메소드(method)

메소드

메소드

메소드

→ 수학의 함수와 비슷하며 호출을 통해 사용
→ 타언어에서 함수와 동일한 기능
→ 메소드 호출 시 전달 값이 있거나 없을 수 있으며, 호출하게 되면 내부에 작성된 연산을 수행하게 되며, 연산 후 결과값은 있거나 없을 수 있다.

메소드

메소드의 형태

```
[접근제어지시자] 반환형 메소드명 (매개변수){
        실행코드;
}
```

예)

```
public void printMsg(){
    System.out.println(“HelloJava”);
}
```

메소드

메소드의 접근제어지시자(접근제한자)

구분	해당 클래스 내부	같은 패키지	후손 클래스	전체
public	O	O	O	O
protected	O	O	O	
default	O	O		
private	O			

메소드

메소드의 반환형

구분	설명
void	반환 값이 없을 경우
기본자료형	반환 값이 8가지 기본 자료형일 경우
참조자료형	반환 값이 참조형 자료형일 경우(String, 사용자정의)
배열	기본형 배열, 참조형 배열 모두 가능

메소드

메소드의 매개변수

구분	설명
()	매개변수가 없는 것을 의미함
기본자료형	기본형 매개변수 사용 시 값을 복사하여 전달 메소드에서 해당 값을 변경해도 원래 값은 변경 X call by value
참조자료형	참조자료형, 배열을 매개변수로 전달 시에는 데이터의 주소값이 전달 메소드에서 해당 값을 변경하면 원래 값도 같이 변경 call by reference
배열	

※ 매개변수의 개수에는 제한이 없으며, 여러 개 인 경우 ,를 통해서 구분

메소드

메소드 호출 1. 같은 클래스 안의 메소드 호출

```
public class MethodTest1 {
    public void method1(){
        method2();                  //기본적으로 클래스 내부의 메소드 호출 시 메소드이름();
        method3(10);                //method3의 경우 전달인자로 정수 1개를 반드시 포함해야함
                                    //전달 시 숫자, 또는 변수의 값을 통해서도 전달 가능
    }

    public void method2(){
        System.out.println("method2() 메소드 호출 완료");
    }

    public int method3(int num){
        System.out.println("method3() 메소드 호출 완료");
        System.out.println("method3 호출 시 전달 받은 인자의 값 : "+num);
        return 2*num;               //return 은 뒤에 데이터를 반환하면서 메소드를 종료하는 명령어
                                    //반환타입이 void가 아닌 경우 return은 반드시 있어야 함
    }

}
```

메소드

메소드 호출 2. 다른 클래스의 메소드 호출

```
import kh.java.func.MethodTest1;                    //다른패키지의 클래스인 경우 import 를 한다.
public class MethodTest2{
    public void method1(){
        //기능제공 클래스의 메소드를 부르는 방식과 동일
        MethodTest1 mt = new MethodTest1();
        mt.method2();
        mt.method3(100);
        //반환타입이 void인경우 return을 생략해도 무관
    }
}
```

메소드 오버로딩

메소드 오버로딩

메소드 오버로딩

→ 한 클래스 내에서 매개변수(파라미터) 선언부가 다르고, 이름이 같은 메소드를 여러 개 정의하는 것

메소드 오버로딩 성립조건

1. 메소드의 이름이 같아야 한다.

2. 매개변수 선언부가 달라야 한다.
   - 매개변수 타입, 개수, 순서

※ 오버로딩 주의사항

1. **매개변수에서 사용하는 변수의 이름은 상관하지 않는다.(오버로딩 된 곳에서 동일하게 사용 가능)**
2. **리턴타입은 오버로딩에 영향을 주지 않는다.(같거나 다르거나 무관)**

메소드 오버로딩

메소드 오버로딩 예1

```
public class Sum{
    public int sum(int num1, int num2){
        int sum = num1 + num2;
        return sum;
    }
    public int sum(int num1, int num2, int num3){
        int sum = num1 + num2 + num3;
        return sum;
    }
    public int sum(int su1, int su2){
        int sum = num1 + num2;
        return sum;
    }
    public void sum(int num1, int num2){
        int sum = num1 + num2;
        return sum;
    }

}
```

sum으로 메소드 이름은 동일하지만 매개변수의 수가 3개이므로 오버로딩 가능

매개변수가 정수형 2개로 동일하면, 매개변수로 사용하는 변수의 이름이 다르더라도 불가능

리턴타입이 달라지더라도 오버로딩에는 영향을 줄 수 없으므로 불가능

메소드 오버로딩

메소드 오버로딩 예2

```
public class Print{
    public void print(String msg){
        System.out.println(msg);
    }
    public void print(int num){
        System.out.println(num);
    }
    public void print(String msg, int num){
        System.out.println(msg+num);
    }
    public void print(int num, String msg){
        System.out.println(msg+num);
    }
}
```

매개변수의 개수는 1개로 동일하지만, 매개변수의 데이터 타입이 다르므로 가능

매개변수의 개수가 다르기 때문에 가능

매개변수의 개수도 같고 이름도 모두 같지만 데이터타입 순서가 다르므로 가능

※ 오버로딩은 결국 동일한 메소드 명을 가지고 있는 메소드들의 매개변수가 어떠한 데이터타입이 어떤 순서로 있는 지만 중요하며, 매개변수의 내용이 다르면 가능