

Method

Byeongjoon Noh

Dept. of AI/Bigdata, SCH Univ.

powernoh@sch.ac.kr

Field와 method

```
public class _01_Method {
```

```
private int year;  
private String month;  
private int day;  
  
public String name;
```

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

```
// 메소드 정의
```

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

Method (definition)
→ 멤버 함수 (Member function)

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

Method call

→ main() function

```
}
```

Method의 구성

```
public static double setSize(int w, int i) {  
    double result = (w * i / 2.);  
    return result;  
}
```

Method의 구성

접근 지정자	반환형	함수의 이름	매개변수
<code>public</code>	<code>static</code>	<code>double</code>	<code>setSize(int w, int i)</code>
			{
			<code>double</code> result = (w * i / 2.);
			<code>return</code> result;
			}
			함수값 반환을 위한 예약어

접근 지정자

public

- 이 메소드는 모든 클래스로부터 접근 가능하다.

private

- 클래스 내부에서만 접근이 가능하다.

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

```
private static void sayHello2() {  
    System.out.println("This is sayHello() method");  
}
```

반환형

메소드로부터 반환되는 데이터의 타입

- int형 반환 → int
- double형 반환 → double
- float형 반환 → float
- ...
- 아무 반환값이 없을 때 → void

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

```
public static double setSize(int w, int i) {  
    double result = (w * i / 2.);  
    return result;  
}
```

매개 변수

메소드 호출 시 해당 변수들이 호출되어 사용됨

메소드 정의/호출 방법

- 호출 시 매개변수 순서대로 대응되어 대입
- 매개 변수가 기본형 변수인 경우 값이 복사됨

매개변수는 메소드 내부에서만 활용

➔ 지역 변수

➔ 함수 밖에서 사용할 수 없음!!

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

public static double setSize(int w, int i) {
    double result = (w * i / 2.);
    return result;
}

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

    int num1 = 1;
    int num2 = 3;
    double result = setSize(1, 3);
    System.out.println(result);
}
```

매개변수

예제

// 전달값이 있는 메소드

```
public static void power(int number) { // Parameter, 매개변수
    int result = number * number;
    System.out.println(number + " 의 제곱은 " + result);
}
```

```
public static void powerByExp(int number, int exponent) {
    int result = 1;
    for (int i = 0; i < exponent; i++) {
        result *= number;
    }
    System.out.println(number + " 의 " + exponent + " 제곱은 " + result);
}
```

```
public static void main(String[] args) {
    power(2); // 2 * 2 = 4
    power(3); // 3 * 3 = 9

    powerByExp(2, 3); // 2 * 2 * 2 = 8
    powerByExp(3, 3); // 3 * 3 * 3 = 27
    powerByExp(10, 0); // 1
}
```


*Note

거듭제곱

- `Math.pow(number, exponent)`
- Return type: double

제곱근

- `Math.sqrt(number)`
- Return type: double

`pow()`와 `sqrt()`는 Math 클래스의 메소드임

중복 메소드

Method overloading

- 매개변수의 자료형 또는 반환형이 다른 경우
 - ➔ 같은 이름의 메소드를 중복하여 정의가능
 - ➔ 호출 시 매개변수의 자료형을 보고
컴파일러가 알아서 호출함

➔ “메소드 오버로딩”

```
public static int square1(int i) {  
    return i * i;  
}  
  
public static double square2(double d) {  
    return d * d;  
}  
  
public static void main(String[] args) {  
    int i = 10;  
    double d = 20;  
  
    System.out.println(square1(i));  
    System.out.println(square2(d));  
}
```



```
public static int square(int i) {  
    return i * i;  
}  
  
public static double square(double d) {  
    return d * d;  
}  
  
public static void main(String[] args) {  
    int i = 10;  
    double d = 20;  
  
    System.out.println(square(i));  
    System.out.println(square(d));  
}
```

메소드에 Array 전달

Array를 전달받는 메소드 정의

```
public static void addNums(int[] arr) {  
    int sum = 0;  
  
    for (int ele : arr) {  
        sum += ele;  
    }  
  
    System.out.println(sum);  
}  
  
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    int[] arr = {1, 2, 3, 4, 5, 6};  
  
    addNums(arr);  
}
```

메소드에 Array 전달

2차원 Array를 전달받는 메소드 정의

```
public static void addNums(int[] arr) {
    int sum = 0;

    for (int ele : arr) {
        sum += ele;
    }

    System.out.println(sum);
}

public static void addNums(int[][] arr2D) {
    int sum = 0;

    for (int inner[] : arr2D) {
        for (int ele : inner) {
            sum += ele;
        }
    }

    System.out.println(sum);
}

public static void main(String[] args) {
    // TODO Auto-generated method stub
    int[] arr = {1, 2, 3, 4, 5, 6};
    int[][] arr2D = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 10}};

    addNums(arr);
    addNums(arr2D);
}
```

메소드에 Array 전달

Java에서 array의 전달은 “레퍼런스 복사” 로 이루어 짐

- 함수 내에서 값이 바뀌면 main에서도 값이 바뀜

```
public static int sum(int[] array) {  
    int total = 0;  
    for (int i = 0; i < array.length; i++) {  
        total += array[i];  
    }  
    return total;  
}  
  
public static void main(String[] args) {  
    int[] nums = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
  
    System.out.println(sum(nums)); // 55  
}
```

참조를 인자로 받음



Array를 반환하는 메소드

메소드에서 Array를 반환하였다면, 그 값은 레퍼런스가 아닌
별도의 메모리에 할당됨

```
public static void main(String[] args) {  
    int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
    int[] b = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };  
  
    int[] c = add(a, b);  
  
    for (int num : c)  
        System.out.println(num);  
}  
  
static int[] add(int[] array1, int[] array2) {  
    int[] array3 = new int[array1.length];  
  
    for (int i = 0; i < array1.length; i++) {  
        array3[i] += array1[i] + array2[i];  
    }  
    return array3;  
}
```



가변 길이 인수

매개변수로 몇 개의 인자를 전달할 지 정할 수 없을 때 활용

```
public static void addNumsVariableLength(int... nums) {  
    int sum = 0;  
    for (int num : nums) {  
        sum += num;  
    }  
    System.out.println(sum);  
}  
  
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    addNumsVariableLength(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);  
    addNumsVariableLength(1, 2, 3);  
}
```

End of slide