

static 멤버

54

□ static 멤버 선언

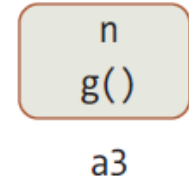
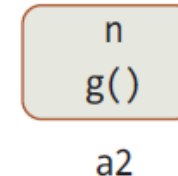
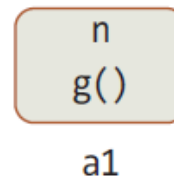
```
class StaticSample {  
    int n;           // non-static 필드  
    void g() {...}   // non-static 메소드  
  
    static int m;     // static 필드  
    static void f() {...} // static 메소드  
}
```

□ 객체 생성과 non-static 멤버의 생성

- non-static 멤버는 객체가 생성될 때, 객체마다 생긴다

```
class A {  
    int n;  
    void g() {...}  
}
```

```
A a1 = new A();  
A a2 = new A();  
A a3 = new A();
```



객체마다 `n`, `g()`의 non-static 멤버들이 생긴다

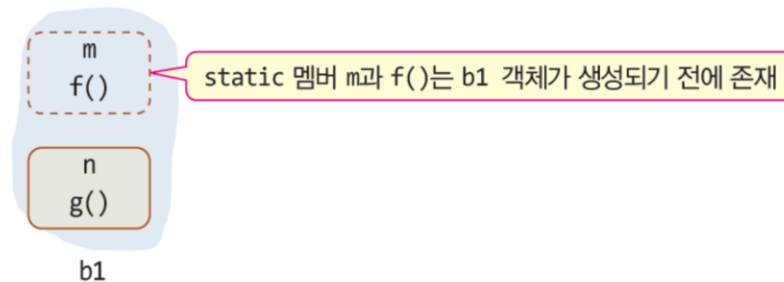
static 멤버의 생성

55

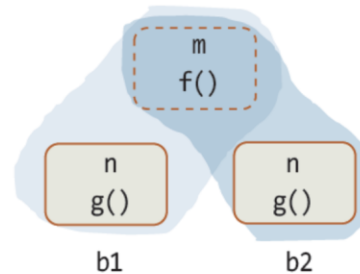
- static 멤버는 클래스당 하나만 생성
- 객체들에 의해 공유됨

```
class StaticSample {  
    int n;  
    void g() {...}  
    static int m;  
    static void f() {...}  
}
```

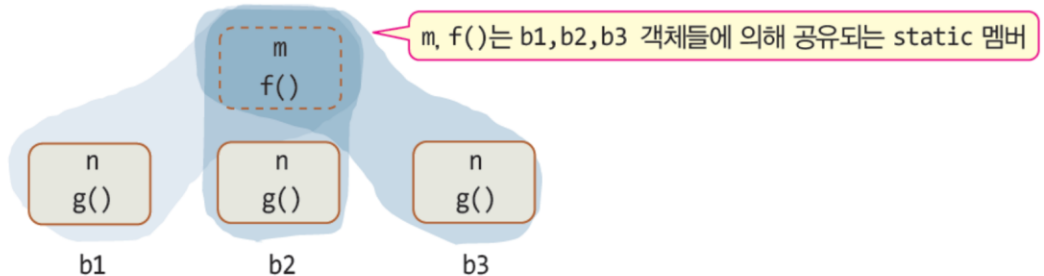
StaticSample b1 = new StaticSample(); 생성 후



StaticSample b2 = new StaticSample(); 생성 후



StaticSample b3 = new StaticSample(); 생성 후



static 멤버와 non-static 멤버 특성 정리

56

	non-static 멤버	static 멤버
선언	<pre>class Sample { int n; void g() {...} }</pre>	<pre>class Sample { static int m; static void g() {...} }</pre>
공간적 특성	멤버는 객체마다 별도 존재 • 인스턴스 멤버라고 부름	멤버는 클래스당 하나 생성 • 멤버는 객체 내부가 아닌 별도의 공간에 생성 • 클래스 멤버라고 부름
시간적 특성	객체 생성 시에 멤버 생성됨 • 객체가 생길 때 멤버도 생성 • 객체 생성 후 멤버 사용 가능 • 객체가 사라지면 멤버도 사라짐	클래스 로딩 시에 멤버 생성 • 객체가 생기기 전에 이미 생성 • 객체가 생기기 전에도 사용 가능 • 객체가 사라져도 멤버는 사라지지 않음 • 멤버는 프로그램이 종료될 때 사라짐
공유의 특성	공유되지 않음 • 멤버는 객체 내에 각각 공간 유지	동일한 클래스의 모든 객체들에 의해 공유됨

static 멤버 사용

57

□ 클래스 이름으로 접근 가능

```
StaticSample.m = 3; // 클래스 이름으로 static 필드 접근  
StaticSample.f();   // 클래스 이름으로 static 메소드 호출
```

□ 객체의 멤버로 접근 가능

```
StaticSample b1 = new StaticSample();  
  
b1.m = 3; // 객체 이름으로 static 필드 접근  
b1.f();   // 객체 이름으로 static 메소드 호출
```

□ non-static 멤버는 클래스 이름으로 접근 안 됨



```
StaticSample.n = 5; // n은 non-static이므로 컴파일 오류  
StaticSample.g();   // g()는 non-static이므로 컴파일 오류
```

static의 활용

58

1. 전역 변수와 전역 함수를 만들 때 활용
2. 공유 멤버를 만들고자 할 때
 - ▣ static으로 선언한 멤버는 클래스의 객체들 사이에 공유

예제 4-11 : static 멤버를 가진 Calc 클래스 작성

59

전역 함수로 작성하고자 하는 abs, max, min의 3개 함수를 static 메소드를 작성하고 호출하는 사례를 보여라.

```
class Calc {  
    public static int abs(int a) { return a>0?a:-a; }  
    public static int max(int a, int b) { return (a>b)?a:b; }  
    public static int min(int a, int b) { return (a>b)?b:a; }  
}  
  
public class CalcEx {  
    public static void main(String[] args) {  
        System.out.println(Calc.abs(-5));  
        System.out.println(Calc.max(10, 8));  
        System.out.println(Calc.min(-3, -8));  
    }  
}
```

5
10
-8

static 메소드의 제약 조건 1

60



- ▣ static 메소드는 오직 static 멤버만 접근 가능
 - 객체가 생성되지 않은 상황에서도 static 메소드는 실행될 수 있기 때문에, non-static 멤버 활용 불가
 - non-static 메소드는 static 멤버 사용 가능

```
class StaticMethod {  
    int n;  
    void f1(int x) {n = x;} // 정상  
    void f2(int x) {m = x;} // 정상  
    static int m;  
    static void s1(int x) {n = x;} // 컴파일 오류. static 메소드는 non-static 필드 n 사용 불가  
    static void s2(int x) {f1(3);} // 컴파일 오류. static 메소드는 non-static 메소드 f1() 사용 불가  
    static void s3(int x) {m = x;} // 정상. static 메소드는 static 필드 m 사용 가능  
    static void s4(int x) {s3(3);} // 정상. static 메소드는 static 메소드 s3() 호출 가능  
}
```

static 메소드의 제약 조건 2

61


- ▣ static 메소드는 this 사용불가
 - static 메소드는 객체 없이도 사용 가능하므로, this 레퍼런스 사용할 수 없음

 `static void f() { this.n = x;}` // 오류. static 메소드에서는 this 사용 불가능
 `static void g() { this.m = x;}` // 오류. static 메소드에서는 this 사용 불가능


final 클래스와 메소드

62

□ final 클래스 - 더 이상 클래스 상속 불가능

```
final class FinalClass {  
    ....  
}  
 class DerivedClass extends FinalClass { // 컴파일 오류  
    ....  
}
```

□ final 메소드 - 더 이상 오버라이딩 불가능

```
public class SuperClass {  
    protected final int finalMethod() { ... }  
}  
  
 class SubClass extends SuperClass {  
    protected int finalMethod() { ... } // 컴파일 오류, 오버라이딩 할 수 없음  
}
```

final 필드

63

- final 필드, 상수 선언
 - ▣ 상수를 선언할 때 사용

```
class SharedClass {  
    public static final double PI = 3.14;  
}
```

- ▣ 상수 필드는 선언 시에 초기 값을 지정하여야 한다
- ▣ 상수 필드는 실행 중에 값을 변경할 수 없다

```
public class FinalFieldClass {  
    final int ROWS = 10; // 상수 정의, 이때 초기 값(10)을 반드시 설정  
  
    void f() {  
        int [] intArray = new int [ROWS]; // 상수 활용  
        ROWS = 30; // 컴파일 오류 발생, final 필드 값을 변경할 수 없다.  
    }  
}
```

오류