

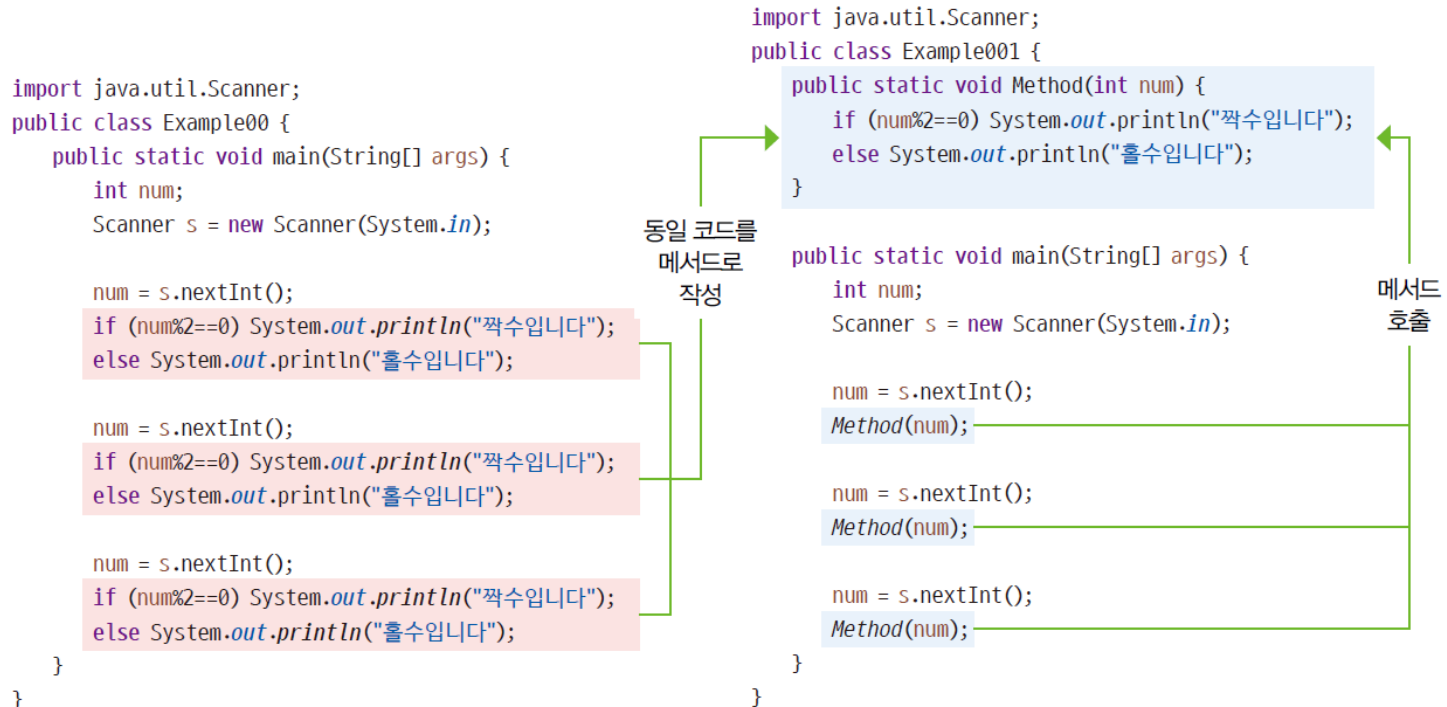
# Section 01

메서드

# 1. 메서드

## ■ 메서드(method)

- 프로그램에서 특정 작업을 수행하기 위한 코드의 집합을 말함
- [예] System.out.println() 메서드 : 콘솔에 메시지를 출력하는 기능의 함수



(a) 같은 코드를 반복하는 경우

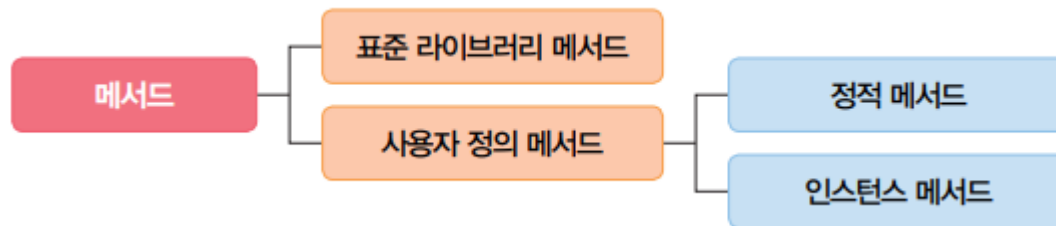
(b) 메서드를 만드는 경우

[그림 5-1] 짝수, 홀수를 판별하는 코드의 반복 구현 방법

# 1. 메서드

## ■ 메서드의 유형

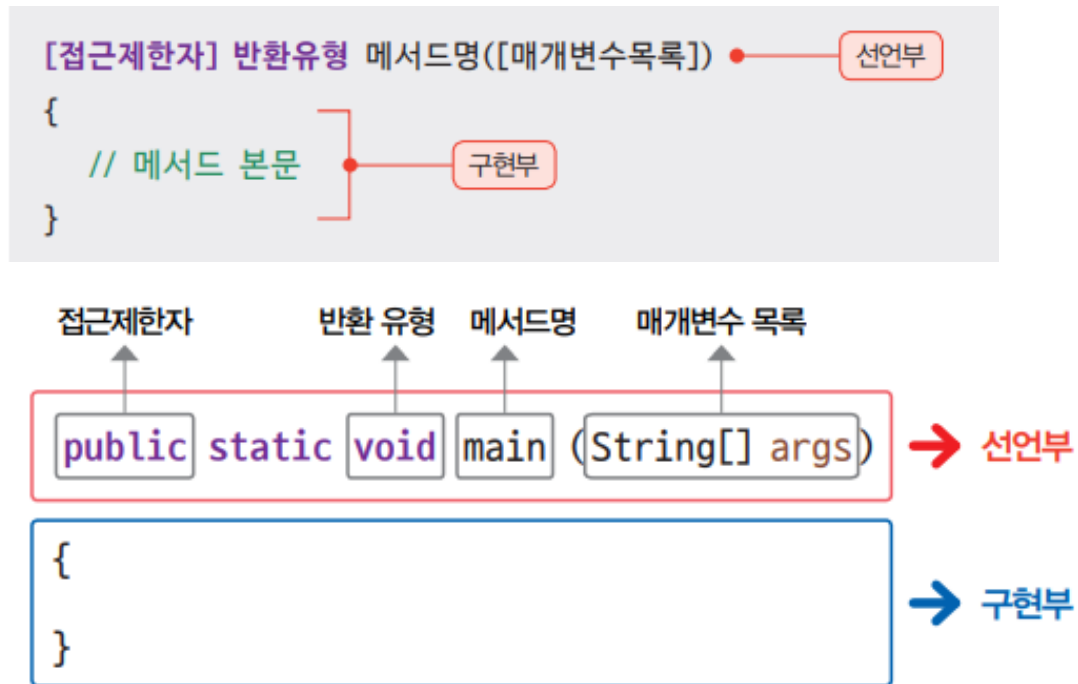
- 표준 라이브러리 메서드
  - 자바 클래스 라이브러리에 이미 정의되어 있는 메서드
  - 사전 정의 메서드 또는 내장 메서드라고도 함
  - 언제든지 프로그램에서 호출하기만 하면 사용할 수 있음
- 사용자 정의 메서드
  - 사용자 또는 프로그래머가 작성한 메서드
  - 필요에 따라 추가·보완·수정·삭제할 수 있음



[그림 5-3] 메서드의 유형

# 1. 메서드

## ■ 메서드의 기본 구조



[그림 5-4] 메서드의 기본 구조: main() 메서드

# 1. 메서드

## ■ 메서드 호출

- 메서드명 뒤의 괄호 안에 인수(매개변수 목록)가 있는 경우  
→ 인수를 사용하여 메서드를 호출

```
메서드명();  
add();  
getName();  
getAddressDetails();
```

### 메서드 선언 및 호출 예시

```
public class Example01 {  
    public static void method() {  
        System.out.println("static 메서드입니다.");  
        System.out.println(5 + 6);  
    }  
    public static void main(String[] args) {  
        method();  
    }  
}
```

#### 실행 결과

```
static 메서드입니다.  
11
```

# 1. 메서드

## ■ 메서드 호출

- 메서드 호출 예제

```
public class Example01 {  
    public static void main(String[] args) {  
        System.out.println("static 메서드입니다");  
        System.out.println(5 + 6);  
    }  
}
```

(a) main() 메서드에 작성된 코드

[그림 5-5] 메서드 선언 및 호출1

메서드로  
작성

```
public class Example001 {  
    public static void Method() {  
        System.out.println("static 메서드입니다");  
        System.out.println(5 + 6);  
    }  
    public static void main(String[] args) {  
        method( );  
    }  
}
```

메서드  
호출

(b) method() 메서드 작성과 호출

# 1. 메서드

## 예제 5-1 메서드 선언하고 호출하기

```
01 public class Method01 {  
02     public static void method() {  
03         System.out.println("static 메서드입니다.");  
04         System.out.println(5 + 6);  
05     }  
06  
07     public static void main(String[] args) {  
08         System.out.println("첫 번째 호출 메서드입니다.");  
09  
10         method();  
11  
12         System.out.println("두 번째 호출 메서드입니다.");  
13  
14         method();  
15     }  
16 }
```

### 실행 결과

```
첫 번째 호출 메서드입니다.  
static 메서드입니다.  
11  
두 번째 호출 메서드입니다.  
static 메서드입니다.  
11
```

# Section 02

## 사용자 정의 메서드 생성

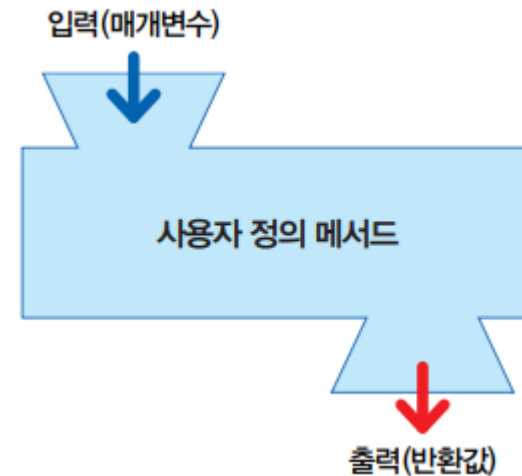


## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 있는 메서드

- 메서드명 앞에 String, int, boolean과 같은 자료형을 정의함
- 메서드 내부의 마지막 행에 return 키워드를 사용하여 메서드명 앞의 자료형과 동일한 값을 반환함

```
[접근제한자] 자료형 메서드명([매개변수목록]) {  
    // 메서드 본문  
    return 반환값;  
}
```



[그림 5-7] 반환 유형이 있는 메서드의 구조

## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 있는 메서드

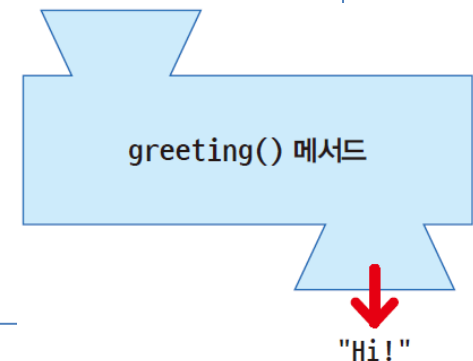
- 입력이 없고 출력이 있는 메서드
  - 입력(매개변수)이 없고 출력(반환 유형)만 있는 메서드

#### 입력이 없고 출력이 있는 메서드 예시

```
public class Example02 {  
    public static String greeting() {  
        return "Hi";  
    }  
    public static void main(String[] args) {  
        String str = greeting();  
        System.out.println(str + "JAVA");  
    }  
}
```

#### 실행 결과

Hi! Java

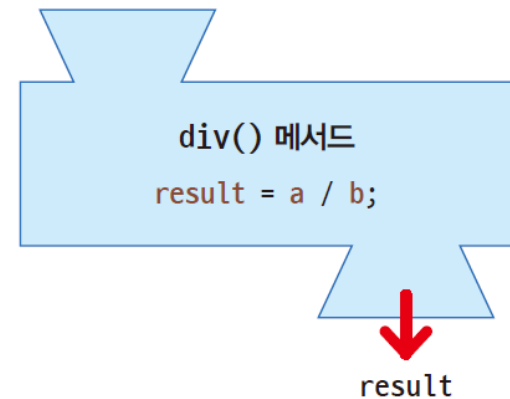


[그림 5-8] greeting() 메서드의 구조

## 2. 사용자 정의 메서드 생성

### 예제 5-2 매개변수가 없고 반환값이 있는 메서드 선언하고 호출하기

```
01 public class Method02 {  
02     public static int div( ) {  
03         int a = 10, b = 5;  
04         int result = a / b;  
05  
06         return result;  
07     }  
08     public static void main(String[] args) {  
09         int num = div ();  
10         System.out .println(num);  
11     }  
12 }
```



[그림 5-9] div() 메서드의 구조

실행 결과

2

## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 있는 메서드

- 입력과 출력이 있는 메서드

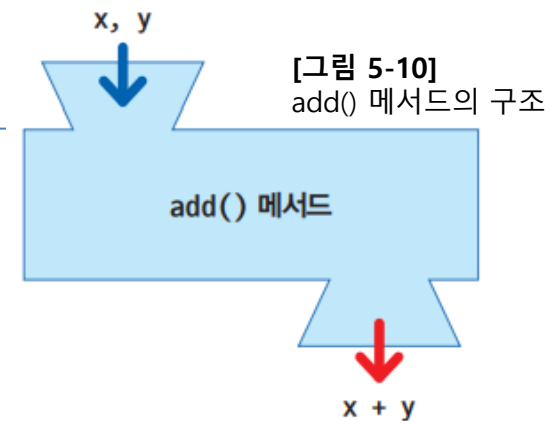
→ 입력(매개변수)과 출력(반환 유형)이 둘 다 있는 메서드

#### 입력과 출력이 있는 메서드 예시

```
public class Example03 {  
    public static int add(int x, int y) {  
        return x + y;  
    }  
    public static void main(String[] args) {  
        int a = 5, b = 6;  
        int sum = add(a,b) ;  
        System.out.println(a +"(와)과 "+ b +"의 합은 "+ sum +"입니다.");  
    }  
}
```

#### 실행 결과

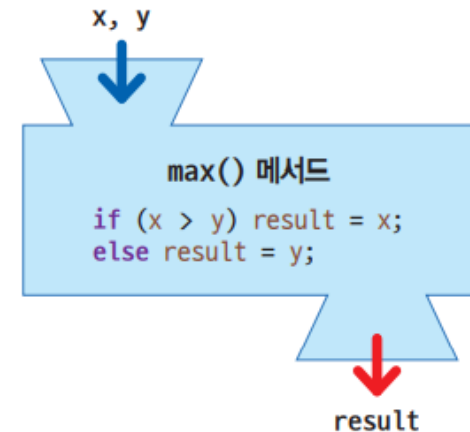
5(와)과 6의 합은 11입니다.



## 2. 사용자 정의 메서드 생성

### 예제 5-3 매개변수와 반환값이 있는 메서드 선언하고 호출하기

```
01 public class Method03 {  
02     public static int max(int x, int y) {  
03         int result;  
04         if (x > y) result = x;  
05         else result = y;  
06         return result;  
07     }  
08  
09     public static void main(String[] args) {  
10         int a = 5, b = 6;  
11         int num = max(a,b);  
12         System.out.println(a + "(와)과 " + b + "의 수 중 " + num + "이 큼니다.");  
13     }  
14 }
```



[그림 5-11] max() 메서드의 구조

#### 실행 결과

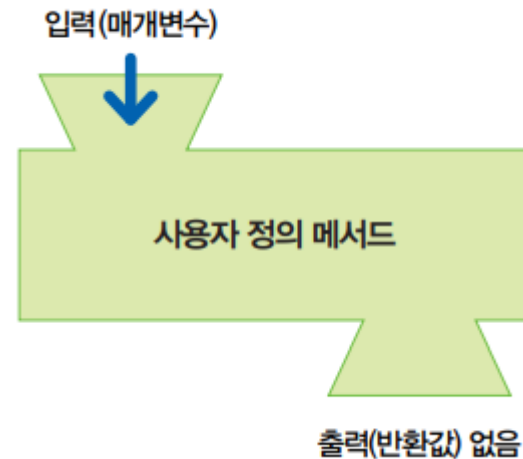
5(와)과 6의 수 중 6이 큼니다.

## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 없는 메서드

- 반환 유형이 없는 메서드는 메서드명 앞에 void 키워드를 사용함
- 메서드 안에 return이 없음

```
[접근제한자] void 메서드명([매개변수목록]) {  
    // 메서드 본문  
}
```



[그림 5-12] 반환 유형이 없는 메서드의 구조

## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 없는 메서드

- 입력과 출력이 없는 메서드

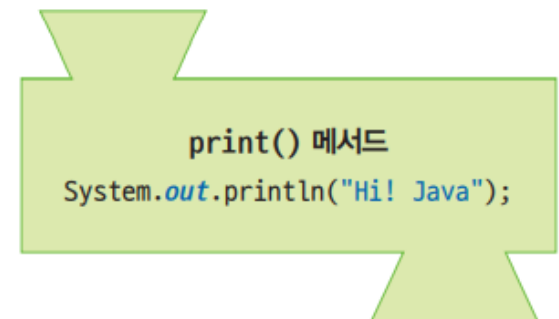
→ 입력(매개변수)도 출력(반환 유형)도 없는 메서드

#### 입력과 출력이 없는 메서드 예시

```
public class Example04 {  
    public static void print( ) {  
        System.out.println("Hi! Java");  
    }  
    public static void main(String[] args) {  
        print();  
    }  
}
```

실행 결과

Hi! Java

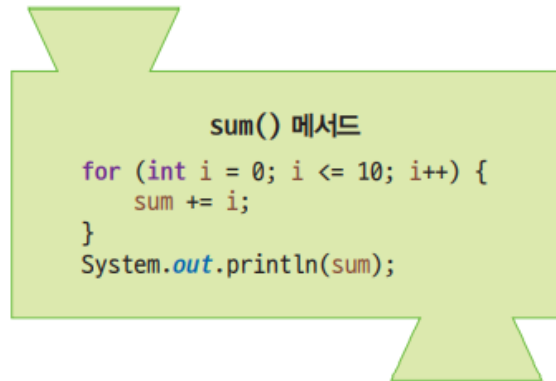


[그림 5-13] print() 메서드의 구조

## 2. 사용자 정의 메서드 생성

### 예제 5-4 매개변수와 반환값이 없는 메서드 선언하고 호출하기

```
01 public class Method04 {  
02     public static void sum() {  
03         int sum = 0;  
04         for (int i = 0; i <= 10; i++) {  
05             sum += i;  
06         }  
07         System.out.println(sum);  
08     }  
09  
10     public static void main(String[] args) {  
11         System.out.print("1부터 10의 합계 : ");  
12         sum();  
13     }  
14 }
```



[그림 5-14] sum() 메서드의 구조

#### 실행 결과

1부터 10의 합계 : 55



## 2. 사용자 정의 메서드 생성

### ■ 반환 유형이 없는 메서드

- 입력이 있고 출력이 없는 메서드

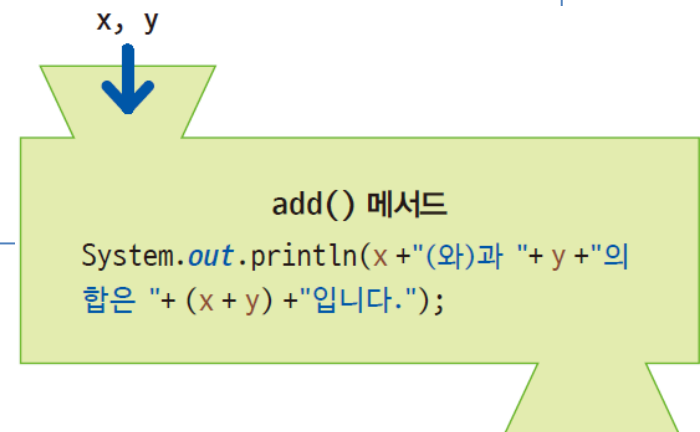
→ 입력(매개변수)이 있고 출력(반환 유형)이 없는 메서드

#### 입력이 있고 출력이 없는 메서드 예시

```
public class Example05 {  
    public static void add(int x, int y) {  
        System.out.println(x + "(와)과 " + y + "의 합은 " + (x + y) + "입니다.");  
    }  
    public static void main(String[] args) {  
        int a = 5, b = 6;  
        add(a,b);  
    }  
}
```

#### 실행 결과

5(와)과 6의 합은 11입니다.

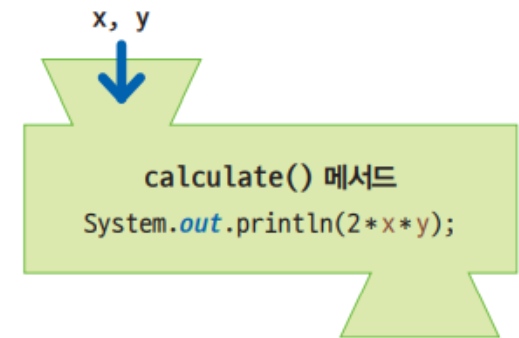


[그림 5-15] `add()` 메서드의 구조

## 2. 사용자 정의 메서드 생성

### 예제 5-5 매개변수가 있고 반환값이 없는 메서드 선언하고 호출하기

```
01 public class Method05 {  
02     public static void calculate(int x, double y) {  
03         System.out.println(2 * x * y);  
04     }  
05  
06     public static void main(String[] args) {  
07         int a = 4; // 반지름  
08         double pi = 3.14;  
09         System.out.println("원의 둘레 구하는 공식 : 2 x 반지름 x 원주율 ");  
10  
11         System.out.print("2 x " + a + " x " + pi + " = ");  
12         calculate(a, pi);  
13     }  
14 }
```



[그림 5-16] calculate() 메서드의 구조

#### 실행 결과

원의 둘레 구하는 공식 : 2 x 반지름 x 원주율  
2 x 4 x 3.14 = 25.12

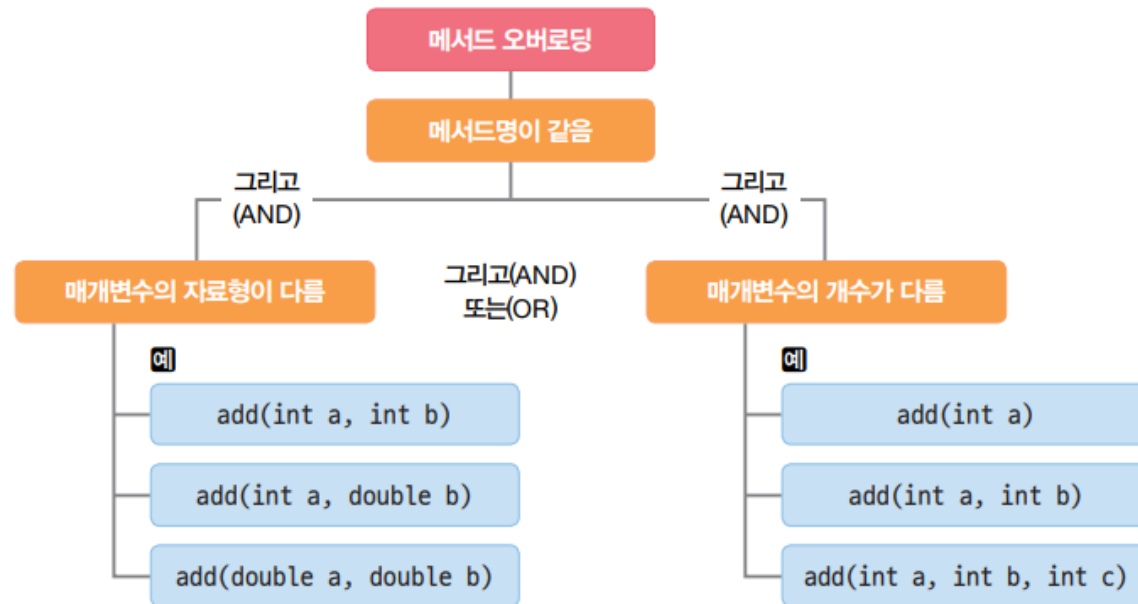
# Section 03

## 메서드 오버로딩

### 3. 메서드 오버로딩

#### ■ 메서드 오버로딩(method overloading)

- 메서드명이 같지만 매개변수가 다른 메서드를 하나의 메서드명으로 정의하는 것
- 메서드 오버로딩을 위한 조건
  - 메서드명이 같음
  - 매개변수의 자료형이나 개수가 다름



[그림 5-17] add() 메서드의 오버로딩 조건

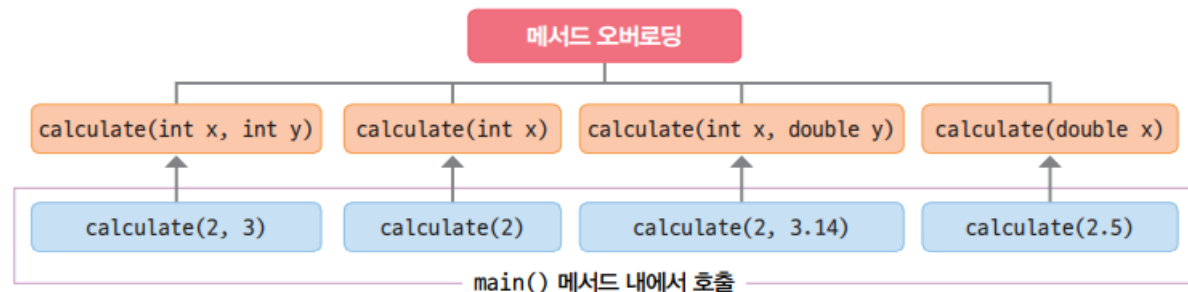
### 3. 메서드 오버로딩

#### 메서드 오버로딩 예시

```
public class Example06 {  
    public static void calculate(int x, int y) {  
        System.out.println(x * y);  
    }  
    public static void calculate(int x) {  
        System.out.println(x * x);  
    }  
    public static void calculate(int x, double y) {  
        System.out.println(x * y);  
    }  
    public static void calculate(double x) {  
        System.out.println(x * x);  
    }  
    public static void main(String[] args) {  
        calculate(2, 3);  
        calculate(2, 3.14);  
        calculate(2);  
        calculate(2.5);  
    }  
}
```

#### 실행 결과

6  
6.28  
4  
6.25

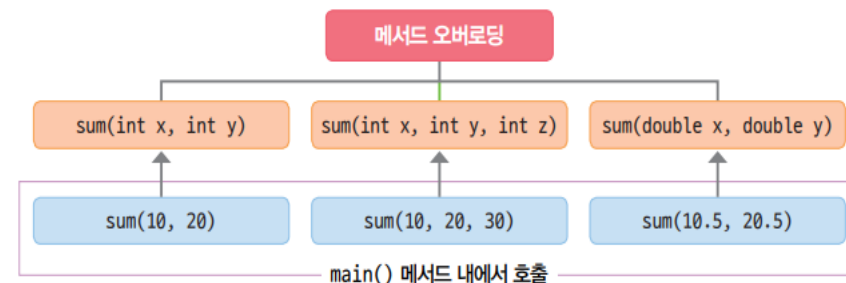


[그림 5-18] 오버로딩된 calculate() 메서드의 호출

### 3. 메서드 오버로딩

#### 예제 5-6 메서드 오버로딩하기

```
01 public class Method06 {  
02     public static int sum(int x, int y) {  
03         return (x + y);  
04     }  
05  
06     public static int sum(int x, int y, int z) {  
07         return (x + y + z);  
08     }  
09  
10     public static double sum(double x, double y) {  
11         return (x + y);  
12     }  
13  
14     public static void main(String[] args) {  
15         System.out.println("sum(10, 20)의 값 : "+ sum(10, 20));  
16         System.out.println("sum(10, 20, 30)의 값 : "+ sum(10, 20, 30));  
17         System.out.println("sum(10.5, 20.5)의 값 : "+ sum(10.5, 20.5));  
18     }  
19 }
```



[그림 5-19] 오버로딩된 sum() 메서드의 호출

#### 실행 결과

sum(10, 20)의 값 : 30  
sum(10, 20, 30)의 값 : 60  
sum(10.5, 20.5)의 값 : 31.0

**[프로젝트]**

**메뉴별 메서드 만들기**

# 메뉴별 메서드 만들기

- 메서드를 적용하여 온라인 서점의 메뉴별 메서드를 만들어봅시다.



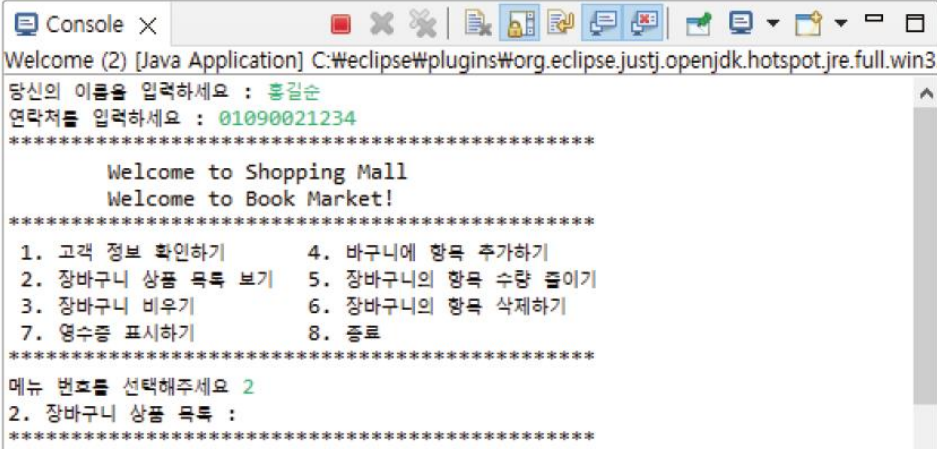
[그림 5-20] 메뉴별 메서드 만들기



# 메뉴별 메서드 만들기

- 만들게 될 메뉴별 메서드

- 1. 고객 정보 확인하기: menuGuestInfo()
- 2. 장바구니 상품 목록 보기: menuCartItemList()
- 3. 장바구니 비우기: menuCartClear()
- 4. 바구니에 항목 추가하기: menuCartAddItem()
- 5. 장바구니의 항목 수량 줄이기: menuCartRemoveItemCount()
- 6. 장바구니의 항목 삭제하기: menuCartRemoveItem()
- 7. 영수증 표시하기: menuCartBill()
- 8. 종료: menuExit()



```
Console X
Welcome (2) [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3
당신의 이름을 입력하세요 : 홍길순
연락처를 입력하세요 : 01090021234
*****
Welcome to Shopping Mall
Welcome to Book Market!
*****
1. 고객 정보 확인하기      4. 바구니에 항목 추가하기
2. 장바구니 상품 목록 보기  5. 장바구니의 항목 수량 줄이기
3. 장바구니 비우기        6. 장바구니의 항목 삭제하기
7. 영수증 표시하기        8. 종료
*****
메뉴 번호를 선택해주세요 2
2. 장바구니 상품 목록 :
*****
```

[그림 5-20] 메뉴별 메서드 만들기 실행 결과

# 메뉴별 메서드 만들기

## ■ 메뉴 선택 시 메뉴 정보 출력하기

- 01 선택 가능한 메뉴 목록 출력 메서드

- Welcome.java 파일을 열고, main() 메서드가 끝난 다음에 menuIntroduction() 작성하기
- main() 메서드 내부의 메뉴 목록을 출력하는 부분을 주석 처리하고 menuIntroduction() 호출하기

- 02 입력받은 고객 정보 출력 메서드

- Welcome.java 파일에 '메뉴 번호 1'을 선택한 고객의 정보를 메서드 내에 전달하여 출력하는 메서드 menuGuestInfo()를 작성하기
- main() 메서드의 case 1:에서 메시지 출력 부분을 주석 처리한 뒤 menuGuestInfo() 호출하기

- 03 메뉴 선택별 메서드

- Welcome.java 파일에 '메뉴 번호 2~8'의 목록별 메서드를 작성하고 호출하기