메소드 오버로딩

메소드 오버로딩(overloading)

- 클래스 내에 **같은 이름의 메소드를 여러 개** 선언하는 것
- **하나의 메소드 이름**으로 **여러 기능**을 담는다.
- 매개변수의 타입, 개수, 순서 중 하나가 달라야 함.
- 리턴타입은 상관 없다.

메소드 오버로딩(overloading)

- **리턴 타입만 다를 경우** 에러발생!
- 오버로딩 조건이 성립되지 않음.
- 어떤 메소드를 호출하는지 **특정할 수 없다!**

```
public static int getTen() { return 10; }
public static double getTen() { return 10.0; }
```

메소드 오버로딩(overloading)

- 아래와 같이 매개변수의 타입, 개수, 순서 중 적어도 하나가 달라야 정확히 **어떤 메소드를 호출하는지 특정할 수 있기 때문**

```
public static void method() {}
public static void method(int a) {}
public static void method(int a, int b) {}
public static void method(int b, int c) {}  // 변수명만 다른것은 의미없음
public static void mehtod(String a) {}
public static void mehtod(String a, String b) {}
public static void method(int a, String b) {}
public static void method(String a, int b) {}
```

오버로딩이 필요한 이유

- **매개값을 다양하게 받아 처리**할 수 있도록 하기 위함.
- 오버로딩의 가장 대표적인 예는 **System.out.println()**

```
public static void main(String args[]) {

    plus(10, 20);
    plus(10.5, 20.3);

}

public static void plus(int a, int b) {System.out.println(a+b);}
public static void plus(double a, double b) {System.out.println(a+b);}
```

가변인자

- 매개 변수의 **개수를 모를 경우** 사용됨
- 매개변수를 여러 개 보내야 할 경우 사용 (배열도 가능)

```
int sum1(int[] values) { }

int[] values = {1, 2, 3};
int result = sum1(values);
int result = sum1(new int[] {1,2,3});
```

```
int sum2(int ... values) { }

int result = sum2(1, 2, 3);
int result = sum2(1, 2, 3, 4, 5);
```