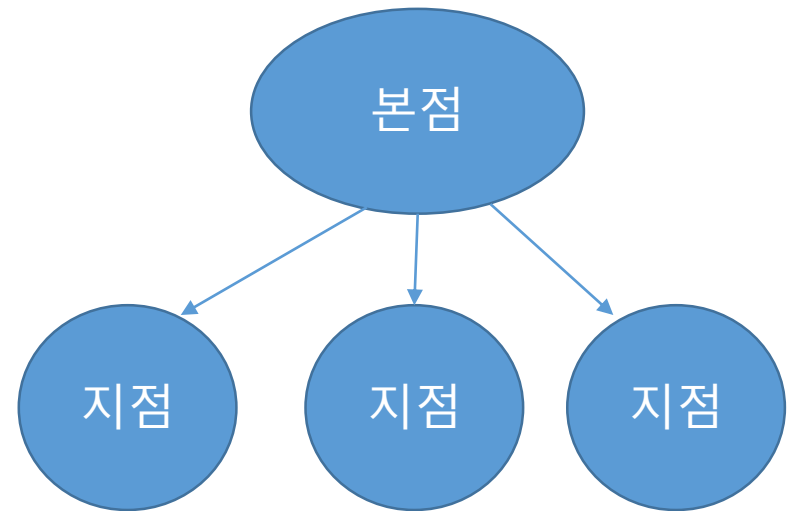
# abstract의 의미

## abstract


- abstract 키워드는 **클래스**와 **메서드**에 적용됩니다.
- 추상(abstract) 클래스는 실제 클래스들의 메서드들의 이름을 **통일할 목적**으로 사용합니다.
- 추상(abstract) 메서드가 있는 클래스는 **반드시 추상 클래스**여야 합니다.
- 그러나 추상 클래스에 반드시 추상 메서드만 선언할 필요는 없고 일반 메서드도 선언할 수 있습니다.



공통 기능만 담고 있는 animal 클래스는 객체로 생성될 필요가 있나?



본점에 있는 메뉴 목록은 각각 지점에서 모두 갖고 있어야 한다



# 추상클래스 - 추상메서드

## 추상 클래스

- 추상 클래스는 new 키워드를 이용해서 객체를 만들지 못하고 오직 상속을 통해서 **자식 클래스로 구체화** 시켜야 합니다.
- 추상 클래스도 일반 클래스와 마찬가지로 **멤버변수, 생성자, 메서드**를 선언할 수 있습니다.
- new를 사용하여 직접 생성자를 호출할 수는 없지만 자식 객체가 생성될 때 super()를 호출하여 추상 클래스 객체를 생성하므로 추상 클래스도 생성자가 반드시 있어야 합니다.

---

## 추상 메서드

- 추상 메서드는 추상 클래스 내에서만 선언할 수 있습니다.
- 추상 메서드는 메서드의 선언부만 있고 메서드 실행 내용이 들어가는 **중괄호 {}가 없는 메서드**를 말합니다.
- 추상 클래스를 설계할 때 **자식 클래스가 반드시 실행 내용을 채우도록 강요하고 싶은 메서드가 있을 경우**, 해당 메서드를 추상 메서드로 선언합니다.
- 자식 클래스에서 반드시 부모 추상클래스의 추상 메서드를 **재정의하여 실행 내용을 작성**해야 합니다. 그렇지 않으면 컴파일 에러가 납니다.



# abstract 예제

abstract\_/Shape.java

```
1: package abstract_;
2:
3: public abstract class Shape {
4:     private int x;
5:     private int y;
6:
7:     public Shape() {}
8:     public Shape(int x, int y) {
9:         this.x = x;
10:        this.y = y;
11:    }
12:
13:    public abstract double getArea(); //구현이 되어있지 않음
14:
15:    public String position() {
16:        return "[x=" + x + ", y=" + y + "]";
17:    }
18: }
```

abstract\_/Circle.java

```
1: package abstract_;
2:
3: public class Circle extends Shape {
4:     private int radius;
5:
6:     public Circle(int r) {
7:         this(0, 0, r);
8:     }
9:     public Circle(int x, int y, int r) {
10:         super(x, y);
11:         radius = r;
12:     }
13:
14:     public double getArea() {
15:         return (Math.PI * radius * radius);
16:     }
```

반드시 overriding되어야 한다

abstract\_/AbstractExample.java

```
1: package abstract_;
2:
3: public class AbstractExample {
4:     public static void main(String[] args) {
5:         Shape circle = new Circle(10);
6:
7:         System.out.println("원의 넓이는 : " + circle.getArea());
8:         System.out.println("도형의 위치는 : " + circle.position());
9:     }
10: }
```

Console

<terminated> AbstractExample [Java Application]  
원의 넓이는 : 314.1592653589793  
도형의 위치는 : [x=0, y=0]

다형성  
자식 타입으로 생성해서  
부모 타입에 저장한다



# 자바에서 abstract의 의미

사용자 클래스를 정의할 때 굳이 abstract클래스로 설계할 필요는 없다

하지만, 자바 내부의 많은 클래스는 abstract클래스로 정의 되어있음을 알아둬야 한다

또한,

자식 클래스로 생성해서 부모클래스(추상클래스)에 저장해서 사용가능하다는 점에 익숙해져야 한다



# Chapter 15

## 수고하셨습니다