# 프로그래밍기초

Chapter 05. 메서드

# Section 01

# 메서드

# 메서드

■ 메서드(method)

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

```java
import java.util.Scanner;
public class Example00 {
    public static void main(String[] args) {
        int num;
        Scanner s = new Scanner(System.in);

        num = s.nextInt();
        if (num%2==0) System.out.println("짝수입니다");
        else System.out.println("홀수입니다");

        num = s.nextInt();
        if (num%2==0) System.out.println("짝수입니다");
        else System.out.println("홀수입니다");

        num = s.nextInt();
        if (num%2==0) System.out.println("짝수입니다");
        else System.out.println("홀수입니다");
    }
}
```

(a) 같은 코드를 반복하는 경우

동일 코드를 메서드로 작성

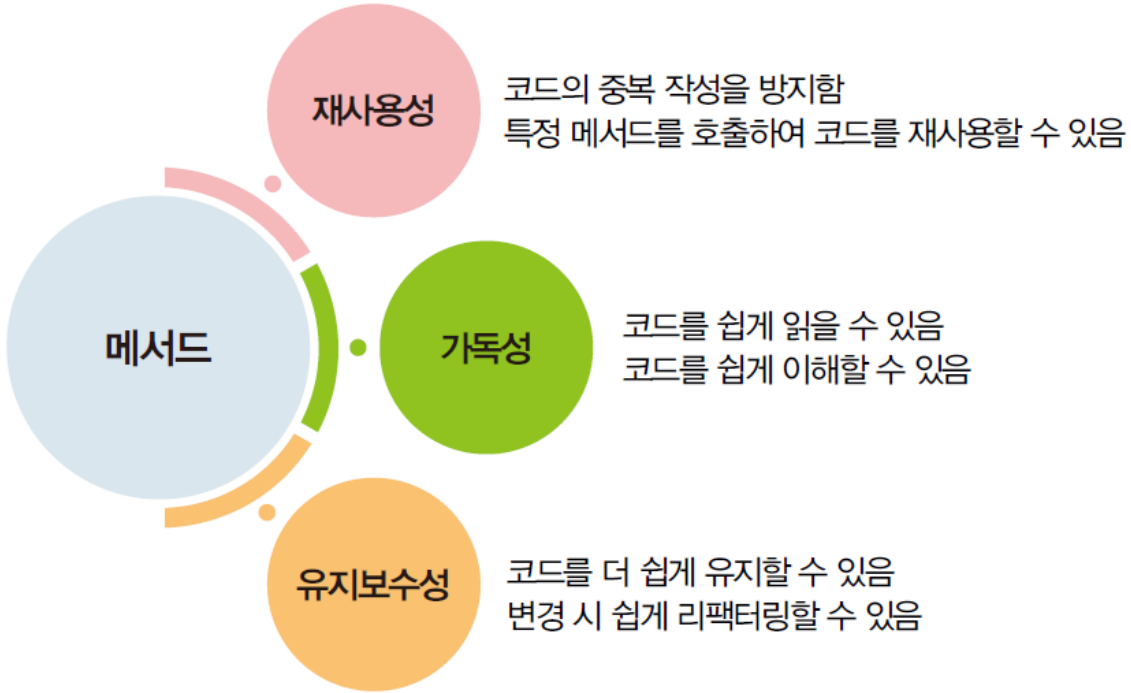
메서드 호출

```java
import java.util.Scanner;
public class Example001 {
    public static void Method(int num) {
        if (num%2==0) System.out.println("짝수입니다");
        else System.out.println("홀수입니다");
    }

    public static void main(String[] args) {
        int num;
        Scanner s = new Scanner(System.in);

        num = s.nextInt();
        Method(num);

        num = s.nextInt();
        Method(num);

        num = s.nextInt();
        Method(num);
    }
}
```

(b) 메서드를 만드는 경우

# 메서드

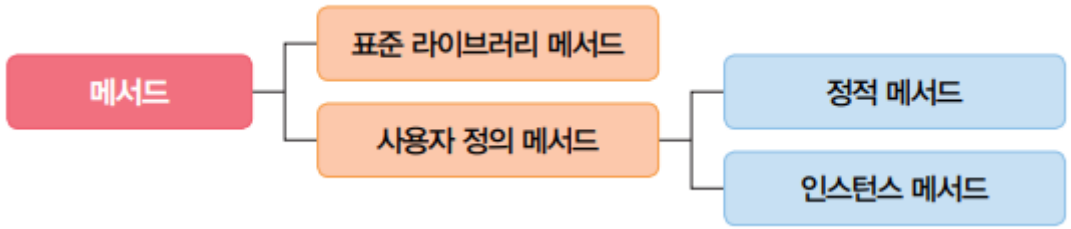
■ 메서드의 필요성


메서드
재사용성
코드의 중복 작성을 방지함
특정 메서드를 호출하여 코드를 재사용할 수 있음
가독성
코드를 쉽게 읽을 수 있음
코드를 쉽게 이해할 수 있음
유지보수성
코드를 더 쉽게 유지할 수 있음
변경 시 쉽게 리팩터링할 수 있음

# 메서드

■ 메서드의 유형

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



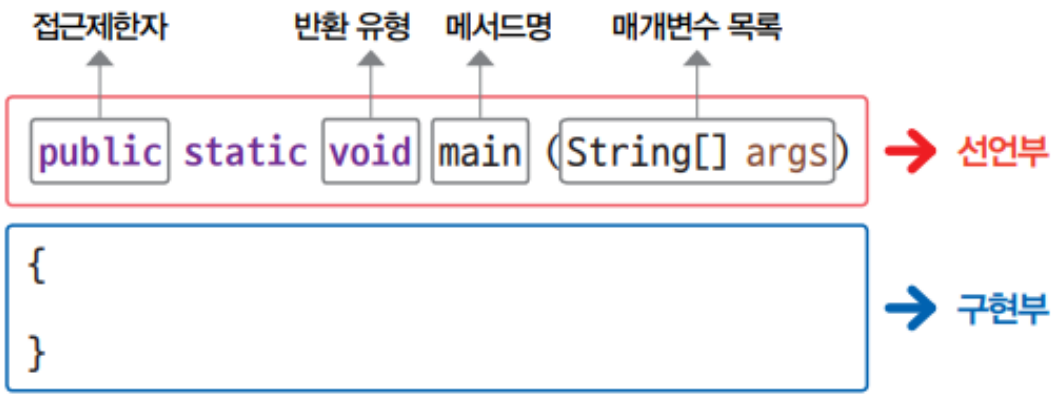
# 메서드

■ 메서드의 기본 구조

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

선언부

구현부



# 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
```

# 메서드

■ 메서드 호출

- 메서드 호출 예제

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

(a) main() 메서드에 작성된 코드

메서드로 작성

```java
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

# 메서드

■ 예제 5-1. 메서드 선언하고 호출하기

```
01 public class Mehod01 {
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

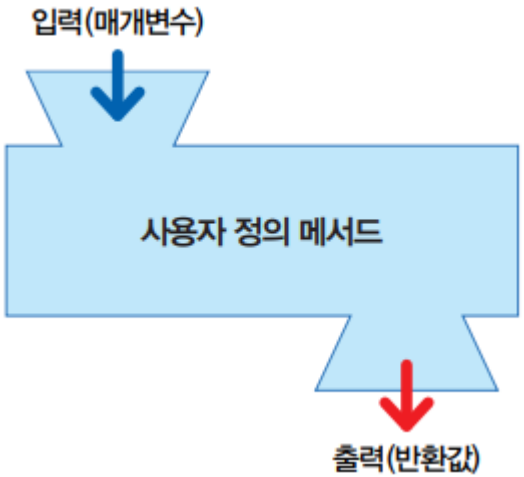
# 사용자 정의 메서드 생성

# 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;
}
```



# 사용자 정의 메서드 생성

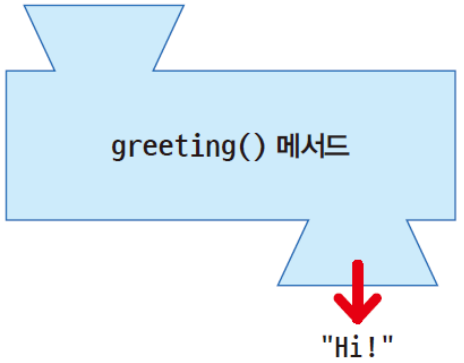
■ 반환 유형이 있는 메서드
- 입력이 없고 출력이 있는 메서드
  → 입력(매개변수)이 없고 출력(반환 유형)만 있는 메서드

■ 입력이 없고 출력이 있는 메서드 예시

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

실행 결과

```
Hi! Java
```



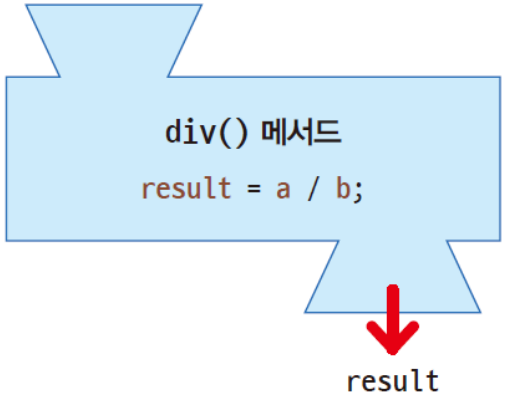
# 사용자 정의 메서드 생성

■ 예제 5-2 매개변수가 없고 반환 값이 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 }
```

실행 결과

```
2
```



# 사용자 정의 메서드 생성

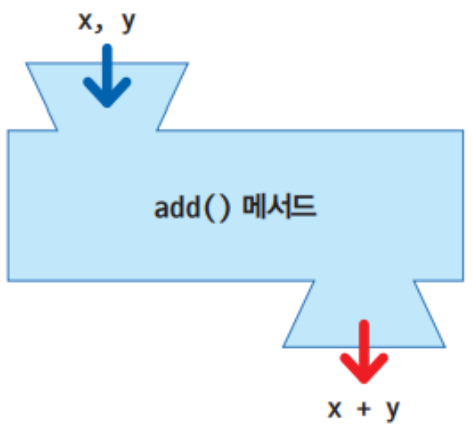
■ 반환 유형이 있는 메서드
- 입력과 출력이 있는 메서드
  → 입력(매개변수)과 출력(반환 유형)이 둘 다 있는 메서드

■ 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.
```



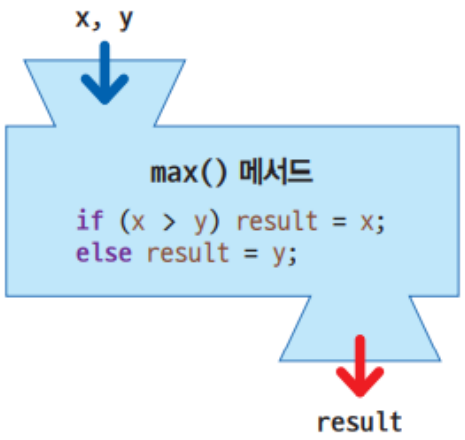
# 사용자 정의 메서드 생성

■ 예제 5-3. 매개변수와 반환 값이 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 큽니다.");
13     }
14 }
```

실행 결과

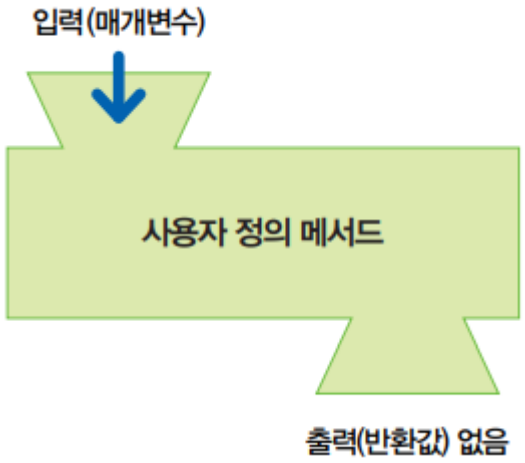
```
5(와)과 6의 수 중 6이 큽니다.
```



# 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
}
```



# 사용자 정의 메서드 생성

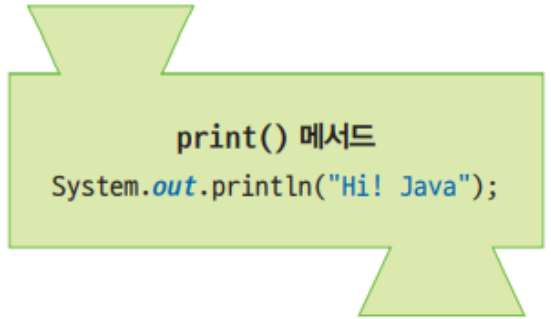
■ 반환 유형이 없는 메서드
- 입력과 출력이 없는 메서드
  → 입력(매개변수)도 출력(반환 유형)도 없는 메서드

■ 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
```



# 사용자 정의 메서드 생성

■ 예제 5-4. 매개변수와 반환 값이 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 }
```

실행 결과

```
1부터 10의 합계 : 55
```

sum() 메서드

```
for (int i = 0; i <= 10; i++) {
    sum += i;
}
System.out.println(sum);
```

# 사용자 정의 메서드 생성

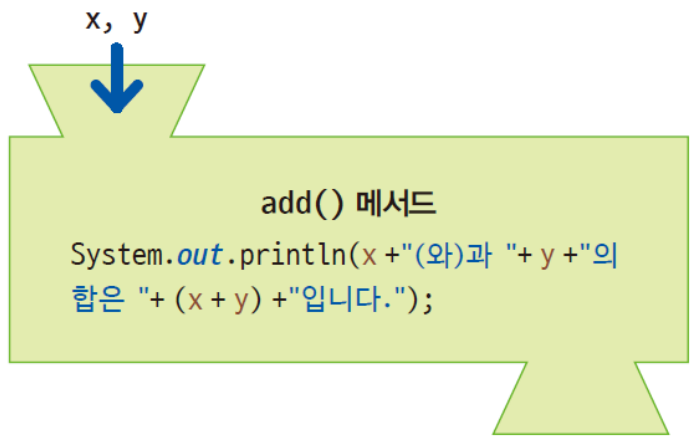
■ 반환 유형이 없는 메서드
- 입력이 있고 출력이 없는 메서드
  → 입력(매개변수)이 있고 출력(반환 유형)이 없는 메서드

■ 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.
```



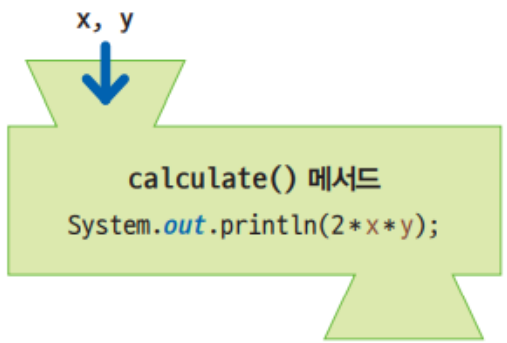
# 사용자 정의 메서드 생성

■ 예제 5-5. 매개변수가 있고 반환 값이 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 }
```

실행 결과

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



# Section 03

# 메서드 오버로딩

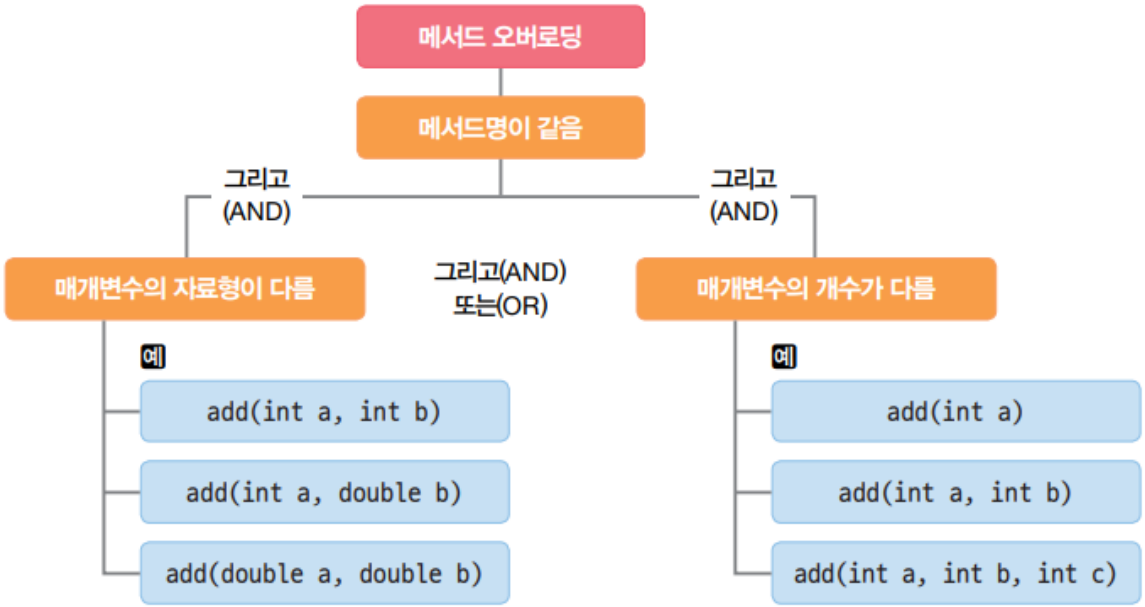
# 메서드 오버로딩

■ 메서드 오버로딩(method overloading)

- 메서드명이 같지만 매개변수가 다른 메서드를 하나의 메서드명으로 정의하는 것
- 메서드 오버로딩을 위한 조건

  → 메서드명이 같음

  → 매개변수의 자료형이나 개수가 다름



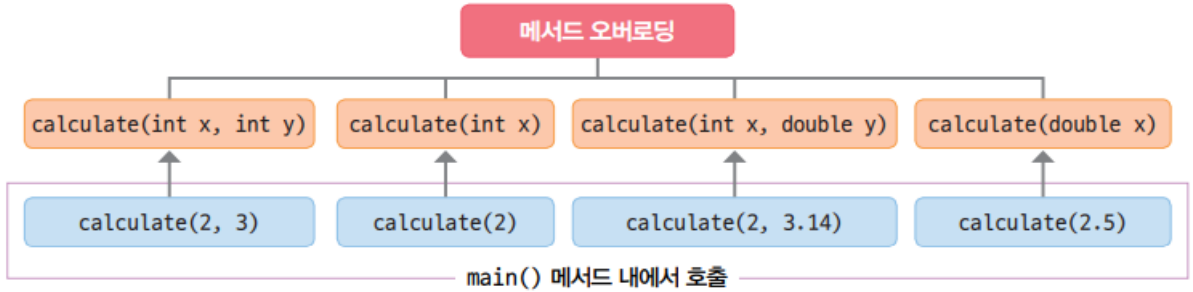
# 메서드 오버로딩

■ 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
```



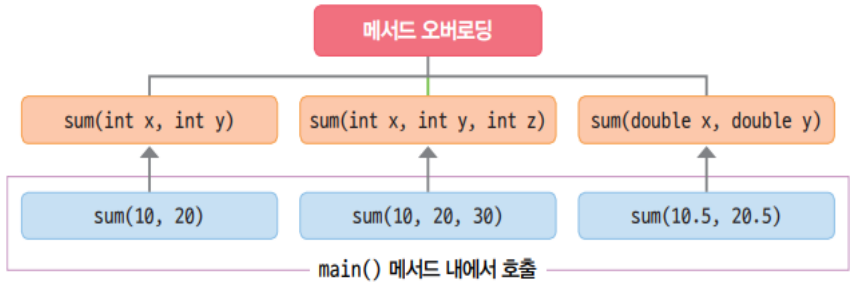
# 메서드 오버로딩

■ 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 }
```

실행 결과

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



프로그래밍기초



<Q&A : lsw@kopo.ac.kr>