메소드

메소드(method)

- 메소드(method)는 **코드를 재사용** 할 수 있게 해준다.
- 특정 로직을 메소드로 작성해 놓고 호출만 하면 언제든지 사용할 수 있다.
- 따라서, 중복되는 로직을 작성할 필요가 없다.

메소드의 정의와 호출

- 1~10까지 합하는 sum()이라는 메소드를 작성하고 호출 해보자
- **자주 사용하는 로직을 메소드**로 만들어 두고 언제든지 호출

```
public static void sum() {
    int sum = 0;
    for(int i=1; i<=10; i++) {
        sum += i;
    }
    System.out.println("합: " + sum);
}

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

메소드의 매개변수

- 이전 예제를 수정하여 **특정 정수까지** 합하도록 해보자
- 이제는 특정 정수까지 합하도록 변경되어 조금 더 **다양한 상황에서 메소드를 호출할 수 있게 되었다!**

```
public static void sum(int limit) {
    int sum = 0;
    for(int i=1; i<=limit; i++) {
        sum += i;
    }
    System.out.println("합: " + sum);
}

public static void main(String args[]) {
    sum(100);
}
```

복수 매개변수(parameter)

- 이전 예제를 2개의 매개변수를 받아 그 범위의 합을 출력하도록 수정해보자

```
public static void sum(int start, int end) {
    int sum = 0;
    for(int i=start; i<=end; i++) {
        sum += i;
    }
    System.out.println("합: " + sum);
}

public static void main(String args[]) {
    sum(50, 100);
}
```

return

- 메서드에서 출력하지않고 **결과값을 리턴** 받아서 출력해보자

```
public static int sum(int start, int end) {
    int sum = 0;
    for(int i=start; i<=end; i++) {
        sum += i;
    }
    return sum;
}

public static void main(String args[]) {
    int result = sum(50, 100);
    System.out.println("합 : " + result);
}
```

return

- 리턴타입을 지정했다면 반드시 return이 있어야 한다.
- 리턴타입이 없을(void)경우 return은 break와 유사하게 메서드를 빠져나오는 역할을 한다.
- 기본타입 뿐만 아니라 참조타입도 리턴 가능.