Method

Java Programming

Byeongjoon Noh

powernoh@sch.ac.kr

SOON CHUN HYANG UNIVERSITY

Most of the slides are available on Senseable AI Lab homepage: https://sailab.space/courses/

Contents

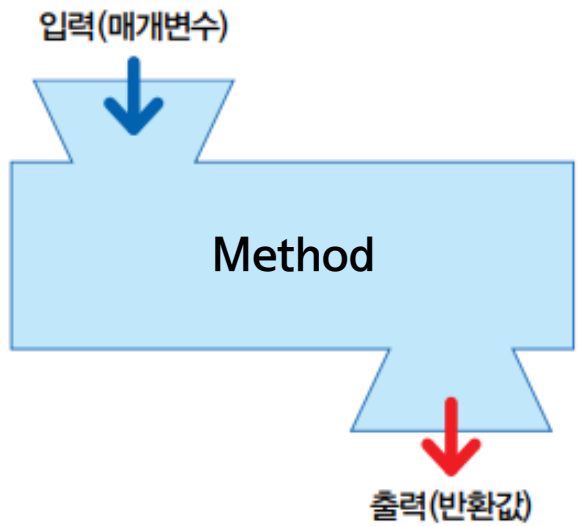
1. Introduction on method

2. Method overloading

1. Introduction on method

What is method?

- Method
  - a block of code that performs a specific task
  - a way to encapsulate a sequence of statements into a unit
    - can be executed whenever you need to perform the task, often with different inputs
    - ex) System.out.println() ← a method to display your input messages on the console



What is method?

- Field and method

```
public class _01_Method {

    private int year;
    private String month;
    private int day;

    public String name;

    // 메소드 정의
    public static void sayHello() {
        System.out.println("This is sayHello() method");
    }

    public static void main(String[] args) {
        // 메소드 호출
        System.out.println("Before method call");
        sayHello();
        sayHello();
        sayHello();
        System.out.println("After method call");
    }
}
```

필드 (Field)
멤버 변수 (Member variable)
전역 변수 (Global variable)

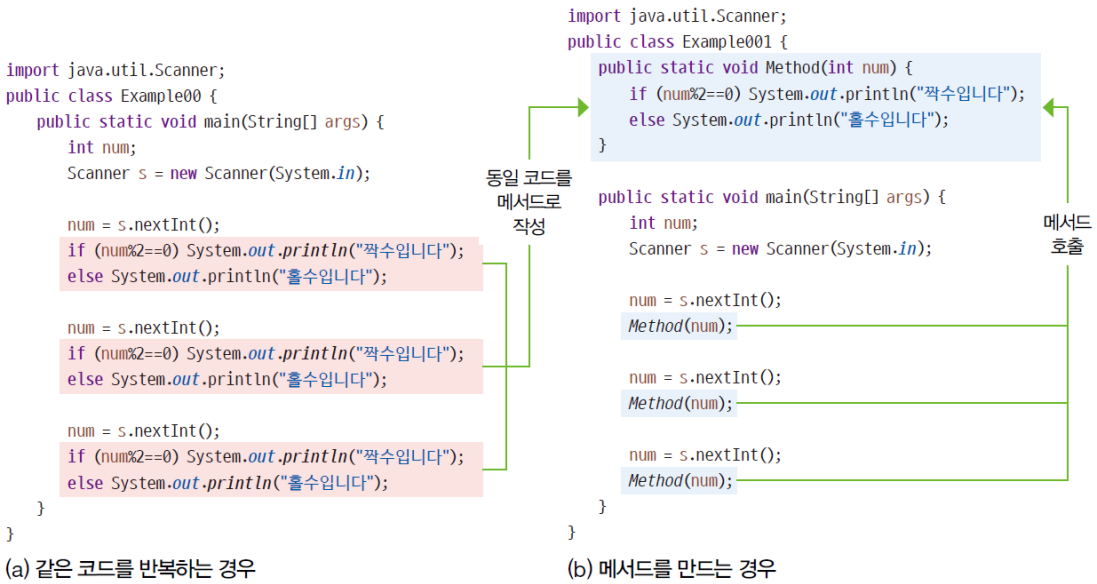
Method (definition)
멤버 함수 (Member function)

Method call

main() function

Why methods matter

- Reusability
  - enhance code reusability across different parts of a program
  - avoid repetition (write once, use many times)
  - customizing methods for varied inputs (parameterization)
- Modularity (encapsulation)
  - breaking down a complex problem into manageable units
  - hiding the complexity of operations
- Readability
  - improve code readability and maintenance
  - facilitate debugging by isolating code segments

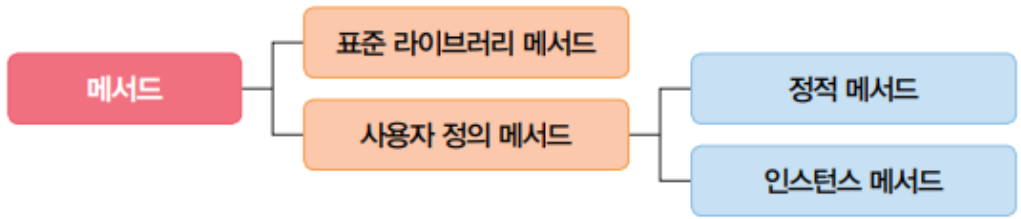


(a) 같은 코드를 반복하는 경우

(b) 메서드를 만드는 경우

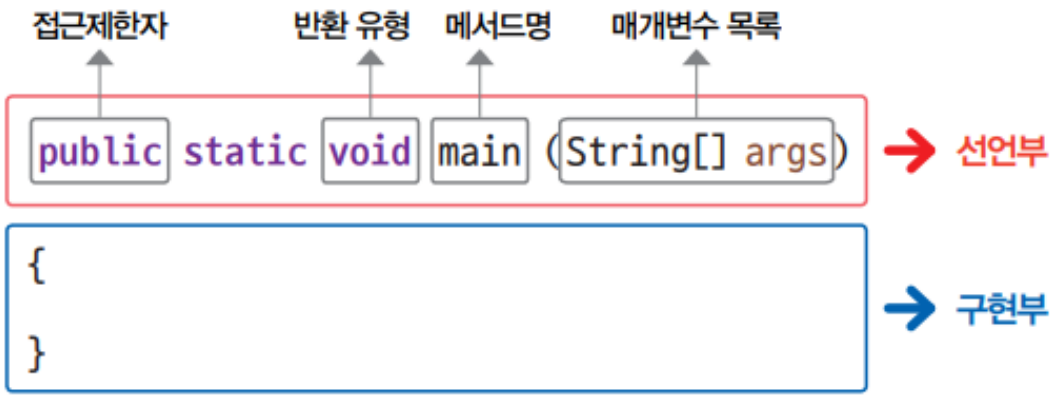
Types of method

- Two types of method in Java
  - Standard library methods: predefined in Java's API
    - offer ready-made functionality to perform common tasks
    - efficiency and reliability; extensively tested and coumented
  - examples
    - Math.sqrt(double a): calculates the square root
    - System.out.println(String x): prints text to the console
  - User-defined methods
    - created by programmers to perform specific tasks
    - flexibility to address specific problems



Basic structure

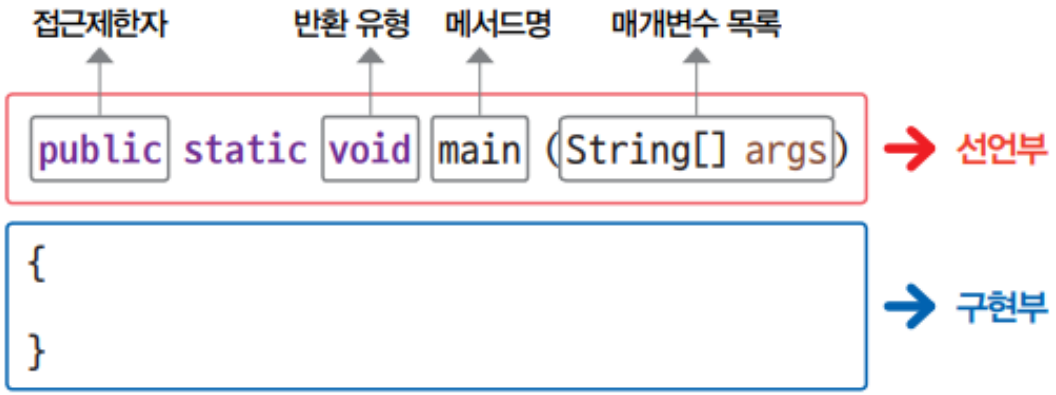
- Basic syntax of Java methods



  - method name
    - identifies the method
    - follows Java naming conventions
    - should be descriptive of its purpose

Basic structure

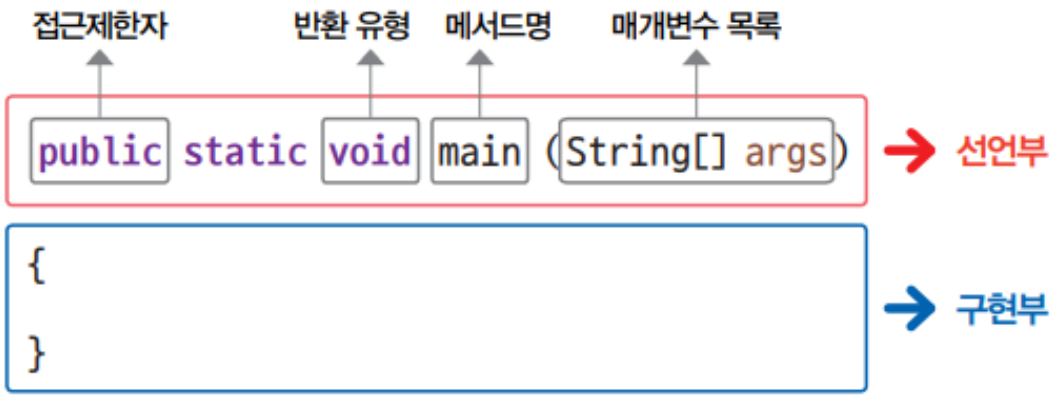
- Basic syntax of Java methods



  - return type
    - specifies the types of value the method returns
      - int, double, String, etc.
    - use ‘void’ if no value is returned

Basic structure

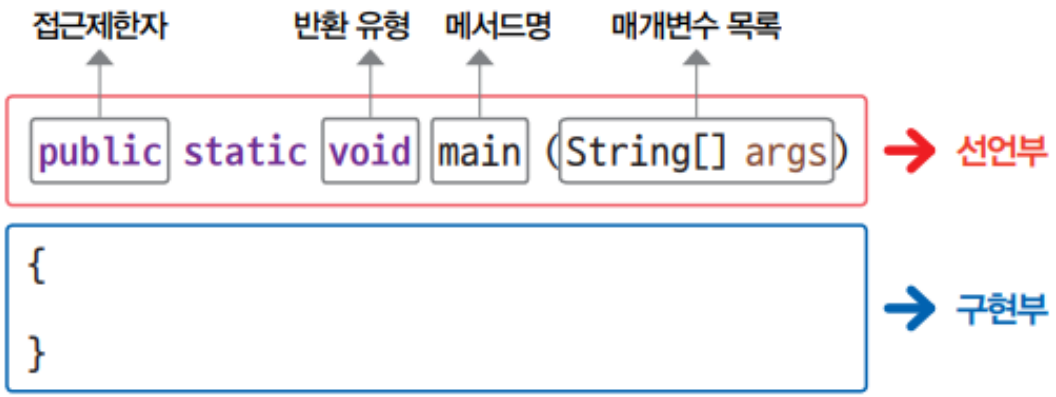
- Basic syntax of Java methods



  - parameters (매개변수)
    - optional: methods may accept none, one, or multiple parameters
    - provide input to a method
    - each parameter has a type and names

Basic structure

- Basic syntax of Java methods



  - method body
    - contains code that defines the method's operations
    - enclosed in curly braces { }
    - can contain any number of statements

Basic structure

- Example: adding two numbers

```
public int addNumbers(int a, int b) {
  int result = a + b;
  return result
}
```

  - method name: `addNumbers`
  - return type: `int`
    - actually return integer type (result)
  - parameters: `int a, int b`
  - method body:
    - `int result = a + b;`
    - `return result`

Note: 함수는 반환값이 없거나 오직 1개의 값만 반환할 수 있음

Note: 접근 지정자

- public: 모든 class로부터 접근 가능한 메소드

```
public static void sayHello() {
  System.out.println("This is sayHello() mehtod");
}
```

- private: class 내부에서만 접근 가능

```
private static void sayHello() {
  System.out.println("This is sayHello() mehtod");
}
```

Basic structure - examples

```
public static double calculateAreaOfCircle(double radius) {
  return Math.PI * radius * radius;
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static double fahrenheitToCelsius(double fahrenheit) {
  return (fahrenheit - 32) * 5 / 9;
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static double calculatePrismVolume(double length, double width, double height) {
  return length * width * height;
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static String createFullName(String firstName, String middleName, String lastName) {
  return firstName + " " + middleName + " " + lastName;
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static double calculateFinalPrice(double price, double taxRate, double discount) {
  double taxAmount = price * taxRate / 100;
  double discountAmount = price * discount / 100;
  return price + taxAmount - discountAmount;
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static void printGreeting(String name, int hourOfDay) {
  String greeting;
  if (hourOfDay < 12) {
    greeting = "Good morning";
  } else if (hourOfDay < 18) {
    greeting = "Good afternoon";
  } else {
    greeting = "Good evening";
  }
  System.out.println(greeting + ", " + name + "!");
}
```

- method name:
- return type:
- parameters:
- method body:

Basic structure - examples

```
public static boolean isPrime(int number) {
  if (number <= 1) {
    return false;
  }
  for (int i = 2; i <= Math.sqrt(number); i++) {
    if (number % i == 0) {
      return false;
    }
  }
  return true;
}
```

- method name:
- return type:
- parameters:
- method body:

Method call

- Method call
  - a process of invoking a method written in a program so that the instructions contained within that method are executed

- Structure of a method call
  - consisting of the method name followed by parentheses
    - WITHOUT data type

```
methodName(argument1, argument2, ..., argumentN);
```

    - argument: 인수

Method call

- Example of method call

```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

```java
int square(int num) {
  return num * num;
}
...
...
result = square(10);
// code
```

return value

method call

Usage of method

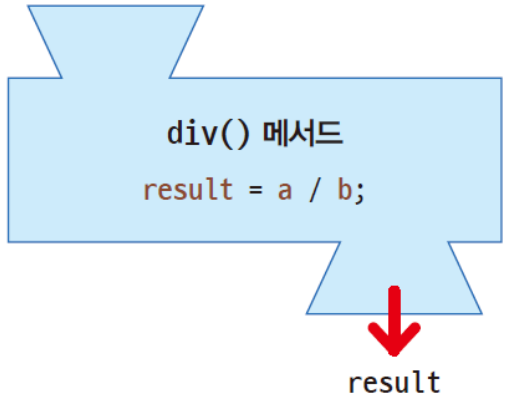
- Example for method definition and call
  - no parameters, no return type

```
public static void method() {
  System.out.println("static 메서드입니다.");
  System.out.println(5 + 6);
}

public static void main(String[] args) {
  System.out.println("첫 번째 호출 메서드입니다.");
  method();
  System.out.println("두 번째 호출 메서드입니다.");
  method();
}
```

Usage of method

- Example for method definition and call
  - no parameters, but return value exists

```
public static int div( ) {
  int a = 10, b = 5;
  int result = a / b;
  return result;
}
public static void main(String[] args) {
  int num = div();
  System.out.println(num);
 }
```



Note: Local variable and global variable

- Local variable
  - variables defined within a method and their scope is limited to the method in which they are declared

```
public static void printX(int x) {
  System.out.println("X in printX method = " + x);
  x++;
  System.out.println("X in printX method = " + x);
}

public static void main(String[] args) {
  int x = 10;
  System.out.println("X in main method = " + x);
  printX(x);
  x = 50;
  System.out.println("X in main method = " + x);
}
```

```
X in main method = 10
X in printX method = 10
X in printX method = 11
X in main method = 50
```

Note: Local variable and global variable

- Global variable
  - commonly used to refer to variables that have a global scope
  - declared with the 'static' keyword within a class but outside any method
  - default value is 0 (without initialization)

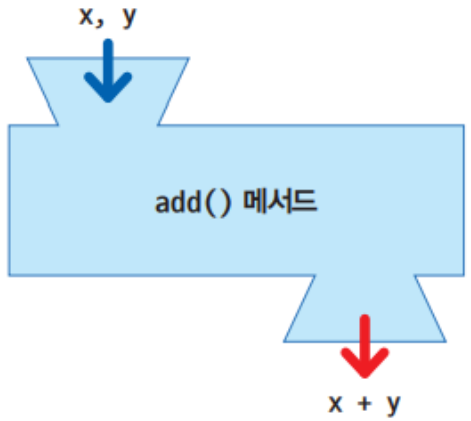
```
public static int count;
public static void counter1() { count++;}
public static void counter2() { count = count + 100;}
  System.out.println("Count = " + count); // Output: 0
  counter1();
  System.out.println("Count = " + count); // Output: 1
  counter2();
  System.out.println("Count = " + count); // Output: 101
  count = 10;
  System.out.println("Count = " + count); // Output: 10
}
```

Usage of method

- Example for method definition and call
  - return value and parameters exist

```
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.");
}
```

Pay attention to a format for method call

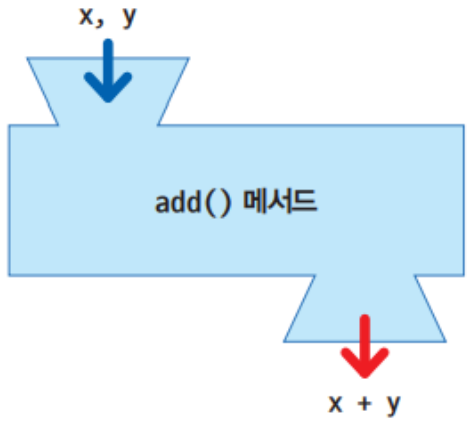


  - no data type designation in arguments when method calls

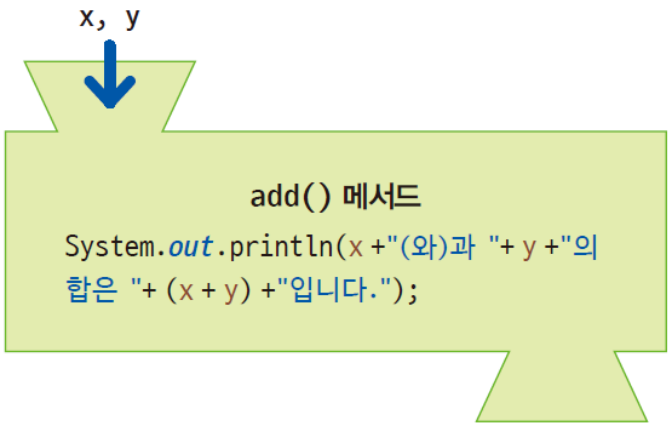
Usage of method

- Example for method definition and call – return or not?

```
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.");
}
```



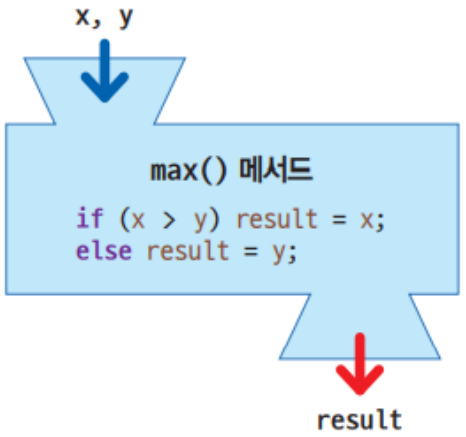

```
public static void add(int x, int y) {
  System.out.println(x +"(와)과 "+ y +"의 합은 "+ (x + y) +"입니다.");
}

public static void main(String[] args) {
  int a = 5, b = 6;
  add(a,b);
}
```



Usage of method

- Example for method definition and call
  - return value and parameters exist

```
public static int max(int x, int y) {
  int result;
  if (x > y) result = x;
  else result = y;
  return result;
}

public static void main(String[] args) {
  int a = 5, b = 6;
  int num = max(a,b);
  System.out.println(a + "(와)과 "+ b +"의 수 중 "+ num +"이 큽니다.");
}
```

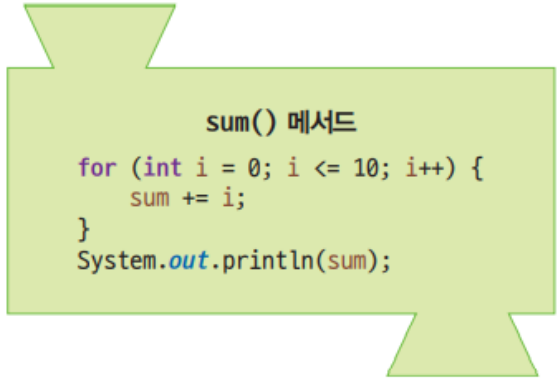


  - no data type designation in arguments when method calls

Usage of method

- Example for method definition and call

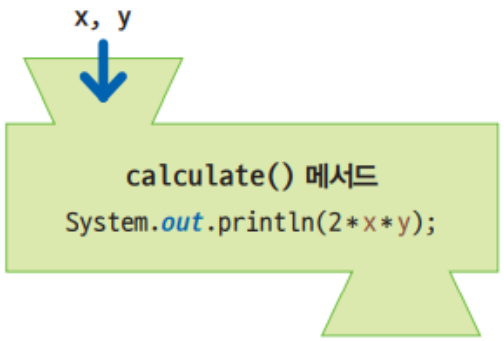
```
public static void sum() {
  int sum = 0;
  for (int i = 0; i <= 10; i++) {
    sum += i;
  }
  System.out.println(sum);
}

public static void main(String[] args) {
  System.out.print("1부터 10의 합계 : ");
  sum();
}
```



Usage of method

- Example for method definition and call

```
public static void calculate(int x, double y) {
  System.out.println(2 * x * y);
}

public static void main(String[] args) {
  int r = 4; //
  double pi = 3.14;
  System.out.println("원의 둘레 구하는 공식 : 2 * 반지름 * 원주율 ");
  System.out.print("2 * "+ r + " x " + pi +" = ");
  calculate(r, pi);
}
```



Examples and practices for method

- 사용자로부터 두 실수를 입력받고 두 수 중 큰 수를 반환하는 함수와 이를 출력하는 프로그램을 작성해보세요.
  - file path and name: Chap04Example/MethodPractice01.java
  - requirement
    - method name: getLarger
  - input and output examples

```
Enter the two real numbers: 5 -1
The larger number is 5.0
```

```
Enter the two real numbers: 17.4
17.33
The larger number is 17.4
```

Examples and practices for method

- 사용자로부터 실수 하나를 입력받고 양수, 음수 또는 0을 판별하여 출력하는 함수를 작성해보세요.
  - file path and name: Chap04Example/MethodPractice02.java
  - requirement
    - method name: checkNumber
  - input and output examples

```
Enter the one number: 5.71
5.71 is positive.
```

```
Enter the one number: 0.1
0.1 is positive.
```

```
Enter the one number: -1.8
-1.8 is negative.
```

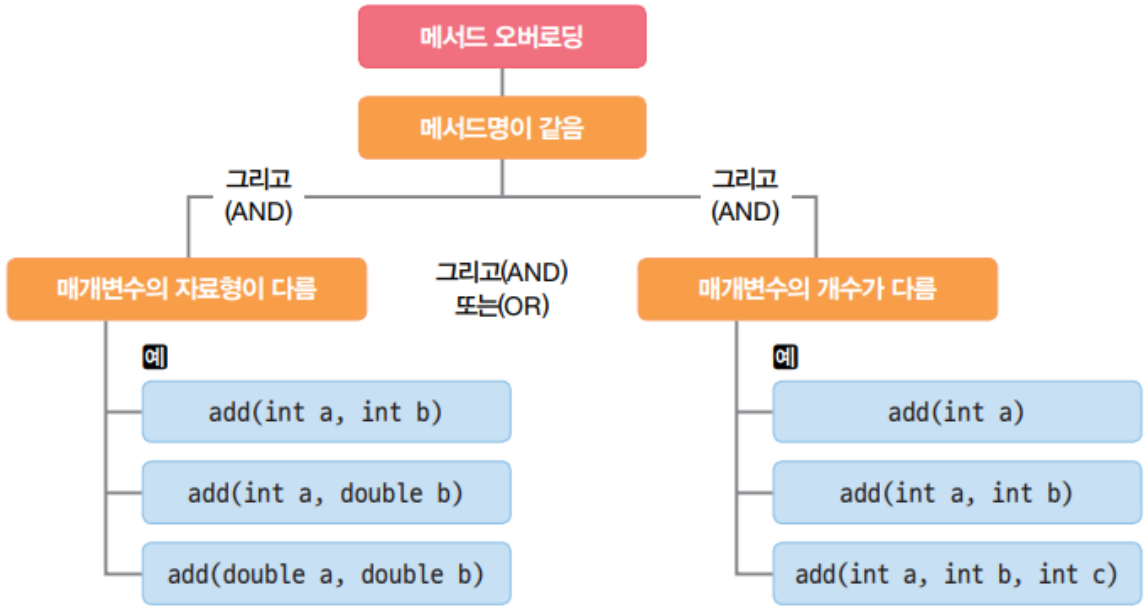
2. Method overloading

What is concept of “overloading”?

- Method overloading
  - allows to have more than one method with the same name
    - as long as their parameter lists are different
  - enables methods to perform similar functions but with different types or numbers of inputs
  - supports polymorphism, allowing developers to write more flexible and readable code

Key points of method overloading

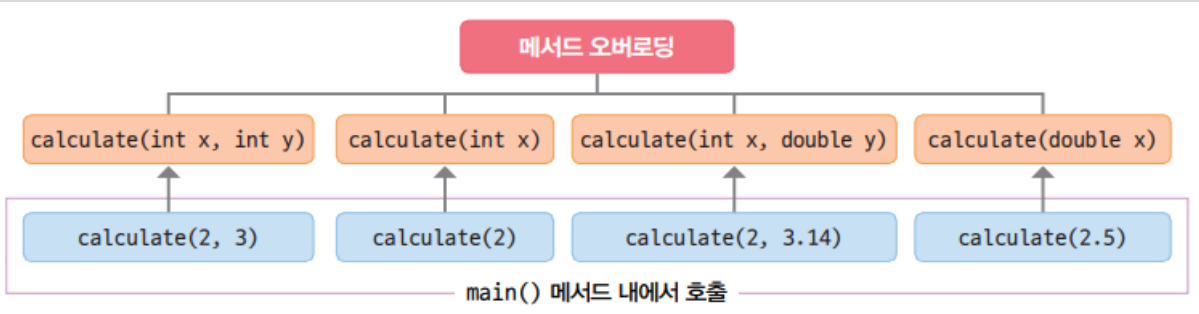
- Different parameter lists
  - the overloaded methods must differ in the number or type of parameters
- Return type irrelevance
  - return type of the methods can be the same or different
    - in fact, return type alone is not sufficient for method overloading
- Same method name
  - the methods must have the same name but must differ in their parameter lists



- parameter의 data type을 보고 컴파일러가 알아서 적절한 method call을 수행

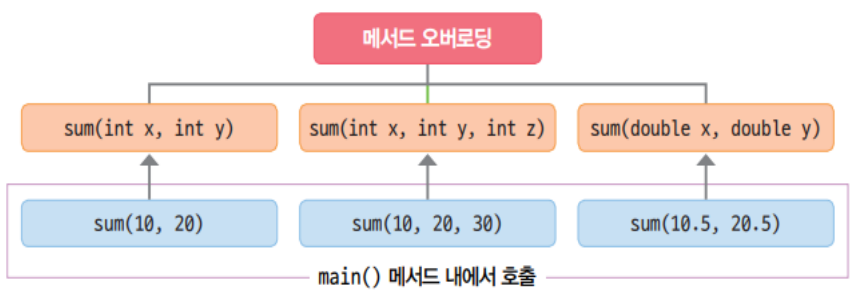
Usage of method overloading

```
public class MethodOverladingExample01 {
  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);
  }
}
```



Usage of method overloading

```
public class MethodOverladingExample02 {
  public static int sum(int x, int y) {
    return (x + y);
  }
  public static int sum(int x, int y, int z) {
    return (x + y + z);
  }
  public static double sum(double x, double y) {
    return (x + y);
  }

  public static void main(String[] args) {
    System.out.println("sum(10, 20)의 값 : "+ sum(10, 20));
    System.out.println("sum(10, 20, 30)의 값 : "+ sum(10, 20, 30));
    System.out.println("sum(10.5, 20.5)의 값 : "+ sum(10.5, 20.5));
  }
}
```



Examples and practices for method overloading

- 메소드 오버로딩을 활용하여 first name(이름)만 입력된 경우와 last name(성)이 함께 입력된 경우와 이름과 성이 함께 입력된 경우를 구분하도록 하는 메소드 2개를 작성해보세요.
  - file path and name: Chap04Example/MethodOverloadingPractice01.java
  - requirement
    - method name: printName()
  - output examples

```
Hello, Noh
Hello, Noh Byeongjoon
```

Examples and practices for method overloading

- 사용자로부터 삼각형의 밑변(base)과 높이(height)를 입력받고, 삼각형의 넓이를 계산하고 반환하는 메소드를 작성하세요.
  - file path and name: Chap04Example/MethodOverloadingPractice02.java
  - requirement
    - method name: getArea()
    - 입력받은 수의 조합이 아래와 같은 경우가 되며, 이를 메소드 오버로딩을 활용하여 처리한다.
      - 정수-정수, 정수-실수, 실수-정수, 실수-실수
    - 결과는 소숫점 이하 둘째자리까지 출력한다.
  - input and output examples

```
Enter the base of triangle: 5
Enter the height of triangle: 1.4
The area is 3.50
```

```
Enter the base of triangle: 5
Enter the height of triangle: 5
The area is 12.50
```

End of slide