

컴퓨터 프로그래밍 (Computer Programming)

이 선 순



11. 문자열과 메소드



목차

1. 문자열
2. 메소드
3. 지역변수와 전역변수
4. 메소드의 반환 값과 매개변수

03

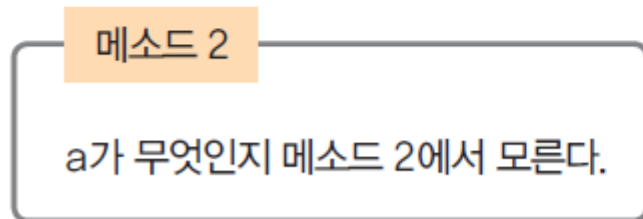
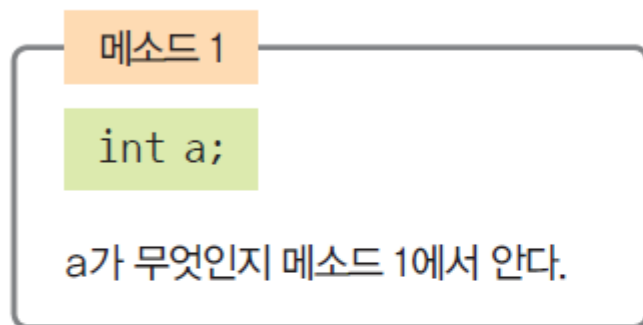
지역변수와 전역변수

03 지역변수와 전역변수

■ 지역변수와 전역변수

- 지역(local) 변수 : 한정된 지역, 메소드 안에서만 사용되는 변수
- 전역(global) 변수 : 메소드 밖, 프로그램 전체에서 사용되는 변수

① 지역변수의 생존 범위



② 전역변수의 생존 범위

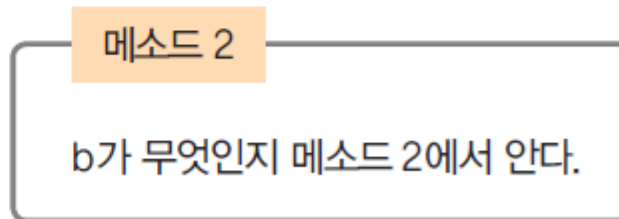
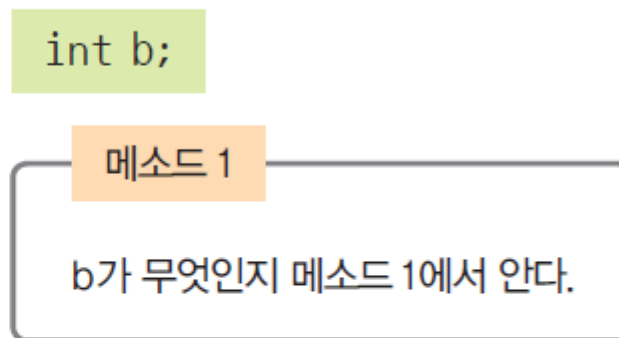


그림 9-21 지역변수와 전역변수의 생존 범위

03 지역변수와 전역변수

■ 지역변수와 전역변수

■ 지역변수와 전역변수의 공존

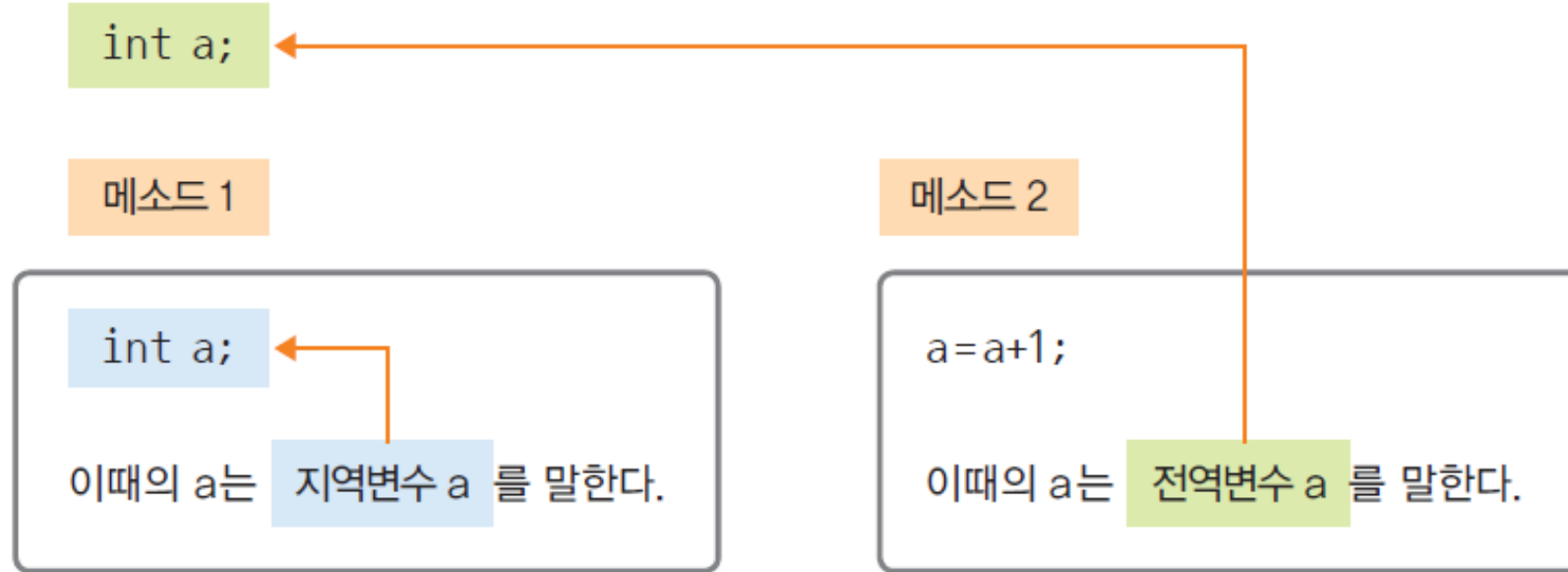


그림 9-22 지역변수와 전역변수의 공존

03 지역변수와 전역변수

■ 인스턴스 변수, 클래스 변수, 전역변수

- 전역변수에 해당하는 변수를 JAVA에서는 주로 인스턴스 변수(instance variable) 또는 클래스 변수(class variable)라고 부른다. 아직 클래스에 대해 배우지 않았으니 이를 구분하기 어려울 것이다. 그러므로 이 장에서는 JAVA의 클래스 변수를 다른 프로그래밍 언어와 공통되게 전역변수라고 부를 것이다.
- 참고로 클래스 변수 앞에는 항상 static 키워드가 붙는데, 지금은 JAVA의 전역변수 앞에 static이 붙는다고만 알아두자.

03 지역변수와 전역변수

■ 지역변수와 전역변수의 비교

■ [실습 9-15]

```
1 public class Ex09_15 {  
2  
3     static int a = 100;  
4  
5  
6     static void func1() {  
7  
8         int a = 200;  
9  
10        System.out.printf("func1()에서 a의 값==> %d\n", a);  
11    }  
12  
13  
14    public static void main(String[] args) {  
15  
16        func1();  
17  
18        System.out.printf("main() 에서 a의 값==> %d\n", a);  
19    }  
20 }
```

전역변수 **a**를 선언하고 초깃값을 대입한다
(**static**은 항상 전역변수 앞에 붙는다고 가정한다)

지역변수 **a**를 선언하고 초깃값을 대입한다
지역변수를 출력한다

메소드를 호출한다

전역변수 **a**를 선언하고 초깃값을 대입한다

Problems @ Javadoc Declaration Console

<terminated> Ex09_15 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.exe

func1()에서 a의 값==> 200
main() 에서 a의 값==> 100

04

메소드의 반환 값과 매개변수

04 메소드의 반환 값과 매개변수

■ 반환 값

- 메소드 내부에서 어떠한 처리 과정을 거친 뒤에는 값을 돌려주게 되어 있는데 이것을 반환 값이라 하고, 메소드에 전달 되는 값은 매개변수
- **TIP** : 반환 값은 return 문에 의해 반환되므로 '리턴 값'. 매개변수(parameter)는 영문 그대로 '파라미터'

04 메소드의 반환 값과 매개변수

■ 반환 값 유무에 따른 메소드 구분

■ 반환 값이 있는 메소드

- 메소드를 실행한 다음 나온 결과값은 메소드의 데이터형을 따름
- int 메소드이름() : 메소드의 결과 정수형 변수나 정수형 값 반환
- 'return()정수형 변수', 또는 'return 정수'로 표현해야 함
- func1() : int형이므로 return()문에서 정수형 변수 result를 반환
- func1()이 float형 메소드라면 return문에서 실수형 반환
- Char형 메소드라면 return문에서 문자형을 반환

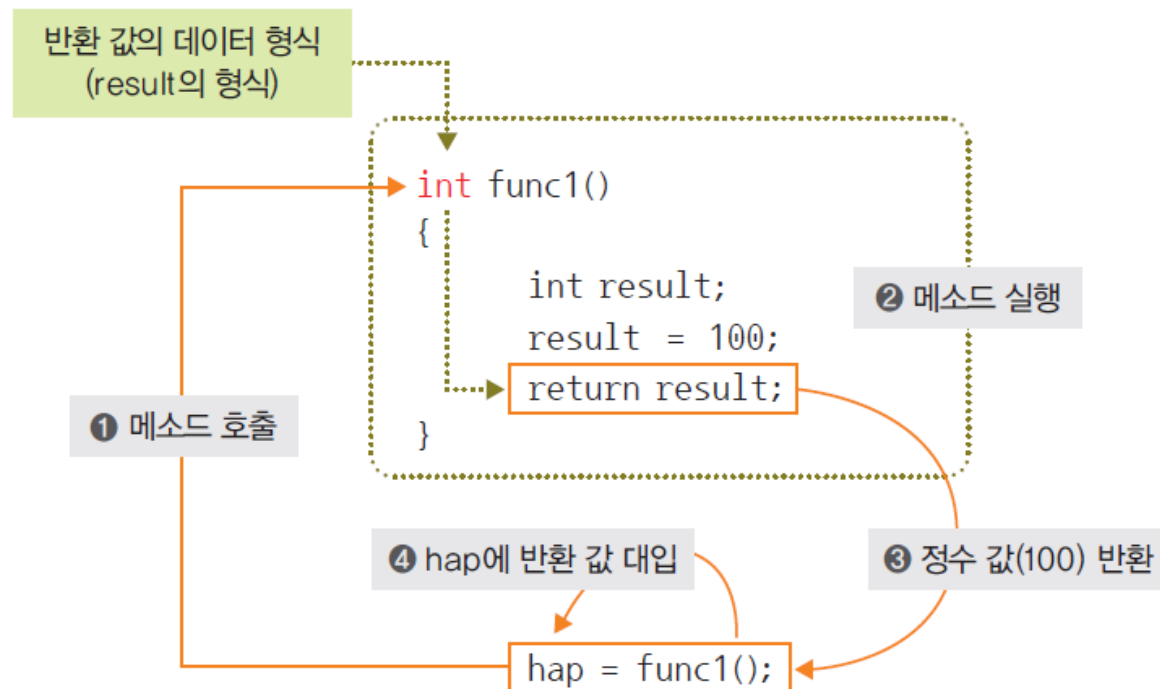


그림 9-24 int형 값의 반환

04 메소드의 반환 값과 매개변수

■ 반환 값 유무에 따른 메소드 구분

■ 반환 값이 없는 메소드

- 메소드를 실행한 결과 돌려줄 값이 없는 경우에는 메소드의 데이터형을 void (무치형) 으로 함

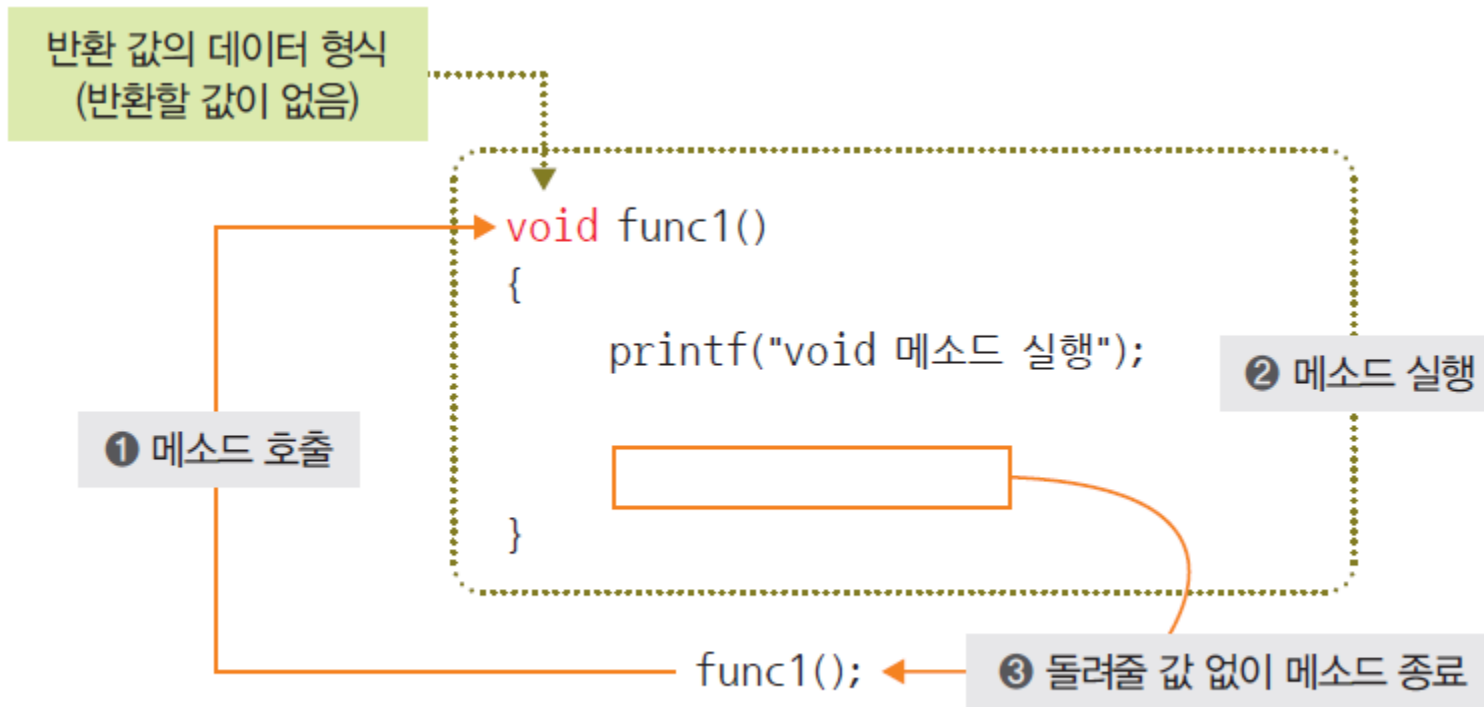


그림 9-25 void형 메소드의 작동

04 메소드의 반환 값과 매개변수

■ main () 메소드의 반환 값

- 반환 값이 없는 메소드
- 지금까지 `void main ()`으로 메인 메소드도 `void`형으로 선언했다. `main ()` 메소드의 끝이 프로그램의 끝이기 때문에 `return` 문을 사용하지 않아도 별 문제가 없었던 것이다.

04 메소드의 반환 값과 매개변수

■ 반환값 유무에 따른 메소드 비교

■ 실습9-16

```
1 public class Ex09_16 {  
2  
3     static void func1() {  
4         System.out.printf("void 형 메소드는 돌려줄게 없음.\n");  
5     }  
6  
7     static int func2() {  
8         return 100;  
9     }  
10  
11     public static void main(String[] args) {  
12         int a;  
13  
14         func1();  
15  
16         a = func2();  
17  
18         System.out.printf("int 형 메소드에서 돌려준 값 ==> %d\n", a);  
19     }  
20 }
```

void형 메소드로서 반환값이 없다

int형 메소드로서 반환값이 있다

void형 메소드를 호출한다

int형 메소드를 호출한다

Problems @ Javadoc Declaration Console

<terminated> Ex09_16 [Java Application] C:\Program Files\Java\jdk-11

void 형 메소드는 돌려줄게 없음.
int 형 메소드에서 돌려준 값 ==> 100

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법

- 매개변수를 전달할때는 '값의 전달(call by value)'방법과 '주소의 전달(call by reference)' 방법 사용
- 값의 전달(call by value) – 값 자체를 메소드에 넘겨주는 방법
- 주소의 전달(call by reference) – 주소 값(address)을 메소드에 넘겨주는 방법

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법

- 값의 전달(call by value) – 값 자체를 메소드에 넘겨주는 방법
- 실습 9-17 매개변수 전달방법 : 값의 전달

```
1 public class Ex09_17 {  
2  
3     static void func1(int a) {  
4  
5         a = a + 1;  
6         System.out.printf("전달받은 a ==> %d\n", a);  
7     }  
8  
9     public static void main(String[] args) {  
10  
11         int a = 10;    지역변수 a를 선언한다  
12  
13         func1(a);      a값을 매개변수로 넘겨 메소드를 호출한다  
14  
15         System.out.printf("func1() 실행 후의 a ==> %d\n", a);  
16  
17     }  
18 }
```

전달받은 a값을 1증가시킨 후 출력한다

메소드를 호출한 다음 a값을 출력한다

Problems @ Javadoc Declaration Console

<terminated> Ex09_17 [Java Application] C:\Program Files\Java\jdk-1

전달받은 a ==> 11
func1() 실행 후의 a ==> 10

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법

- 값의 전달(call by value) – 값 자체를 메소드에 넘겨주는 방법
- 실습 9-17

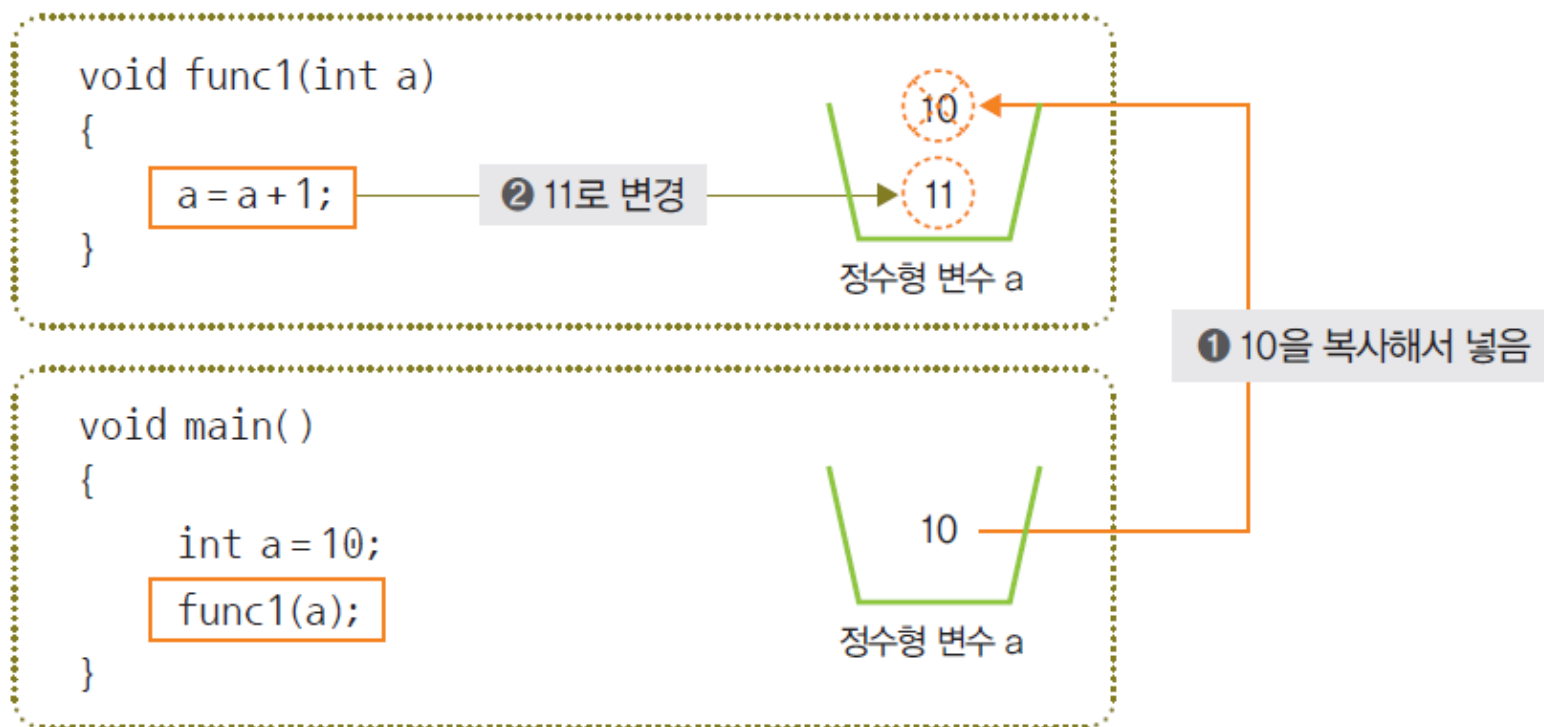


그림 9-28 매개변수 전달 : 값의 전달

04 메소드의 반환 값과 매개변수

■ 클래스 객체 만들기

- 참조의 전달을 위해 간단하게 만드는 방법
- 아래와 같이 만들면 int형처럼 myInt라는 새로운 형식이 생긴다. 그리고 클래스 객체 m을 생성했다.
- m에는 value라는 인스턴스 변수가 존재하고 이 변수를 m.value로 접근할 수 있다.

```
// 클래스 선언
class myInt {
    int value;
}

// 클래스 사용
myInt m = new myInt();
m.value = 10;
```

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법

- 주소(또는 참조)의 전달(call by reference) – 주소 값(address)을 메소드에 넘겨주는 방법
- 실습 9-18

```
1 class myInt {  
2     int a;  
3 }  
4  
5  
6 public class Ex09_18 {  
7  
8     static void func1(myInt m) {  
9  
10         m.a = m.a + 1;  
11  
12         System.out.printf("전달받은 a ==> %d\n", m.a);  
13     }  
14  
15     public static void main(String[] args) {  
16  
17         myInt m = new myInt();  
18  
19         m.a = 10;  
20  
21         func1(m);  
22  
23         System.out.printf("func1() 실행 후의 a ==> %d\n", m.a);  
24     }  
25 }  
26 }
```

myInt클래스를 선언한다

클래스 객체의 정수형 변수 a에 1을 대입한다

a값을 출력한다

클래스 객체의 정수형 변수 a를 10으로 초기화한다

메소드 호출시 클래스 객체 m을 전달한다

메소드 호출 후의 a값을 출력한다

Problems @ Javadoc Declaration Console

<terminated> Ex09_18 [Java Application] C:\Program Files\Java\j

전달받은 a ==> 11
func1() 실행 후의 a ==> 11

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법

- 주소(또는 참조)의 전달(call by reference) – 주소 값(address)을 메소드에 넘겨주는 방법
- 실습 9-18

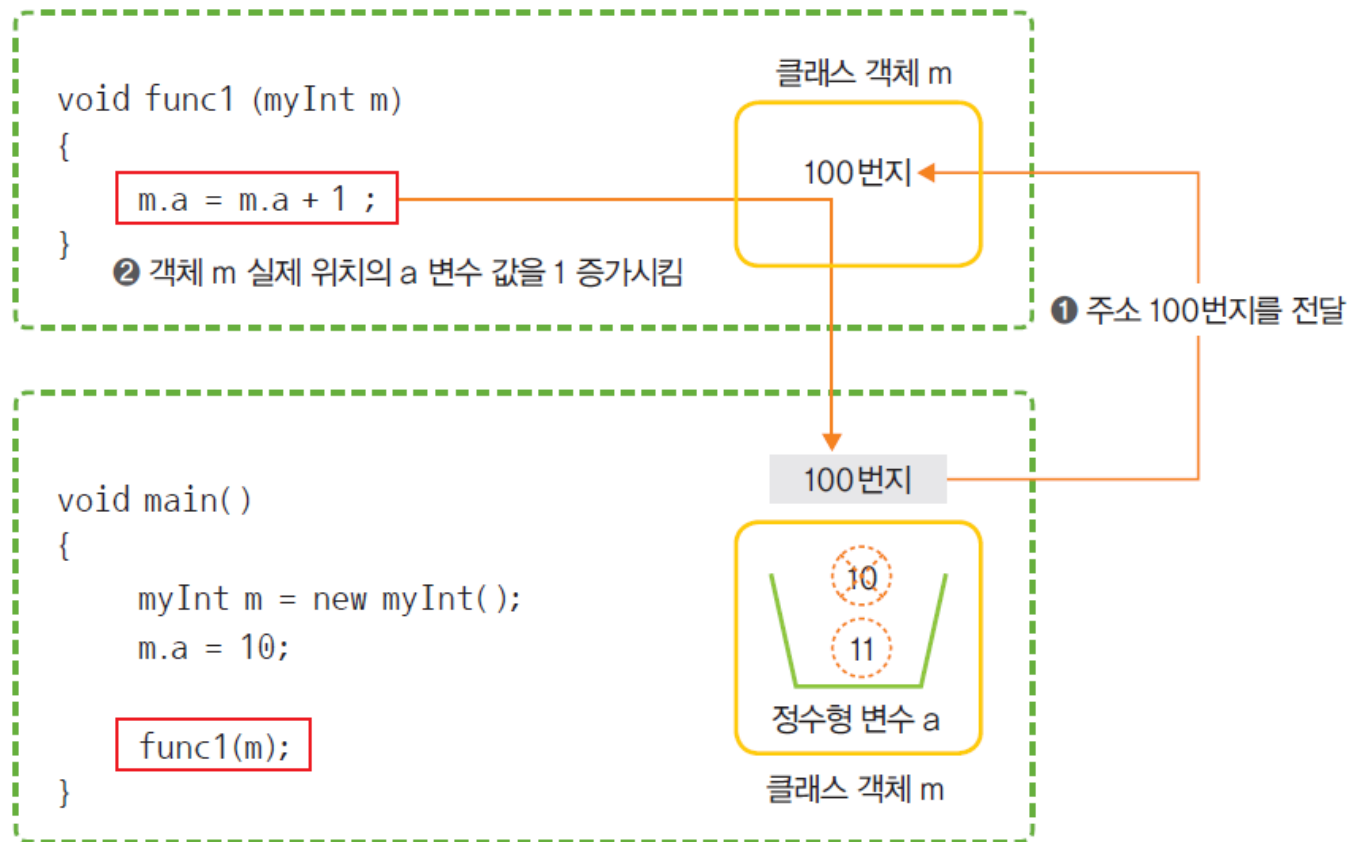


그림 9-30 매개변수 전달 : 주소의 전달

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법 비교

- 두 문자를 교환하는 실습을 통해 값을 전달하는 방법과 주소를 전달하는 방법을 구분
- 실습 9-19

```
1 class myChar {  
2     char x;  
3     char y;  
4 }  
5  
6 public class Ex09_19 {  
7  
8     static void func1(char x, char y) {  
9         char imsi;  
10        imsi = x;  
11        x = y;  
12        y = imsi;  
13    }  
14  
15    static void func2(myChar ch) {  
16        char imsi;  
17        imsi = ch.x;  
18        ch.x = ch.y;  
19        ch.y = imsi;  
20    }  
21 }
```

2개의 문자형 변수를 가진 myChar 클래스를 선언한다

두 문자를 교환한다. 매개변수가 값인 메소드

두 문자를 교환한다. 매개변수가 주소인 메소드

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법 비교

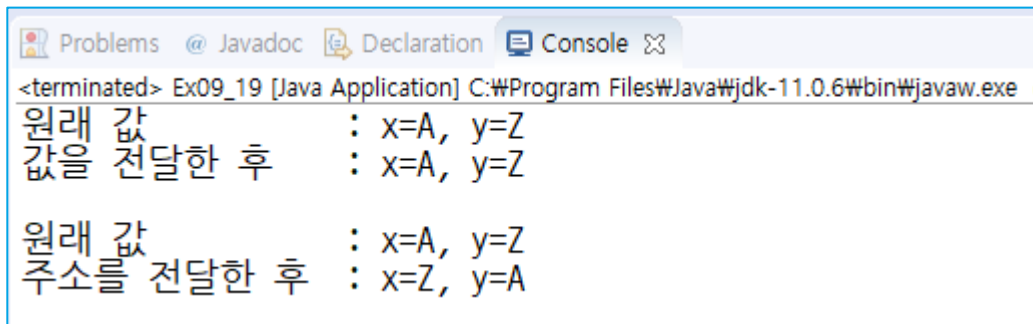
- 두 문자를 교환하는 실습을 통해 값을 전달하는 방법과 주소를 전달하는 방법을 구분
- 실습 9-19

```
22 public static void main(String[] args) {  
23  
24     char x = 'A', y = 'Z';  
25     System.out.printf("원래 값      : x=%c, y=%c\n", x, y);  원래 문자를 출력한다.  
26  
27     func1(x, y);      값을 전달하여 func1() 메소드를 호출한다  
28  
29     System.out.printf("값을 전달한 후   : x=%c, y=%c\n\n", x, y); func1() 메소드를 호출한 후 문자를 출력한다.  
30  
31     myChar ch = new myChar();  
32     ch.x = 'A';  
33     ch.y = 'Z';  
34     System.out.printf("원래 값      : x=%c, y=%c\n", ch.x, ch.y);  원래 문자를 출력한다.  
35  
36     func2(ch);        주소를 전달하여 func2() 메소드를 호출한다  
37  
38     System.out.printf("주소를 전달한 후  : x=%c, y=%c\n", ch.x, ch.y); func2() 메소드를 호출한 후 문자를 출력한다.  
39 }  
40 }
```

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법 비교

- 두 문자를 교환하는 실습을 통해 값을 전달하는 방법과 주소를 전달하는 방법을 구분
- 실습 9-19



```
<terminated> Ex09_19 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.exe
원래 값      : x=A, y=Z
값을 전달한 후  : x=A, y=Z

원래 값      : x=A, y=Z
주소를 전달한 후  : x=Z, y=A
```

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법 비교

- 두 문자를 교환하는 실습을 통해 값을 전달하는 방법과 주소를 전달하는 방법을 구분
- 실습 9-19
- 27행

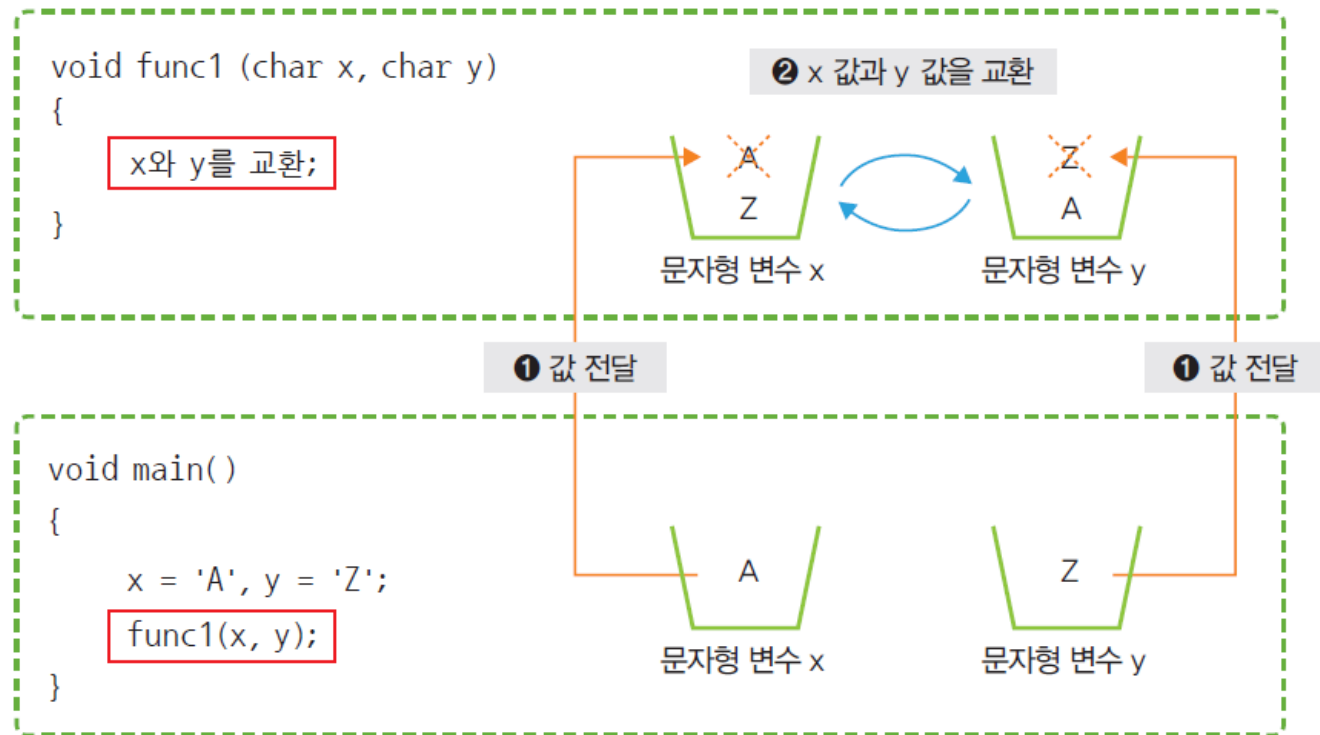


그림 9-32 값의 전달을 통한 교환

04 메소드의 반환 값과 매개변수

■ 매개변수 전달방법 비교

- 두 문자를 교환하는 실습을 통해 값을 전달하는 방법과 주소를 전달하는 방법을 구분
- 실습 9-19
- 36행

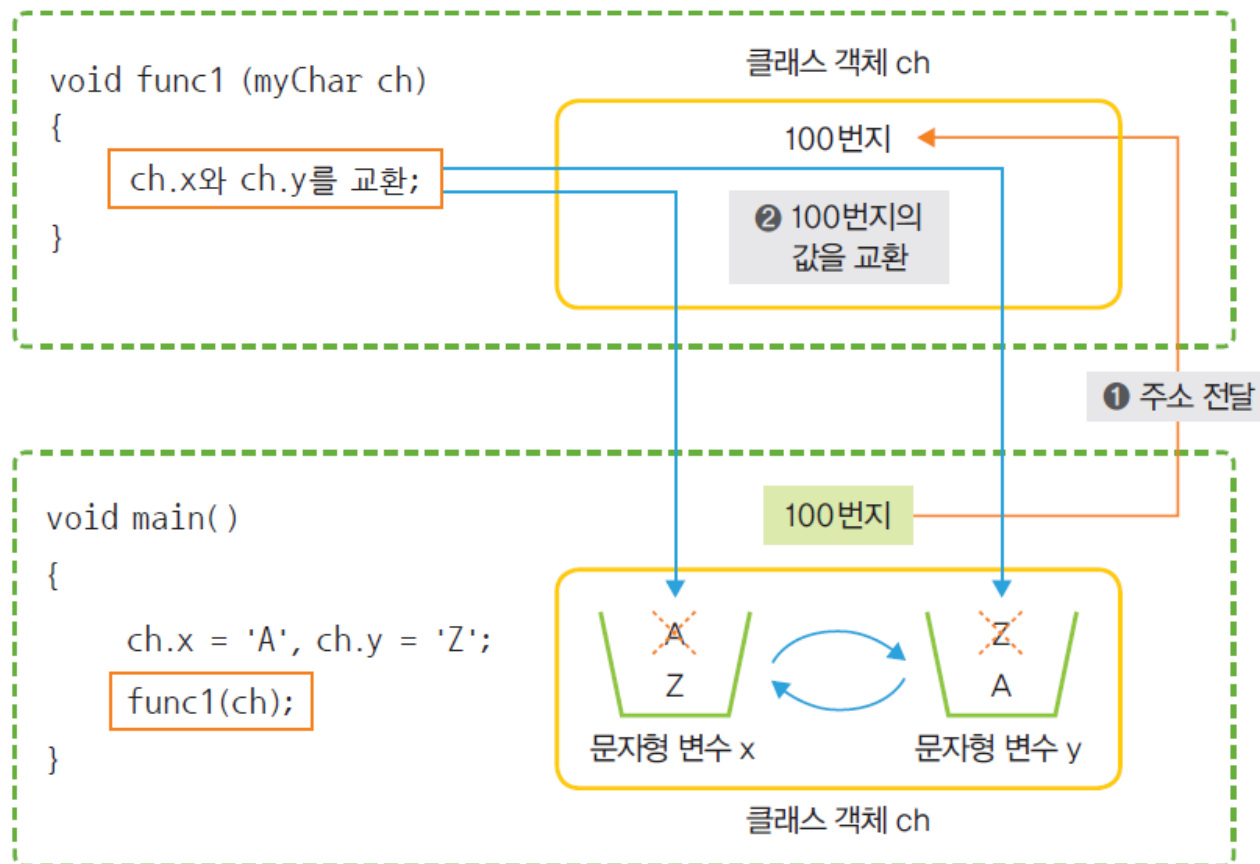


그림 9-33 주소의 전달을 통한 교환

질문은 이메일을 이용해주세요.
ds.june2@gmail.com

감사합니다