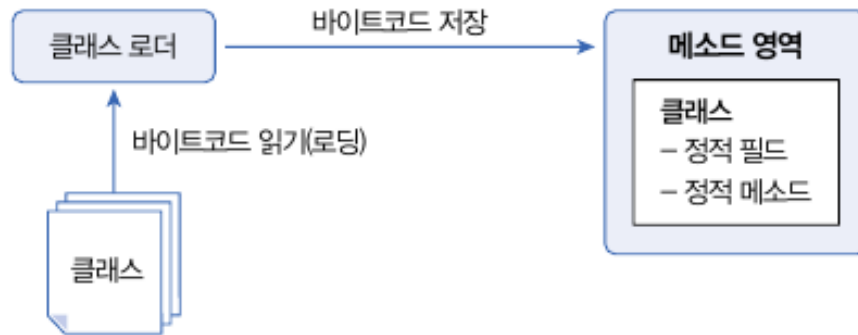


정적 멤버

## ▣ 정적 멤버

- 정적(static) 멤버란 메소드 영역의 클래스에 고정적으로 위치하는 멤버를 말한다.
- 정적 멤버는 객체를 생성할 필요 없이 클래스를 통해 바로 사용이 가능하다.



## ◎ 정적 멤버 선언

- 객체마다 가지고 있을 필요성이 없는 공용적인 필드는 정적 필드로 선언하는 것이 좋다.

```
public class 클래스 {  
    //정적 필드 선언  
    static 타입 필드 [= 초기값];  
  
    //정적 메소드  
    static 리턴타입 메소드( 매개변수, ... ) { ... }  
}
```


```
public class Calculator {  
    String color;           //계산기별로 색깔이 다를 수 있다.  
    static double pi = 3.14159; //계산기에서 사용하는 파이( $\pi$ ) 값은 동일하다.  
    void setColor(String color) { this.color = color; } //인스턴스 메소드  
    static int plus(int x, int y) { return x + y; } //정적 메소드  
}
```

## ◎ 정적 멤버 사용하기

- 클래스가 메모리로 로딩되면 정적 멤버를 바로 사용할 수 있는데, 클래스 이름과 함께 도트(.) 연산자로 접근한다.

```
public class Calculator {  
    static double pi = 3.14159;  
    static int plus(int x, int y) { ... }  
    static int minus(int x, int y) { ... }  
}
```

```
double result1 = 10 * 10 * Calculator.pi;  
int result2 = Calculator.plus(10, 5);  
int result3 = Calculator.minus(10, 5);
```

※ 정적 필드와 정적 메소드는 객체 참조 변수로도 접근이 가능하다.  
이클립스에서 정적 멤버를 객체 참조 변수로 접근 했을 경우, 경고 표시()를 낸다.

```
Calculator myCalcu = new Calculator();  
double result1 = 10 * 10 * myCalcu.pi;  
int result2 = myCalcu.plus(10, 5);  
int result3 = myCalcu.minus(10, 5);
```

## ◎ 정적 블록

- 정적 필드는 필드 선언과 동시에 초기값을 주는 것이 일반적이다.
- 그러나 복잡한 초기화 작업이 필요하다면 정적 블록(static block)을 이용해야 한다.

```
static double pi = 3.14159;
```

```
static {  
    ...  
}
```

### ⚙ 생성자에서 초기화를 하지 않는 정적 필드

정적 필드는 객체 생성 없이도 사용할 수 있기 때문에 생성자에서 초기화 작업을 하지 않는다. 생성자는 객체 생성 후 실행되기 때문이다.

## ◎ 인스턴스 멤버 사용 불가

- 정적 메소드와 정적 블록은 객체가 없어도 실행된다는 특징 때문에 내부에 인스턴스 필드나 인스턴스 메소드를 사용할 수 없다. 또한 객체 자신의 참조인 this도 사용할 수 없다.
- 정적 메소드와 정적 블록에서 인스턴스 멤버를 사용하고 싶다면 객체를 먼저 생성하고 참조 변수로 접근해야 한다.

```
public class ClassName {  
    //인스턴스 필드와 메소드 선언  
    int field1;  
    void method1() { ... }  
  
    //정적 필드와 메소드 선언  
    static int field2;  
    static void method2() { ... }  
  
    //정적 블록 선언  
    static {  
        field1 = 10;      (x)  
        method1();        (x) } 컴파일 에러  
        field2 = 10;      (o)  
        method2();        (o)  
    }  
  
    //정적 메소드 선언  
    static void Method3 {  
        this.field1 = 10;  (x)  
        this.method1();    (x) } 컴파일 에러  
        field2 = 10;      (o)  
        method2();        (o)  
    }  
}
```

```
static void Method3() {  
    //객체 생성  
    ClassName obj = new ClassName();  
    //인스턴스 멤버 사용  
    obj.field1 = 10;  
    obj.method1();  
}
```

# 클래스변수와 인스턴스변수

“인스턴스변수는 인스턴스가 생성될 때마다 생성되므로 인스턴스마다 각기 다른 값을 유지할 수 있지만, 클래스변수는 모든 인스턴스가 하나의 저장공간을 공유하므로 항상 공통된 값을 갖는다.”



속성	무늬 숫자
	폭 높이
기능	...

인스턴스변수

클래스변수

```
class Card {
```

```
    String kind; // 무늬
```

```
    int number; // 숫자
```

```
    static int width = 100; // 폭
```

```
    static int height = 250; // 높이
```

```
}
```