

알기 쉽게 해설한 J A V A

9th edition

8장 클래스-속성

- Section 1 클래스의 일반 구조
- Section 2 클래스 선언
- Section 3 객체의 선언과 생성
- Section 4 멤버 변수와 메소드 변수
- Section 5 변수의 유효 범위
- Section 6 멤버 변수 접근 한정자

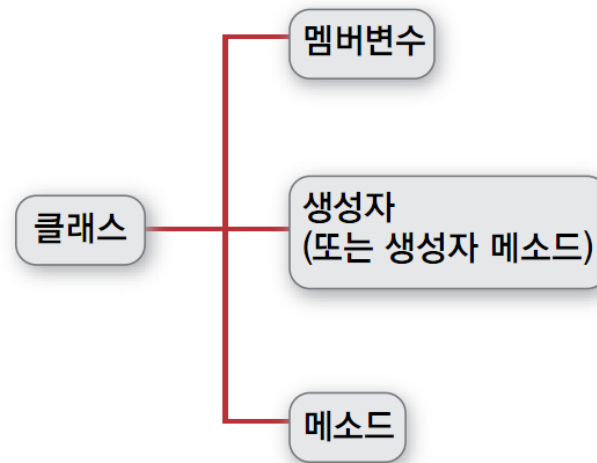


• 학습 목표

9th edition

- 이 장에서는 클래스의 속성에 관해 학습합니다. 클래스는 크게 속성과 기능(메소드)으로 구성됩니다. 이 장에서는 속성에 대해 학습하고, 다음 장에서 기능에 대해 학습합니다.
- 기존의 일반적인 절차 지향의 프로그램과 객체 지향의 프로그램을 비교하여 차이를 학습합니다.
- 클래스의 일반 구조와 클래스 선언 부분에 대해 학습합니다.
- 클래스로부터 객체를 생성하는 과정을 학습합니다.
- 클래스의 속성에 해당하는 멤버 변수와 메소드에서 사용되는 변수에 관해 학습합니다.
- 변수의 유효 범위에 관해 학습합니다.
- 캡슐화를 지원하는 멤버 변수 접근 한정자에 관해 학습합니다.

- 자바 프로그램은 클래스로부터 객체를 생성하여 프로그램이 작성된다
 - 객체를 생성하기 위해서는 클래스를 작성하여야 한다.
- 클래스는 멤버 변수, 생성자, 메소드 3가지 요소로 구성된다
 - 클래스가 항상 3가지 요소를 모두 가지는 것은 아니다



● 클래스의 예 : 멤버변수, 생성자 메소드로 구성된 클래스

```
01: public class Box {  
02:     int width;   
03:     int height;  
04:     int depth;  
05:     public Box(int w, int h, int d)   
06:     {  
07:         width = w;  
08:         height = h;  
09:         depth = d;  
10:     }  
11:     public void volume() {  
12:         int vol;  
13:         vol = width * height * depth;  
14:         System.out.println("Volume is "+vol);  
15:     }  
16: }
```

속성-멤버 변수

기능-생성자 메소드

기능-메소드

● 클래스의 예 : 속성만 가지는 클래스

```
01: public class Box {
02:     int width;
03:     int height;
04:     int depth;
05: }
```

속성-멤버 변수

● 클래스의 예 : 속성과 메소드를 가지는 클래스

```
01: public class Box {
02:     int width;
03:     int height;
04:     int depth;
05:     public void volume() {
06:         int vol;
07:         vol = width * height * depth;
08:         System.out.println("Volume is "+vol);
09:     }
10: }
```

속성-멤버 변수

기능-메소드

● 클래스 선언

【형식】 클래스 선언

```
[public/final/abstract] class Class-name {
```

```
.....
```

```
..... 클래스의 속성과 기능을 기술
```

```
.....
```

```
}
```

● 클래스의 한정자

- public : 모든 클래스에서 접근 가능
- 한정자 사용 안 함 : 같은 패키지 내의 클래스에서만 접근 가능
- final : 서브 클래스를 가질 수 없는 클래스
- 추상(abstract) : 객체를 생성할 수 없는 클래스



● 다수 개의 클래스가 하나의 프로그램의 정의될 때

- 클래스에 붙이는 public 한정자는 main() 메소드를 가진 클래스에만 붙여야 합니다.
- 프로그램의 이름은 main() 메소드를 가진 클래스의 이름과 동일해야 합니다.
- 한 패키지에는 동일한 이름의 클래스가 중복될 수 없습니다.

● 객체의 선언과 생성

【 형식 】 객체의 선언

클래스명 객체변수명;

【 예 】

```
Box mybox1;  
Avg student1;  
String name;
```

객체의 선언만으로 객체가 생성되지 않습니다. 객체가 메모리상에서 생성되기 위해서는 선언된 객체를 명시적으로 생성시켜야 합니다. 객체를 생성하기 위한 구문은 다음과 같습니다.

【 형식 】 객체의 생성

객체변수명 = new 클래스명();

【 예 】

```
mybox1 = new Box();  
student1 = new Avg();  
name = new String("C.S.Kim");
```


- 한 문장으로 객체의 선언과 생성

[형식] 객체의 선언과 생성

클래스명 객체변수명 = new 클래스명();

[예]

```
Box mybox1 = new Box();  
Avg student1 = new Avg();  
String name = new String("C.S.Kim");
```

● 객체의 선언과 생성

객체의 선언 : 객체의 선언은 null값을 가진 변수만을 의미한다.

Box mybox1;



Box mybox2;



객체의 생성 : 객체에 대한 메모리가 할당되고, 변수(객체변수)는 객체에 대한 참조(주소)를 가진다.

mybox1 = new Box();



mybox2 = new Box();



예제 8.1

Box1Test1.java

```

01: class Box1 {
02:     int width;
03:     int height;
04:     int depth;
05: }
06: public class Box1Test1 {
07:     public static void main(String args[]) {
08:         Box1 mybox1 = new Box1();
09:         Box1 mybox2 = new Box1();
10:         int vol1, vol2;
11:
  
```

3개의 속성을 가진 클래스 Box1을 선언

클래스 Box1으로부터 두 개의 객체 생성

```

12:    mybox1.width = 78;
13:    mybox1.height = 145;
14:    mybox1.depth = 87;
15:
16:    mybox2.width = 48;
17:    mybox2.height = 45;
18:    mybox2.depth = 137;
19:
20:    vol1 = mybox1.width * mybox1.height * mybox1.depth;
21:    System.out.println("첫 번째 박스의 부피는 " + vol1 + "입니다");
22:
23:    vol2 = mybox2.width * mybox2.height * mybox2.depth;
24:    System.out.println("두 번째 박스의 부피는 " + vol2 + "입니다");
25: }
26: }
  
```

각 객체의 속성값을 직접 지정

박스의 부피를 계산하여 출력

실행 결과

첫 번째 박스의 부피는 983970입니다
 두 번째 박스의 부피는 295920입니다

● 멤버 변수

- 클래스내에 메소드 밖에 선언된 변수로서 객체변수, 클래스 변수, 종단 변수로 구분된다

● 생성자, 메소드의 변수

- 생성자나 메소드에는 자체적으로 선언하여 사용하는 지역 변수와 호출 시 지정되는 매개 변수로 구분된다

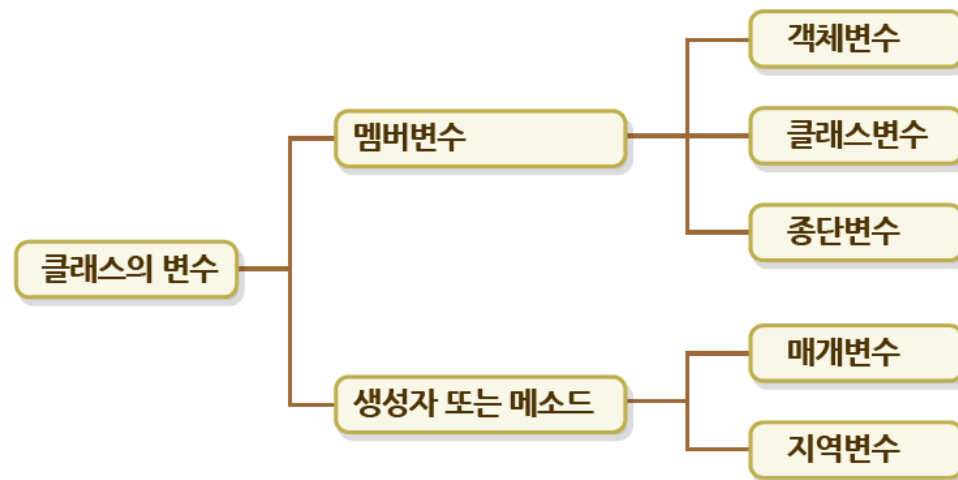


그림 8-3 클래스에서의 변수

● 멤버 변수의 선언

- 한정자인 public, private, protected는 다음절에서 설명

[형식] 멤버 변수의 선언

[public/private/protected] [static] [final] 변수형 변수명;

[예]

```
public int width;  
private double rate;  
static int idnumber;  
final int MAX=100;  
public Box mybox1;  
private String passwd;  
public final int MIN=1;
```

- static : 클래스 변수
- final : 종단 변수

● 생성자나 메소드의 변수(매개변수와 지역변수)

[형식] 생성자, 메소드에서의 매개 변수와 지역 변수의 선언

[final] 변수형 변수명;

[예]

```
public void cc(final int x, final int y) { ←----- 메소드의 매개 변수에 final을 지정.  
    final int Max = 10; ←----- 메소드의 지역 변수로 final을 선언  
    String name="C.S.Kim"; ←----- 지역 변수로 문자열 변수 선언  
    private int num; ←----- 오류 발생. 객체 변수에만 사용 가능  
}
```



4 멤버 변수와 메소드 변수

4-1 객체변수, 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

- 객체변수나 지역 변수는 변수가 가지는 값의 형(기본 자료형, 참조 자료형)에 따라 다른 특성을 가진다
 - 기본 자료형 : 값을 가진다
 - 참조 자료형 : 주소를 가진다



4 멤버 변수와 메소드 변수

4-1 객체변수와 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

● 예

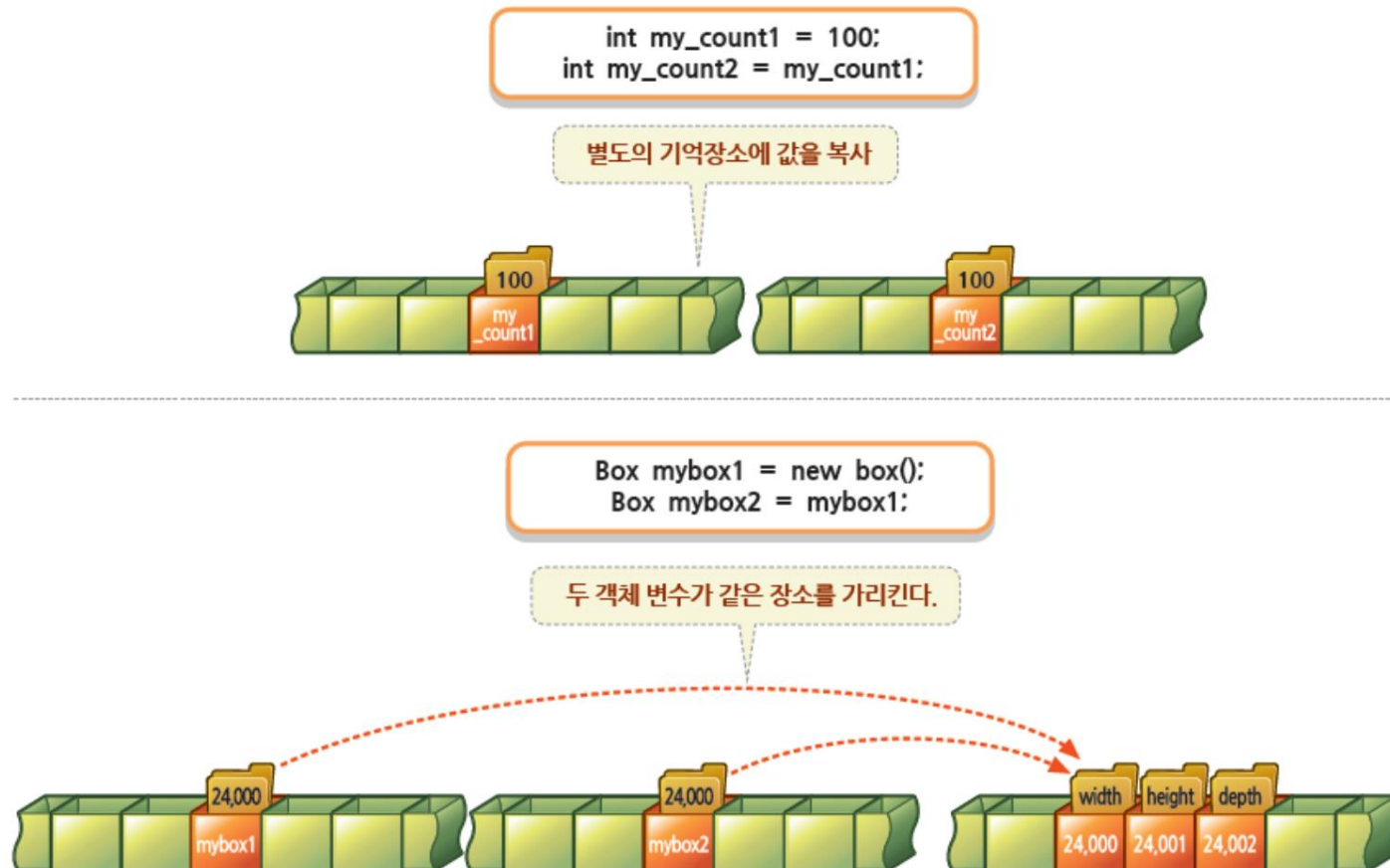
```
.....  
int my_count1 = 100;  
int my_count2 = my_count1;  
Box mybox1 = new Box();  
Box mybox2 = mybox1;  
.....
```

4 멤버 변수와 메소드 변수

4-1 객체변수와 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

● 예



예제 8.2

Box2Test1.java

```

01: class Box2 {
02:     int width=10;
03:     int height=20;
04:     int depth=30;
05: }
06: public class Box2Test1 {
07:     public static void main(String args[]) {
08:         int myint1 = 100;
09:         int myint2 = myint1;
10:         System.out.println("첫 번째 값 : " + myint1 + " 두 번째 값 : " +
            myint2);
11:         myint1 = 200;
12:         System.out.println("첫 번째 값 : " + myint1 + " 두 번째 값 : " +
            myint2);
13:         Box2 mybox1 = new Box2();
14:         Box2 mybox2 = new Box2();
15:         mybox1.width = 20;
16:         mybox2.depth = 123;

```

3개의 속성과 값을 설정

기본 자료형 변수값을 배정

한 변수의 값을 변경하여 출력

두 개의 서로 다른 참조 자료형 변수 생성

각각의 변수를 통하여 속성값을 변경

```

17: System.out.println("mybox1.width : " + mybox1.width);
18: System.out.println("mybox1.height : " + mybox1.height);
19: System.out.println("mybox1.depth : " + mybox1.depth);
20:
21: System.out.println("mybox2.width : " + mybox2.width);
22: System.out.println("mybox2.height : " + mybox2.height);
23: System.out.println("mybox2.depth : " + mybox2.depth);
24:
25: Box2 mybox3 = mybox2;
26: mybox2.width = 1000;
27: mybox2.height = 2000;
28: System.out.println("mybox3.width : " + mybox3.width);
29: System.out.println("mybox3.height : " + mybox3.height);
30: System.out.println("mybox3.depth : " + mybox3.depth);
31: }
32: }

```

서로 영향을 받지 않는다.

참조 자료형 변수에 다른 참조 자료형 변수를 대입

값을 변경

다른 자료형 변수의 변경에 따라 값이 바뀜

실행 결과

```

첫 번째 값 : 100 두 번째 값 : 100
첫 번째 값 : 200 두 번째 값 : 100
mybox1.width : 20
mybox1.height : 20
mybox1.depth : 30
mybox2.width : 10
mybox2.height : 20
mybox2.depth : 123
mybox3.width : 1000
mybox3.height : 2000
mybox3.depth : 123

```

4 멤버 변수와 메소드 변수

4-1 객체변수와 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

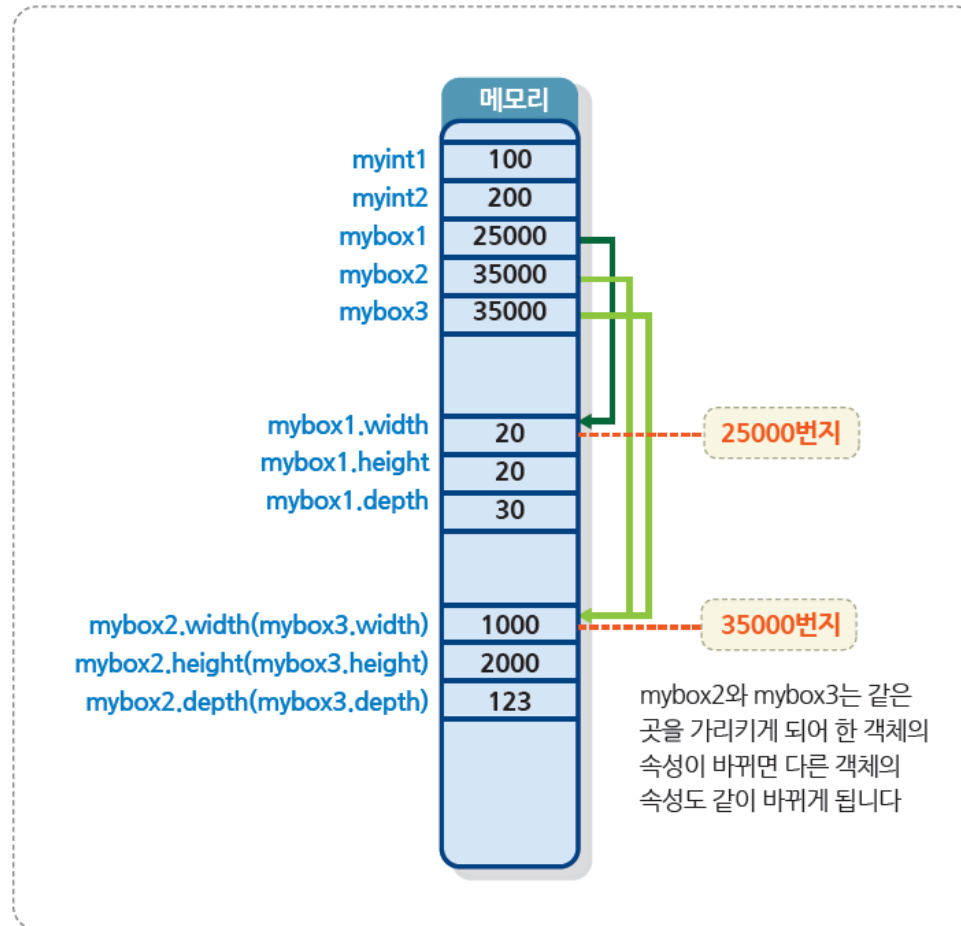


그림 8-5 Box2Test1 프로그램 종료 시점에서의 메모리 구조



4 멤버 변수와 메소드 변수

4-1 객체변수와 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

● 객체 변수와 지역 변수들은 초기화 과정에서 약간의 차이가 있다

- 객체 변수들은 변수를 초기화하지 않아도, 객체가 생성되면 묵시적 값이 자동으로 설정(사실은 객체가 생성되면서 묵시적 값으로 초기화를 수행)
- 메소드 지역 변수는 변수의 값을 명시적으로 초기화하지 않으면 구문 오류가 발생



4 멤버 변수와 메소드 변수

4-1 객체변수와 지역 변수(매개 변수 포함)의 형과 묵시적 초기화

9th edition

예제 8.3

InitialTest1.java

```
01: class Initial{
02:     int number;
03:     double rate;
04:     String name;
05:     int[] score;
06:     public void aMethod() {
07:         int count;
08:         System.out.println(number);
09:         //System.out.println(count);
10:     }
11: }
```

클래스의 속성으로 값을 지정하지 않고
객체 변수만 선언

메소드 선언

메소드 지역 변수 선언

객체 변수의 값 출력

오류 발생. 초기화되지 않은
지역 변수값 출력 불가

```

12: public class InitialTest1 {
13:     public static void main(String args[]) {
14:         int var1;
15:         double var2;
16:         Initial ob1 = new Initial();
17:         //System.out.println("지역 변수 var1의 값은 : " + var1);
18:         //System.out.println("지역 변수 var2의 값은 : " + var2);
19:         System.out.println("객체 변수 number의 값은 : " + ob1.number);
20:         System.out.println("객체 변수 rate의 값은 : " + ob1.rate);
21:         System.out.println("객체 변수 name의 값은 : " + ob1.name);
22:         System.out.println("객체 변수 score의 값은 : " + ob1.score);
23:         ob1.aMethod();
24:     }
25: }
  
```

오류 발생 초기화가 이루어진 다음 사용 가능

메소드의 지역 변수로 값을 지정하지 않고 변수 선언

객체의 생성, 객체 변수의 초기화 수행

객체의 메소드 호출

묵시적인 값이 출력

실행 결과

```

객체 변수 number의 값은 : 0
객체 변수 rate의 값은 : 0.0
객체 변수 name의 값은 : null
객체 변수 score의 값은 : null
0
  
```


● 클래스 변수

- static을 사용하여 선언
- 전역변수(global variable)의 개념

[형식] 클래스 변수

static [final] 변수형 변수명;

[예]

static idnumber; ← 클래스 변수 idnumber 선언

static final fixnumber; ← 클래스 변수이면서 값이 변할 수 없는 종단 변수

● 클래스 변수

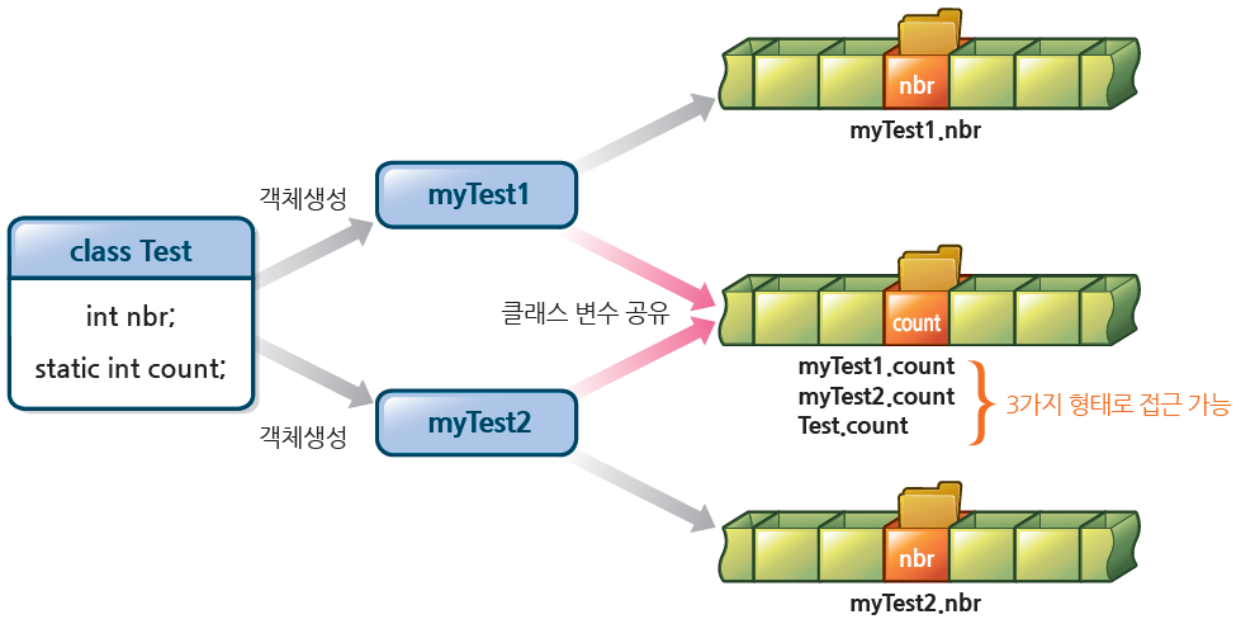


그림 8-6 클래스 변수의 사용과 메모리

예제 8.4

Box3Test1.java

```
01: class Box3 {  
02:     int width;  
03:     int height;  
04:     int depth;  
05:     long idNum;  
06:     static long boxID = 0; ← 클래스 변수로 boxID 선언  
07:     public Box3() { ←  
08:         idNum = ++boxID; ← 생성자에서 클래스 변수값을 증가시켜 속성 idNum에 배정  
09:     } ←  
10: }
```



4 멤버 변수와 메소드 변수

4-2 클래스 변수

실행 결과

mybox1의 id 번호 : 1
mybox2의 id 번호 : 2
mybox3의 id 번호 : 3
mybox4의 id 번호 : 4
마지막 생성된 박스 번호는 4번입니다

```
11: class Box3Test1 {
12:     public static void main(String args[]) {
13:         Box3 mybox1 = new Box3();
14:         Box3 mybox2 = new Box3();
15:         Box3 mybox3 = new Box3();
16:         Box3 mybox4 = new Box3();
17:
18:         System.out.println("mybox1의 id 번호 : " + mybox1.idNum);
19:         System.out.println("mybox2의 id 번호 : " + mybox2.idNum);
20:         System.out.println("mybox3의 id 번호 : " + mybox3.idNum);
21:         System.out.println("mybox4의 id 번호 : " + mybox4.boxID);
22:         System.out.println("마지막 생성된 박스 번호는 " + Box3.boxID + "번
    입니다.");
23:     }
24: }
```

객체를 생성. 생성자 수행

객체의 idNum이 증가되면서 출력

클래스명을 통하여 클래스 변수값 출력

객체명을 통하여 클래스 변수값 출력

● 예제 8.6에서 클래스 변수를 일반 변수로 속성을 변경하여 실행 : 결과 비교

```

01: class Box3 {
02:     .....
03:     long idNum;
04:     long boxID = 0;
05:     public Box3() {
06:         idNum = ++boxID;
07:     }
08: }
09: class Box3Test1 {
10:     public static void main(String args[]) {
11:         .....객체생성.....
12:         System.out.println("mybox1의 id 번호 : " + mybox1.idNum);
13:         System.out.println("mybox2의 id 번호 : " + mybox2.idNum);
14:         System.out.println("mybox3의 id 번호 : " + mybox3.idNum);
15:         System.out.println("mybox4의 id 번호 : " + mybox4.boxID);
16:         System.out.println("마지막 생성된 박스 번호는 " + Box3.boxID + "번
    입니다.");
17:     }
18: }

```

04: long boxID = 0; ← boxID를 일반 속성으로 정의. 객체가 생성될 때 모든 객체에 생성
 11:객체생성..... 모두 1을 출력. 각각의 객체가 boxID를 따로 가진다.
 15: System.out.println("mybox4의 id 번호 : " + mybox4.boxID); ←
 16: System.out.println("마지막 생성된 박스 번호는 " + Box3.boxID + "번
 입니다."); ← 오류 발생. boxID는 클래스 변수가 아니기 때문에 클래스명을 통하여 접근할 수 없다.

● 종단 변수

- final을 사용하여 선언하며 변할 수 없는 상숫값을 갖는다

【 형식 】 종단 변수

final 변수형 변수명 = 초기값;

【 예 】

```
final int MAX = 100;
static final int SONATA_LENGTH = 3200;
final int MIN; ← 오류 발생. 종단 변수는 반드시 초기화 요구
public void inc() {
    MAX = ++MAX; ← 오류 발생. 종단 변수값은 변경할 수 없다
}
public void max(final int x) { ← 메소드 매개 변수를 final로 선언.
    X++; ← 오류 발생. 종단 변수값은 변경할 수 없다
}
```

- **변수의 유효범위**

- 그 변수가 사용될 수 있는 영역을 의미

- **유효범위 측면에서의 변수들을 구분**

- 멤버 변수
- 메소드 매개변수와 지역변수(블록 변수)
- 예외 처리기 매개변수(exception handler parameter)

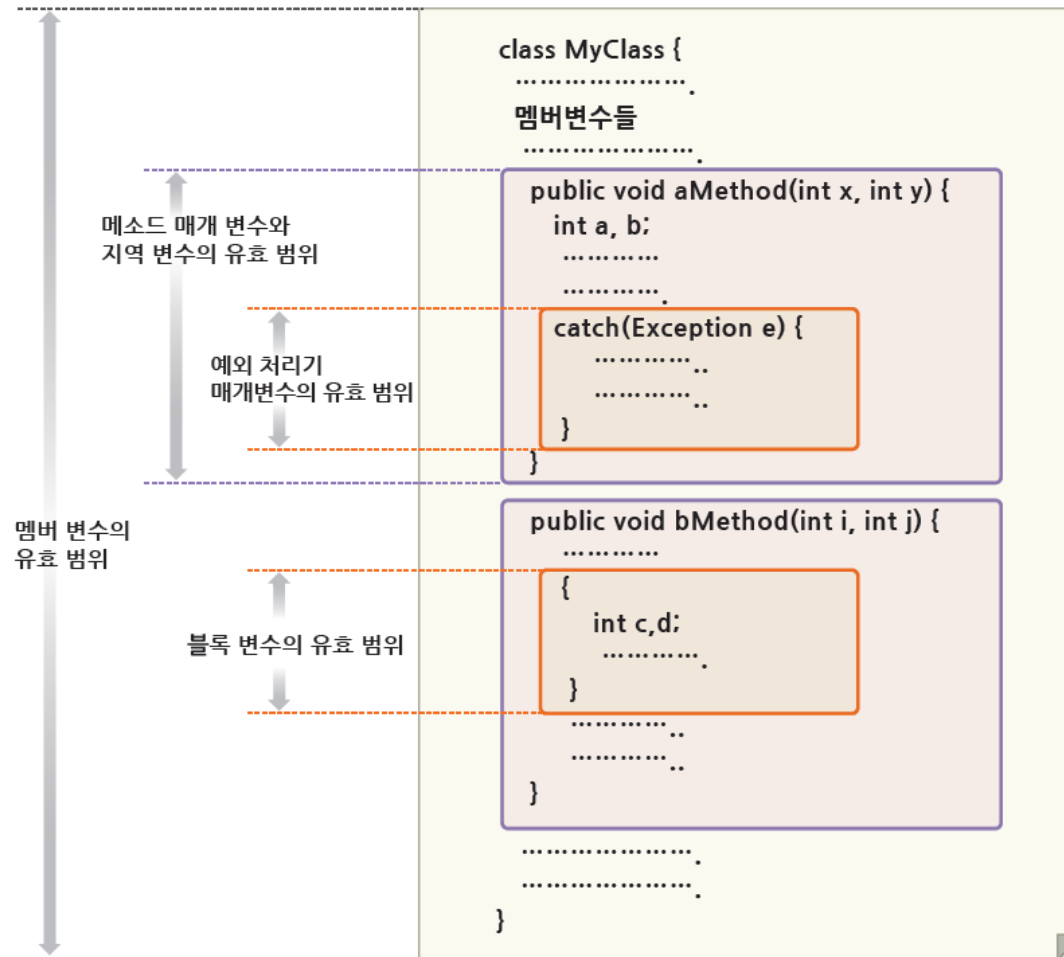


그림 8-7 변수의 유효 범위

● 유효범위와 연관된 변수의 사용 예 1

```

01: public static void main(String args[] ){ ←----- 메소드의 매개 변수 args는 메소드 전체가 유효 범위
02:     int a=1, b=2; ←----- 지역 변수 ab는 메소드 전체가 유효 범위
03:     {
04:         int c=3, d=4; ←----- c,d는 선언된 블록만 유효 범위
05:         System.out.println(a+b); ←-----
06:         System.out.println(c+d); ←----- 값을 출력
07:     }
08:     System.out.println(a+b);
09:     System.out.println(c+d); ←----- 오류 발생 유효 범위를 벗어남
10:     int sum=0;
11:     for (int i=1 ; i <=10 ; i++ ) { ←----- for문 내에서 변수 i 선언. i는 for 블록만 유효 범위
12:         sum=sum+i;
13:         System.out.println(i); ←----- 값을 10을 출력
14:     }
15:     System.out.println(i); ←----- 오류 발생 유효 범위를 벗어남
16: }

```

● 유효범위와 연관된 변수의 사용 예 2

```
01: class MyClass {  
02:     int a; ←----- 멤버 변수로 a 선언. 전체 클래스가 유효 범위  
03:     public void aMethod(int x, int y) { ←----- 메소드 매개 변수 x,y는 aMethod가 유효 범위  
04:         a = 10;  
05:         int b = 20; ←----- 메소드 지역 변수 b 선언. aMethod가 유효 범위  
06:         System.out.println(a+b+x+y); ←----- 값 출력  
07:     }  
08:     int c = x; ←----- 오류 발생. x값 사용 불가  
09:     public void bMethod(int x, int y) { ←----- bMethod 매개 변수로 x, y 사용 가능  
10:     }  
11: }
```

- 클래스 내의 멤버 변수 접근을 제한할 수 있는 방법으로 접근 한정자를 제공
- 접근 한정자를 사용한 멤버 변수의 접근 제한은 객체지향 언어의 중요 특성 중에 하나인 캡슐화와 정보 은폐를 제공
- 자바의 명시적인 접근 한정자
 - public
 - private
 - protected

● public 접근 한정자는 항상 접근 가능함을 의미

- 꼭 공개해야 하는 정보만 public으로 선언

```

01: public class Box3 {
02:     public int width;
03:     public int height;
04:     public int depth;
05:     public long idNum;
06:     static long boxID = 0;
07:     public Box3() {
08:         idNum = ++boxID;
09:     }
10: }
11: .....
12: Box3 mybox1 = new Box3();
13: mybox1.width = 7;
14: mybox2.depth = 20;
15: System.out.println(mybox1.idNum);
16: .....

```

객체 변수의 접근 한정자를 public으로 지정

객체의 속성에 직접 접근하여 값을 설정할 수 있다.

접근 가능. 객체 변수 idNum이 public이기 때문에 접근 가능

- **private 접근 한정자는 소속된 클래스 내에서만 사용 가능**

```

01: public class Box3 {
02:     private int width;
03:     private int height;
04:     private int depth;
05:     private long idNum;
06:     static long boxID = 0;
07:     public Box3() {
08:         idNum = ++boxID;
09:     }
10: }
11: .....
12: Box3 mybox1 = new Box3();
13: mybox1.width = 7;
14: mybox2.depth = 20;
15: System.out.println(mybox1.idNum);
16: .....

```

객체 변수의 접근 한정자를 private로 지정

private로 선언된 idNum 접근 가능

오류 발생 private로 지정된 변수에 접근 불가

오류 발생 private로 지정된 변수에 접근 불가

6-3 한정자를 지정하지 않고 사용

- 접근 한정자를 지정하지 않은 경우 : 같은 패키지내의 클래스에서 사용 가능

- 접근 한정자를 지정하지 않은 것은 좋은 습관이 아니다.

```
01: public class Box3 {  
02:     int width;   
03:     int height;  
04:     int depth;  
05:     long idNum;  
06:     static long boxID = 0;  
07:     public Box3() {  
08:         idNum = ++boxID;  
09:     }  
10: }
```

객체 변수의 접근 한정자를 지정하지 않고 선언

6-3 한정자를 지정하지 않고 사용

● 같은 패키지

```
01: class SamePackageClass { ← 같은 패키지 내의 클래스
02:     .....
03:     Box3 mybox1 = new Box3();
04:     mybox1.width = 7; ←
05:     mybox2.depth = 20; ← 같은 패키지 내의 클래스에서는 접근 가능
06:     System.out.println(mybox1.idNum); ←
07:     .....
08: }
```

6-3 한정자를 지정하지 않고 사용

● 다른 패키지

```
01: class AnotherPackageClass { ← 다른 패키지 클래스
02:     .....
03:     Box3 mybox1 = new Box3();
04:     mybox1.width = 7; ←
05:     mybox2.depth = 20; ← 오류 발생 다른 패키지에서는 접근 불가
06:     System.out.println(mybox1.idNum); ←
07:     .....
08: }
```


6-3 한정자를 지정하지 않고 사용



변수의 유효 범위와 멤버 변수 접근 한정자

변수의 유효 범위와 멤버 변수 접근 한정자의 개념이 혼란스럽습니다. 변수의 유효 범위는 그 변수가 선언된 위치와 연관되어 그 변수를 자유롭게 사용할 수 있는 유효 범위를 의미합니다. 멤버 변수 접근 한정자는 그 멤버 변수가 속해 있는 클래스로부터 객체가 생성되었을 때, 그 객체를 통해서 멤버 변수에 접근이 가능한지를 지정하는 한정자입니다.

● 일반 프로그램과 객체 지향 프로그램

- ① 일반 절차 지향 프로그램과 객체 지향 프로그램의 차이를 명확하게 알아야 합니다.
- ② 객체 지향 프로그래밍은 프로그램의 재사용에 가장 큰 이점이 있습니다.
- ③ 프로그램을 효율적으로 객체화(클래스화) 하는 것이 중요합니다.

● 클래스의 일반 구조

- ① 클래스는 객체를 생성하는 형판template입니다.
- ② 클래스는 멤버 변수, 생성자(생성자 메소드), 메소드로 구성됩니다.
- ③ 클래스는 자유롭게 구성 요소를 가질 수 있습니다.

● 클래스 선언

- ① 클래스를 선언할 때 클래스의 성격을 나타내는 한정자를 지정할 수 있습니다.
- ② 클래스의 한정자로 public/final/abstract 한정자가 있습니다.
- ③ 클래스의 한정자를 지정하지 않고 선언할 수 있습니다.

● 객체의 선언과 생성

- ① 객체를 사용하기 위해서는 객체를 선언하는 과정과 생성하는 과정이 필요합니다.
- ② 객체의 선언과 생성을 하나의 문장으로 지정할 수 있습니다.

● 멤버 변수와 메소드 변수

- ① 자바 프로그램에서 변수는 멤버 변수와 메소드의 지역 변수, 매개 변수로 구분할 수 있습니다.
- ② 변수의 형이 기본 자료형인 경우에는 값을 가지는 반면, 참조 자료형의 경우는 주소를 가집니다.
- ③ 변수에 다른 변수를 대입할 때에도, 그 변수의 형에 따라 값이 복사되거나, 주소가 복사됩니다. 주소가 복사되는 참조 자료형의 경우에는 두 개의 변수가 같은 곳을 가리키게 됩니다.
- ④ 클래스 변수는 생성된 객체들이 공유하는 전역 변수의 개념입니다.
- ⑤ 종단 변수는 상숫값을 가지는 변수로서 생성될 때 초기화되고 나면, 그 값이 변할 수 없습니다.

● 변수의 유효 범위

- ① 변수는 선언된 위치에 따라 유효 범위가 결정됩니다.
- ② 멤버 변수는 클래스 전체를 유효 범위로 가집니다.
- ③ 블록에서 선언된 변수는 그 블록을 유효 범위로 가집니다.
- ④ 메소드 매개 변수는 메소드 전체를 유효 범위로 가집니다.

● 멤버 변수 접근 한정자

- ① 자바는 멤버 변수 접근 한정자를 제공하여 객체 지향의 주요 특성인 캡슐화와 정보 은폐를 제공합니다.
- ② `public`은 접근에 제한이 없는 한정자입니다. 자바 프로그램에서는 정보 은폐를 위하여 가능하면 `public`의 사용을 최소화해야 합니다.
- ③ `private` 한정자는 그 클래스 내부에서만 사용할 수 있는 한정자입니다.
- ④ 한정자를 지정하지 않고 사용하는 경우는 같은 패키지에 속한 클래스에서는 제한 없이 사용할 수 있습니다.