

컴퓨터 프로그래밍 (Computer Programming)

이 선 순



11. 문자열과 메소드



목차

1. 문자열

2. 메소드

3. 지역변수와 전역변수

4. 메소드의 반환값과 매개변수

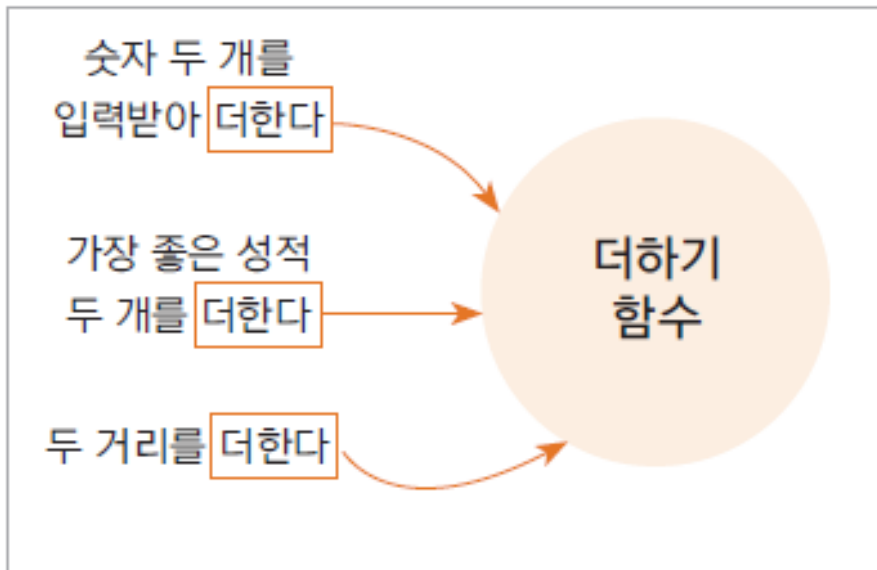
02

메소드

02 메소드

■ 메소드의 개념

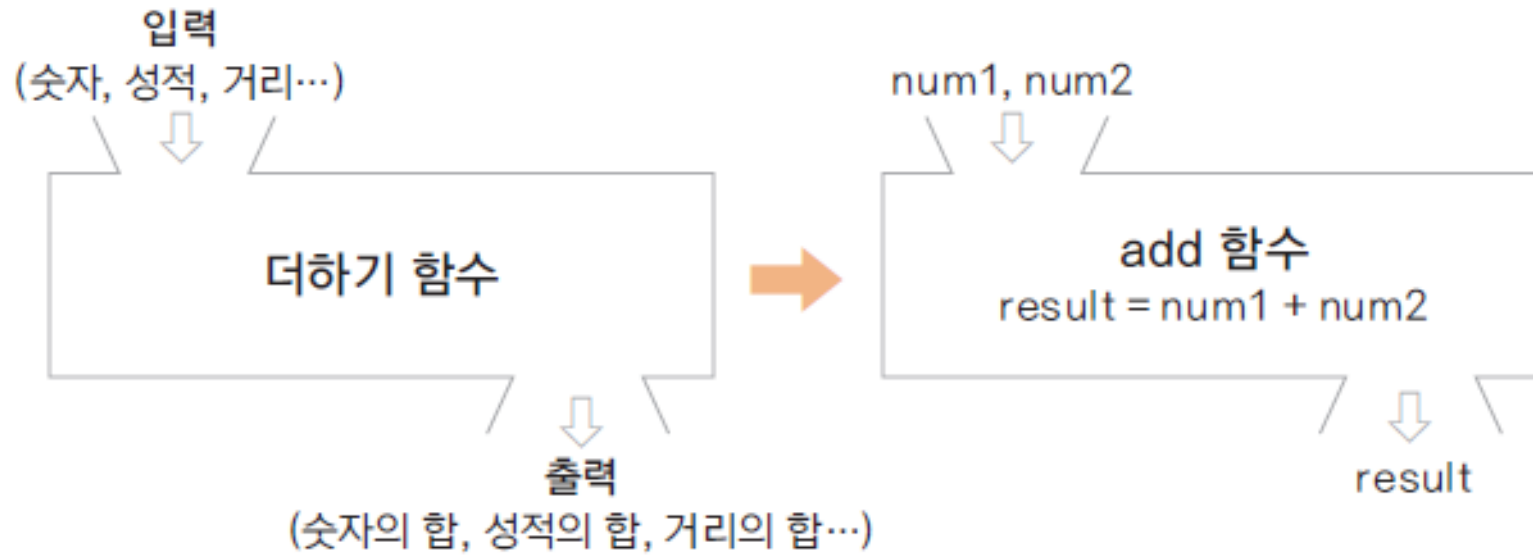
- 함수(function)의 한 종류
- 어떤 것을 넣으면 무언가를 돌려주는 요술 상자. 메소드는 JAVA 프로그램 자체에서 제공하지만 사용자가 직접 만들어서 사용하기도 함
- 객체의 기능을 제공하기 위해 클래스 내부에 구현되는 함수
- 다른 언어에서는 메소드를 함수라 함. 클래스 안에 존재하는 함수를 메소드라고 하는데, JAVA의 함수는 무조건 클래스 안에 존재하기 때문에 결국 모든 함수가 메소드임
- 메소드(함수)는 하나의 기능을 수행하는 일련의 코드로 중복되는 기능은 함수로 구현하여 함수를 호출하여 사용함



02 메소드

■ 메소드의 개념

- 메소드(함수)는 이름이 있고 입력값과 결과값을 가짐



- 메소드에 이름을 붙일때는 의미를 알 수 있는 단어를 사용하는 것이 좋음

02 메소드

■ 메소드의 개념

- JAVA에서 제공하는 메소드 사용

```
메소드이름( );
```

- 가장 많이 사용해온 메소드 `System.out.printf()`

```
System.out.printf("Basic-Java");
```

■ 메소드의 개념

- 예. 직접 커피를 타는 과정

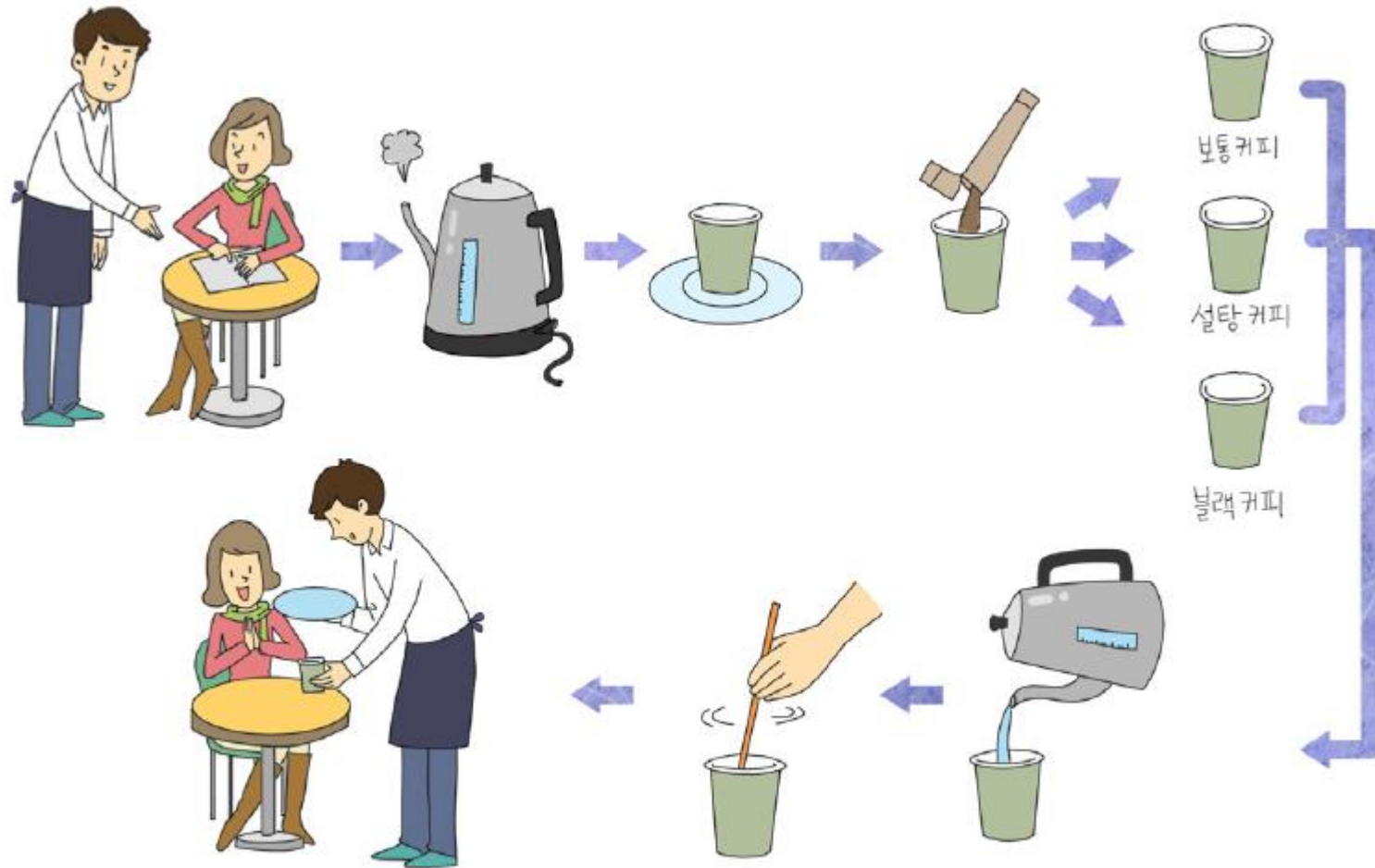


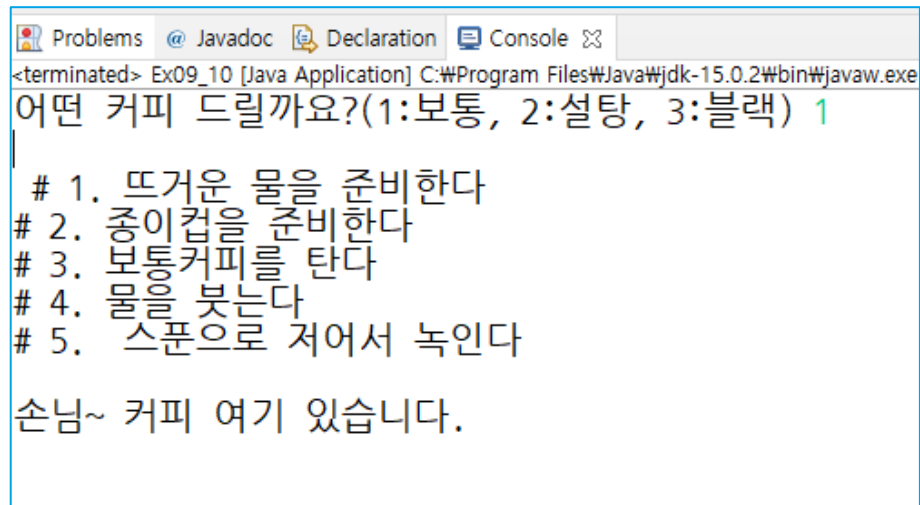
그림 9-11 직접 커피를 타는 과정

02 메소드

■ 메소드의 개념

■ [실습9-10] 직접 커피를 타는 과정

```
1 import java.util.Scanner;
2
3 public class Ex09_10 {
4     public static void main(String[] args) {
5         Scanner s = new Scanner(System.in);
6         int coffee; // 커피 종류 변수 선언
7
8         System.out.printf("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) ");
9         coffee = s.nextInt(); // 커피를 선택
10
11         System.out.printf("\n # 1. 뜨거운 물을 준비한다\n");
12         System.out.printf("# 2. 종이컵을 준비한다\n");
13
14         switch (coffee) { //커피의 종류에 따라 안내문을 출력
15             case 1:
16                 System.out.printf("# 3. 보통커피를 탄다\n"); break;
17             case 2:
18                 System.out.printf("# 3. 설탕커피를 탄다\n"); break;
19             case 3:
20                 System.out.printf("# 3. 블랙커피를 탄다\n"); break;
21             default:
22                 System.out.printf("# 3. 아무거나 탄다\n"); break;
23         }
24
25         System.out.printf("# 4. 물을 붓는다\n");
26         System.out.printf("# 5. 스푼으로 저어서 녹인다\n\n");
27
28         System.out.printf("손님~ 커피 여기 있습니다.\n");
29
30         s.close();
31     }
32 }
```



Problems Javadoc Declaration Console

<terminated> Ex09_10 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe

어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 1

1. 뜨거운 물을 준비한다
2. 종이컵을 준비한다
3. 보통커피를 탄다
4. 물을 붓는다
5. 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.

■ 메소드의 개념

- 예. 커피자판기를 이용하여 커피를 타는 과정

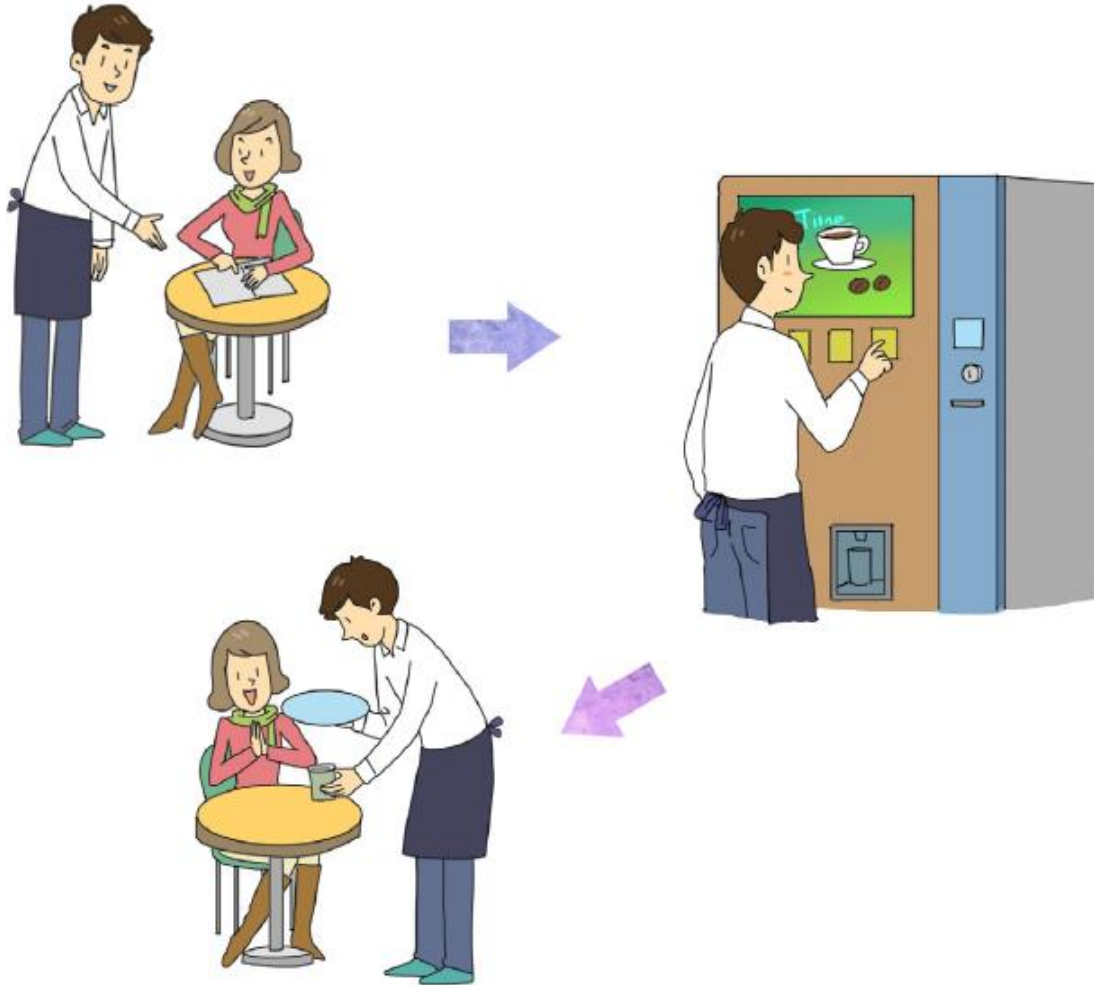


그림 9-13 커피 자판기를 사용하는 과정

02 메소드

■ 메소드의 개념

- [실습9-11] 메소드를 사용하여 [실습9-10] 변경하기(커피자판기를 이용하여 커피를 타는 과정)

```
1 import java.util.Scanner;
2
3 public class Ex09_11 {
4     static int coffee_machine(int button) {
5         System.out.printf("\n# 1.(자동으로)뜨거운 물을 준비한다\n");
6         System.out.printf("# 2.(자동으로)종이컵을 준비한다\n");
7
8         switch (button) {
9             case 1:
10                 System.out.printf("# 3.(자동으로) 보통커피를 탄다\n"); break;
11             case 2:
12                 System.out.printf("# 3.(자동으로) 설탕커피를 탄다\n"); break;
13             case 3:
14                 System.out.printf("# 3.(자동으로 ) 블랙커피를 탄다\n"); break;
15             default:
16                 System.out.printf("# 3.(자동으로) 아무거나 탄다\n"); break;
17         }
18
19         System.out.printf("# 4.(자동으로 ) 물을 붓다\n");
20         System.out.printf("# 5.(자동으로) 스푼으로 저어서 녹인다\n\n");
21
22         return 0;
23     }
24
25     public static void main(String[] args) {
26         Scanner s = new Scanner(System.in);
27         int coffee;
28         int ret;
29
30         System.out.printf("어떤 커피를 드릴까요?(1:보통, 2:설탕, 3:블랙) ");
31         coffee = s.nextInt(); //커피를 주문받는다.
32
33         ret = coffee_machine(coffee); //커피자판기의 버튼을 누른다(coffee_machine()메소드 호출)
34
35         System.out.printf("손님~ 커피 여기 있습니다.\n");
36
37         s.close();
38     }
39 }
```

커피자판기 메소드 구현

Problems @ Javadoc Declaration Console

<terminated> Ex09_11 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe

어떤 커피를 드릴까요?(1:보통, 2:설탕, 3:블랙) 1

|

1.(자동으로)뜨거운 물을 준비한다

2.(자동으로)종이컵을 준비한다

3.(자동으로) 보통커피를 탄다

4.(자동으로) 물을 붓다

5.(자동으로) 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.

■ 메소드의 개념

- [실습9-12] 여러 명의 주문을 받도록 [실습9-11] 변경하기

```
1 import java.util.Scanner;
2
3 public class Ex09_12 {
4     static int coffee_machine(int button) {
5         System.out.printf("\n# 1.(자동으로)뜨거운 물을 준비한다\n");
6         System.out.printf("# 2.(자동으로)종이컵을 준비한다\n");
7
8         switch (button) {
9             case 1:
10                 System.out.printf("# 3.(자동으로) 보통커피를 탄다\n"); break;
11             case 2:
12                 System.out.printf("# 3.(자동으로) 설탕커피를 탄다\n"); break;
13             case 3:
14                 System.out.printf("# 3.(자동으로) 블랙커피를 탄다\n"); break;
15             default:
16                 System.out.printf("# 3.(자동으로) 아무거나 탄다\n"); break;
17         }
18
19         System.out.printf("# 4.(자동으로) 물을 붓다\n");
20         System.out.printf("# 5.(자동으로) 스푼으로 저어서 녹인다\n\n");
21
22         return 0;
23     }
24 }
```

■ 메소드의 개념

- [실습9-12] 여러 명의 주문을 받도록 [실습9-11] 변경하기

```
24
25 public static void main(String[] args) {
26     Scanner s = new Scanner(System.in);
27     int coffee;
28     int ret;
29
30     System.out.printf("A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
31     coffee = s.nextInt();
32     ret = coffee_machine(coffee);
33     System.out.printf("A님 커피 여기 있습니다.\n");
34
35     System.out.printf("B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
36     coffee = s.nextInt();
37     ret = coffee_machine(coffee);
38     System.out.printf("B님 커피 여기 있습니다.\n");
39
40     System.out.printf("C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
41     coffee = s.nextInt();
42     ret = coffee_machine(coffee);
43     System.out.printf("C님 커피 여기 있습니다.\n");
44
45     s.close();
46 }
47 }
```

주문을 받고 커피자판기의 버튼을 누른다
(메소드 호출)

■ 메소드의 개념

- [실습9-12] 여러 명의 주문을 받도록 [실습9-11] 변경하기

```
Problems @ Javadoc Declaration Console
<terminated> Ex09_12 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe
A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 1
# 1.(자동으로)뜨거운 물을 준비한다
# 2.(자동으로)종이컵을 준비한다
# 3.(자동으로) 보통커피를 탄다
# 4.(자동으로) 물을 붓다
# 5.(자동으로) 스푼으로 저어서 녹인다
A님 커피 여기 있습니다.
B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2
# 1.(자동으로)뜨거운 물을 준비한다
# 2.(자동으로)종이컵을 준비한다
# 3.(자동으로) 설탕커피를 탄다
# 4.(자동으로) 물을 붓다
# 5.(자동으로) 스푼으로 저어서 녹인다
B님 커피 여기 있습니다.
C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 3
# 1.(자동으로)뜨거운 물을 준비한다
# 2.(자동으로)종이컵을 준비한다
# 3.(자동으로) 블랙커피를 탄다
# 4.(자동으로) 물을 붓다
# 5.(자동으로) 스푼으로 저어서 녹인다
C님 커피 여기 있습니다.
```

■ 메소드의 장점

- 코드의 모듈화 : 메소드를 기능별로 작성하여 필요한 기능만 조합할 수 있다.
- 코드의 간략화 : 반복되는 문장을 밖으로 빼냄으로써 JAVA 소스코드를 간결하게 만든다.
- 코드의 재사용성 : 한 번 작성한 메소드를 다시 사용할 수 있다.
- 코드의 수정 용이 : 프로그램 오류를 수정하기가 쉽다.

■ 메소드의 모양과 활용

- 메소드는 반복적으로 코딩해야 할 내용을 한번만 코딩해주고 필요할 때마다 가져다 사용할 수 있음
- 일단 메소드로 만들어 놓으면 동일한 동작을 계속 사용하므로 내부 내용이 바뀌지 않음
- 메소드는 매개변수(parameter)를 입력 받은 후 그 매개변수를 가공 및 처리하여 반환 값을 돌려줌
- 매개변수는 인수, 인자, 파라미터 등 다양하게 불린다.

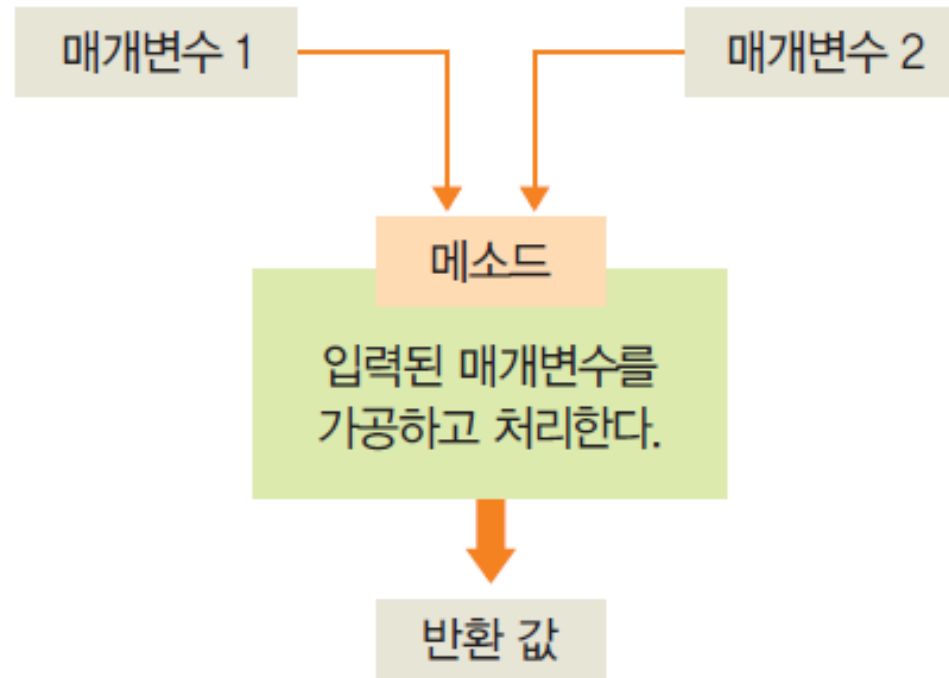


그림 9-16 메소드의 형태

02 메소드

■ 메소드 정의하기

- 메소드(함수)를 코드로 구현
- 예. 메소드의 이름, 매개변수, 반환값을 선언하고 코드를 구현함

```
④ ①           ②  
int add (int num1, int num2){  
    int result ;  
    result = num1 + num2 ;  
    return result ;  
} ③
```

- ① 메소드 이름 : 함수의 기능과 관련하여 명명
- ② 매개 변수 : 함수의 수행을 위해 필요한 변수
- ③ return : 메소드 수행 결과를 반환하기 위한 예약어
- ④ 함수 반환 형 : 반환 값의 자료형을 나타냄
반환 값이 없는 경우 void라고 씀

- **int** : 함수 반환형
- **add** : 메소드 이름, ① 위치에 작성
 - 프로그래머가 임의로 작성, 함수기능과 관련있게 만들어야 나중에 호출하거나 이해하기 좋음
- **int num1, int num2** : ②의 num1과 num2를 매개변수라 함. 함수 내부에서 사용할 괄호 안의 변수
- **return** : return 예약어

■ 메소드 정의하기

■ return 예약어와 반환 값의 데이터형

- `return()` : 메소드의 결과값을 반환해주는 예약어

```
④ ①           ②  
int add (int num1, int num2){  
    int result ;  
    result = num1 + num2 ;  
    return result ;  
} ③
```

- `add()` 메소드를 수행한 후 결과값은 `result`에 저장됨
- `result()`에 저장된 결과 값은 메소드를 호출했을 때 반환되는 값이므로 '반환 값'이라고도 부름
- ③ 의 예약어를 사용하면 정수형 `result` 값을 반환한다.
- 반환 값의 자료형을 반환형이라 하고 ④ 위치에 `int` 라고 선언

■ 메소드 정의하기

■ return의 다른 역할

- 메소드 수행을 끝내고 프로그램 흐름 중에서 호출한 곳으로 다시 되돌아갈 수도 있음

```
void divide(int num1, int num2){  
    if (num2==0) {  
        System.out.println("나누는 수는 0이 될 수 없습니다");  
  
        return ; //메소드 수행 종료  
    }  
    else {  
        int result = num1 /num2 ;  
        System.out.println(num1 + "/" + num2 + "=" + result + "입니다.");  
    }  
}
```

- divide() 메소드: 두 수를 매개변수로 전달받아서 나눗셈을 한 후 몫을 출력하는 메소드
- 만약 나누는 수가 0이라면 수행이 안될 것임. 이 경우에 함수 수행을 종료하는 예약어 return()을 사용함
- **함수 수행을 종료하는 목적이므로** return뒤에 반환값을 적지 않아도 됨.

02 메소드

■ 두 정수를 입력받아 두 정수의 합계를 반환하는 plus()메소드 만들기

- [실습 9-13] 본격적으로 메소드 사용하기

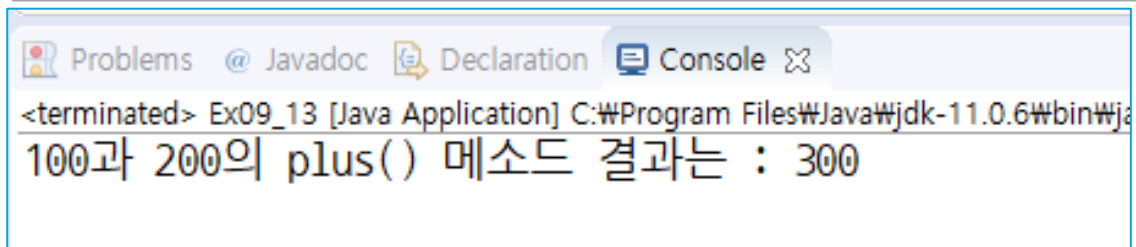
```
1 public class Ex09_13 {  
2  
3     static int plus(int v1, int v2) {  
4         int result;  
5         result = v1 + v2;  
6         return result;  
7     }  
8  
9     public static void main(String[] args) {  
10         int hap;  
11         hap = plus(100, 200);  
12         System.out.printf("100과 200의 plus() 메소드 결과는 : %d\n", hap);  
13     }  
14 }
```

plus() 메소드를 정의

3행에서 받은 두 매개변수의 합을 구한다

plus() 메소드를 호출한 곳에 result값을 반환

매개변수 2개를 지정해서 plus()메소드를 호출하고 반환 값은 hap에 저장한다.



The screenshot shows the IDE's console window with the following content:

```
<terminated> Ex09_13 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\java.exe  
100과 200의 plus() 메소드 결과는 : 300
```

02 메소드

■ 두 정수를 입력받아 두 정수의 합계를 반환하는 plus()메소드 만들기

■ [실습 9-13] 본격적으로 메소드 사용하기

- 11행 : 두 정수를 입력받아 두 정수의 합계를 반환하는 plus() 메소드

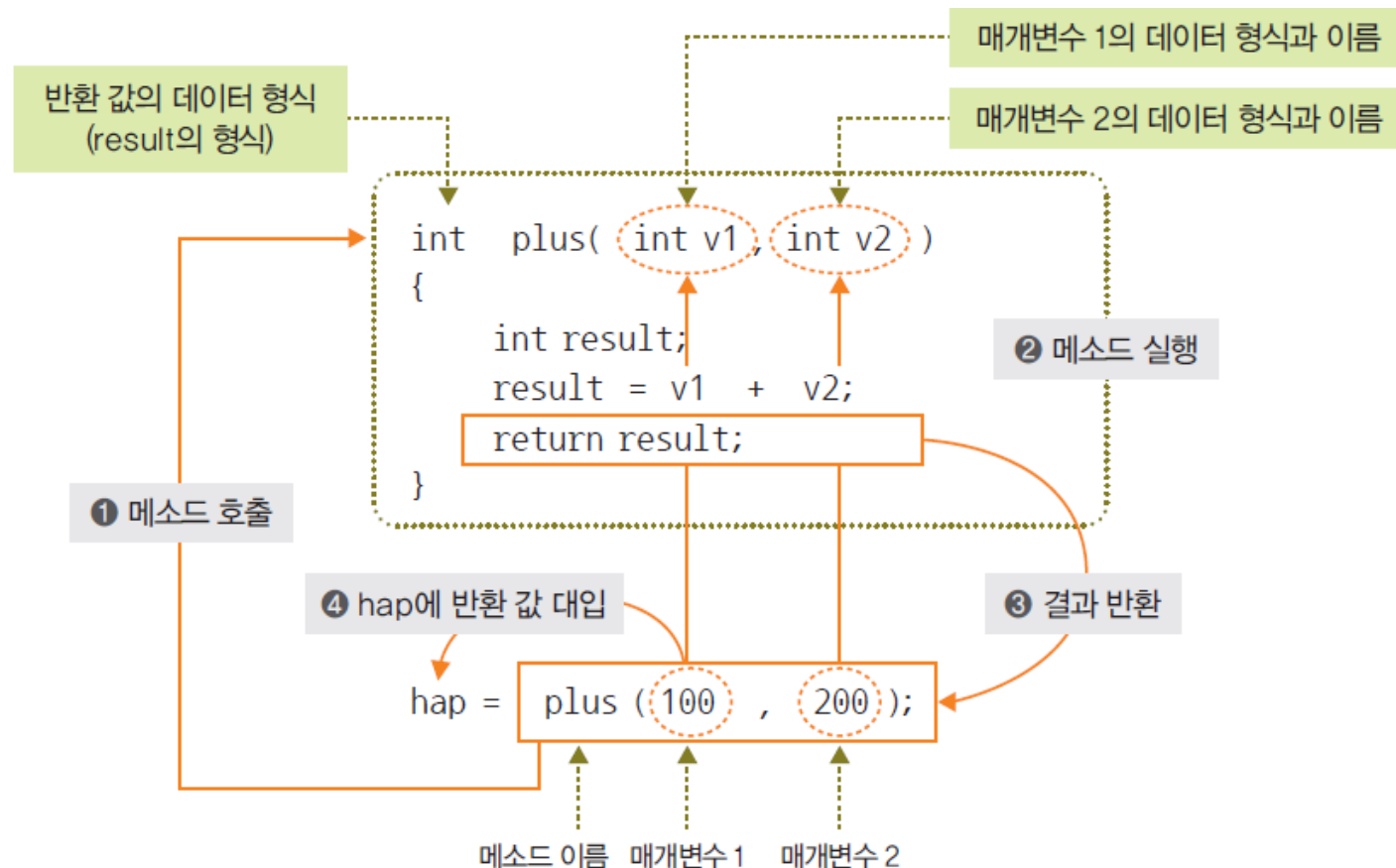


그림 9-18 plus() 메소드의 형태와 호출 순서

02 메소드

■ 두 정수를 입력받아 두 정수의 합계를 반환하는 plus()메소드 만들기

- [실습 9-13] 본격적으로 메소드 사용하기
 - 간단하게 표현한 plus() 메소드의 호출

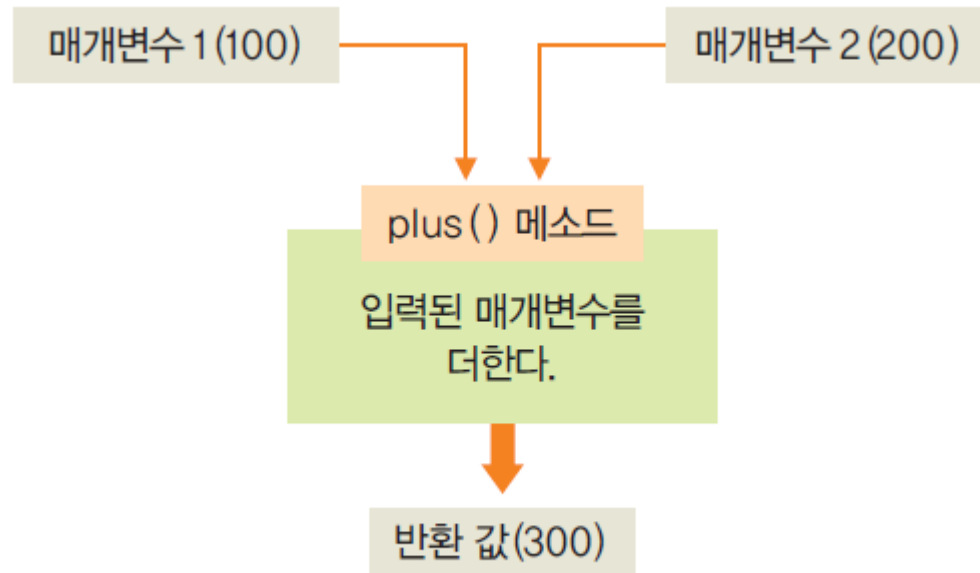


그림 9-19 간단하게 표현한 plus() 메소드의 호출

02 메소드

■ 입력한 두 숫자의 사칙연산을 하는 계산기 메소드

■ [실습 9-14] 계산기 메소드 사용

```
1 import java.util.Scanner;
2 public class Ex09_14 {
3
4     static int calc(int v1, int v2, int op) {
5         int result;
6
7         switch (op) {
8             case 1: result = v1 + v2; break;
9             case 2: result = v1 - v2; break;
10            case 3: result = v1 * v2; break;
11            case 4: result = v1 / v2; break;
12            default: result = 0;
13        }
14
15        return result; 계산결과를 반환한다
16    }
17 }
```

매개변수 3개를 받아서
계산하는 메소드

매개변수 값에 따라서 실행

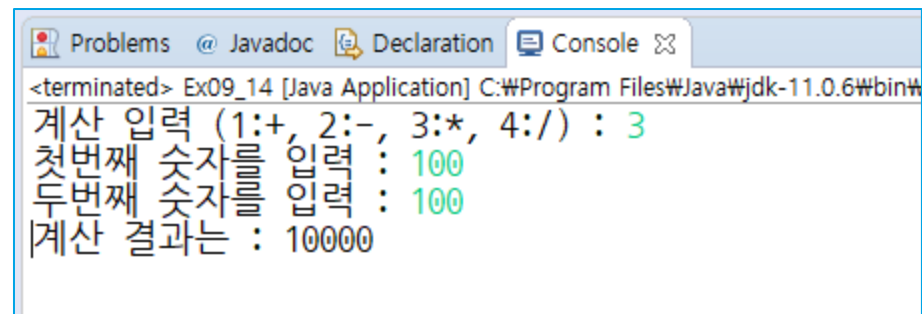
1: 덧셈, 2: 뺄셈, 3: 곱셈, 4: 나눗셈

02 메소드

■ 입력한 두 숫자의 사칙연산을 하는 계산기 메소드

■ [실습 9-14] 계산기 메소드 사용

```
18 public static void main(String[] args) {  
19     Scanner s = new Scanner(System.in);  
20  
21     int res;  
22     int oper, a, b; 계산결과, 연산자, 입력 숫자 2개에 대한 변수를 선언한다.  
23  
24     System.out.printf("계산 입력 (1:+, 2:-, 3:*, 4:/) : ");  
25  
26     oper = s.nextInt(); 연산자를 입력한다.  
27  
28     System.out.printf("첫번째 숫자를 입력 : ");  
29  
30     a = s.nextInt(); 계산할 숫자를 입력한다.  
31  
32     System.out.printf("두번째 숫자를 입력 : ");  
33  
34     b = s.nextInt(); 계산할 숫자를 입력한다.  
35  
36     res = calc(a, b, oper); 매개변수 3개를 넣고 calc() 메소드를 호출한다. 결과는 res에 저장한다.  
37  
38     System.out.printf("계산 결과는 : %d\n", res);  
39  
40     s.close();  
41 }  
42 }
```



```
<terminated> Ex09_14 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\
계산 입력 (1:+, 2:-, 3:*, 4:/) : 3
첫번째 숫자를 입력 : 100
두번째 숫자를 입력 : 100
계산 결과는 : 10000
```


질문은 이메일을 이용해주세요.
ds.june2@gmail.com

감사합니다