06장 함수

자바의 함수

◈ 함수(Method)

어떤 작업을 수행하기 위한 명령문의 집합

어떤 값(파라미터)을 받아서 처리하고, 그 결과(리턴값)를 되돌려 준다.

파라미터를 받거나 받지 않으며, 리턴값 또한 없을 수도 있다.

반복하여 사용하는 코드에 대해 함수로 만들면, 중복 코드를 막을 수 있다.

자바의 함수

◈ 함수의 정의

```
접근제어자 [지정예약어] 리턴타입 함수명(매개변수목록)[throws 예외클래스들]
{
    실행문;
}
```

접근 제어자 : 해당 함수에 접근할 수 있는 범위를 명시

리턴 타입 : 함수가 모든 작업을 마치고 반환하는 데이터의 타입을 명시 (리턴이 없다면 void 입력)

함수명 : 함수를 호출하기 위한 이름을 명시

매개변수 목록 : 함수 호출 시 전달되는 인수의 값을 저장할 변수들을 명시

자바의 함수

◈ 접근 제어자

public : 모든 클래스에서 접근 가능

private : 해당 클래스에서만 접근 가능

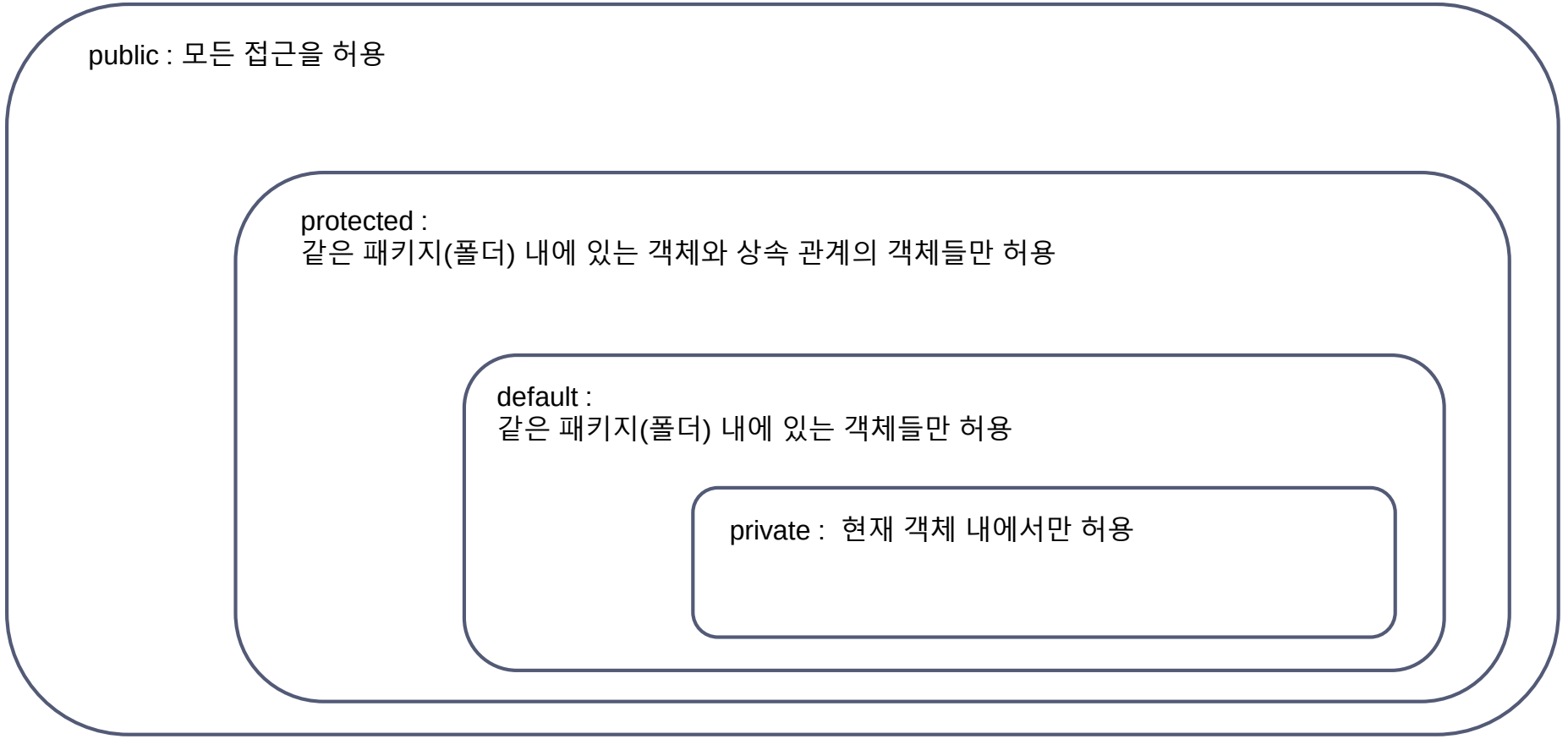
protected : 해당 클래스 및 해당 클래스를 상속받는 클래스에서만 접근 가능

지정하지 않음 : (default)같은 패키지 내에서만 접근 가능

접근 제어자를 잘 이용하면 코딩 실수를 줄이고, 기타 위험요소를 제거할 수 있는 장점이 있다.

자바의 함수

◈ 접근 제어자


public : 모든 접근을 허용
protected :
같은 패키지(폴더) 내에 있는 객체와 상속 관계의 객체들만 허용
default :
같은 패키지(폴더) 내에 있는 객체들만 허용
private : 현재 객체 내에서만 허용

자바의 함수

◈ 함수 호출과 프로그램의 실행 흐름

```
public class Operator {
    public static void main(String[] args) {
        System.out.print("1부터 100까지 다 더하면 "); ①
        System.out.print(sum(1, 100)); ②
        System.out.print(" 입니다."); ④
    }
    private static int sum(int fromNum, int toNum) {
        int sumNum = 0;
        for(int i = fromNum; i <= toNum; i++) {
            sumNum += i;
        }
        return sumNum;
    }
}
```

③

→ 1부터 100까지 다 더하면 5050 입니다.

자바의 함수

◈ 메서드 오버로딩(Overloading)

한 클래스 내에 같은 이름의 메서드를 중복하여 정의하는 것

메서드의 이름은 같으나 파라미터 변수의 수와 타입이 다르면 중복 정의가 가능하다.

```
public static void main(String[] args){
    int sum = add(2, 3);
    int twice = add(2);
    double sumDouble = add(2.1, 3.2);
}

public static int add(int firstNum, int secondNum) {
    return firstNum + secondNum;
}
public static int add(int oneNum) {
    return oneNum * 2;
}
public static double add(double firstNum, double secondNum) {
    return firstNum + secondNum;
}
```

← 매개변수의 개수가 다름

← 매개변수의 타입이 다름

리턴의 타입이 달라도 된다.

자바의 함수

◈ 재귀 함수(Recursion Function)

함수 내부에서 해당 함수를 스스로 호출함으로써 재귀적으로 문제를 해결하는 함수이다.

for문이나 while문 없이 함수를 반복 호출함으로써 코드가 간결해 진다.

```
public static void main(String[] args) {
    Function(5);
}

public static void Function(int num) {
    if (num == 0) {
        return;
    } else {
        System.out.println("반갑습니다");
        Function(num - 1);
    }
}
```