3. 변수와 메서드

● 변수: 선언위치에 따른 변수의 종류

변수의 종류	선언 위치	생성 시기
클래스 변수 (class variable) - static이 붙은 멤버 변수	클래스 영역 (멤버 변수이다)	클래스가 메모리에 올라갈 때. - 인스턴스를 생성하지 않고도 사용 가능하다. '클래스이름.클래스변수'와 같은 형식으로 사용한다. - 클래스변수는 모든 인스턴스가 공통된 저장공간(변수)를 공유하게 된다. - 한 클래스의 모든 인스턴스들이 공통적인 값을 유지해야 하는 속성의 경우, 클래스변수로 선언한다. - 클래스가 메모리에 로딩(loading - 참조변수의 선언이나 객체의 생성과 같이 클래스의 정보가 필요할 때, 클래스는 메모리에 로딩된다)될 때 생성되어 프로그램이 종료할 때까지 유지되며, public을 앞에 붙이면 같은 프로그램 내에서 어디서나 접근할 수 있는 전역변수(global variable)성격을 갖는다. - ex) Card.width가 클래스변수일 경우, c1과 c2는 클래스변수인 width을 공유하기 때문에, c1의 width(c1.width)을 변경하면 c2의 width(c2.width)값도 바뀐 것과 같은 결과를 얻는다. 이는 Card.width, c1.width, c2.width가 모두 같은 저장공간을 참조하므로 항상 같은 값을 갖기 때문이다.
인스턴스 변수 (instance variable) - static이 붙지 않은 멤버 변수		클래스의 인스턴스가 생성되었을 때. - 인스턴스 변수의 값을 읽어오거나 저장하기 위해선 먼저 인스턴스를 생성해야 한다. - 인스턴스는 독립적인 저장공간을 가지므로 서로 다든 값을 가질 수 있다. -인스턴스마다 고유의 상태를 유지해야 하는 경우, 인스턴스 변수로 선언한다.
지역 변수 (local variable)	클래스 이외의 영역 (메서드, 생성자, 초기화 블럭 내부)	변수 선언문이 수행되었을 때. - 메서드 내에 선언되어 메서드 내에서만 사용 가능하며, 메서드가 종료되면 소멸되어 사용할 수 없게 된다. - for문 또는 while문의 블럭 내에 선언된 지역 변수는, 지역변수가 선언된 블럭{} 내에서만 사용 가능하며, 블럭{}을 벗어나면 소멸되어 사용할 수 없게 된다.

- 멤버 변수: 인스턴스변수는 인스턴스가 생성될 때마다 생성되므로 인스턴스마다 각기 다든 값을 유지할 수 있지만, 클래스 변수는 모든 인스턴스가 하나의 저장공간을 공유하므로, 항상 공통된 값을 갖는다.

● 메서드

- 메서드(method): 특정 작업을 수행하는 일련의 문장들을 하나로 묶은 것. 기본적으로 수학의 함수와 유사하며, 어떤 값을 입력하면 이 값으로 작업을 수행해서 결과를 반환한다. 하지만 함수와 달리 메서드는 입력값 또는 출력값(결과값)이 없을 수도 있으며, 심지어는 입출력값이 모두 없을 수도 있다. 메서드 내부에서 어떤 과정을 거쳐 결과를 만들어내는지 몰라도 돼서 메서드를 내부가 보이지 않는 블랙박스(black box)라고도 한다.
- 메서드를 사용하는 이유:

1) 높은 재사용성(reusability): 한번 만들어 놓은 메서드는 몇 번이고 호출할 수 있으며, 다른 프로그램에서도 사용이 가능하다.
2) 중복된 코드의 제거: 반복되는 문장들을 하나의 메서드로 작성해 놓으면, 반복되는 문장들 대신 메서드를 호출하는 한 문장으로 대체할 수 있다. 전체 소스 코드의 길이도 짧아지고, 변경사항이 발생했을 때 수정해야 할 코드의 양이 줄어들어 오류가 발생할 가능성도 함께 줄어든다.
3) 프로그램의 구조화: 큰 규모의 프로그램에서는 문장들을 작업단위로 나눠서 여러 개의 메서드에 담아 프로그램의 구조를 단순화시키는 것이 필수적이다. main메서드는 프로그램의 전체 흐름이 한눈에 들어올 정도로 단순하게 구조화하는 것이 좋다. 그래야 나중에 프로그램에 문제가 발생해도 해당 부분을 쉽게 찾아내서 해결할 수 있다. 처음에 프로그램을 설계할 때 내용이 없는 메서드를 작업 단위로 만들어놓고, 하나씩 완성해나는 것도 프로그램을 구조화하는 좋은 방법이다.

▶ 메서드를 정의한다는 것은 선언부(header, 머리)와 구현부(body, 몸통)을 작성한다는 것이다.
☞ 메서드 선언부(method declaration, method header): 변환타입(출력), 메서드의 이름과 매개변수 선언(입력)으로 구성되어 있다.
- 반환타입(return type): 메서드의 작업 수행 결과(출력)인 반환값(return value)의 타입을 적는다. 반환값이 없는 경우 반환타입으로 void을 적는다.
- 메서드의 이름(method name): 특정 작업을 수행하므로 동사인 경우가 많다.
- 매개변수 선언(parameter declaration): 매개변수는 메서드가 작업을 수행하는데 필요한 값들(입력)을 제공받기 위한 것이며, 필요한 값의 개수만큼 변수를 선언하며 각 변수 간의 구분은 쉼표','를 사용한다. 매개변수 또한 메서드 내부에 위치한 지역 변수이다.

☞ 메서드의 구현부(method body): 메서드를 호출했을 때 수행될 문장들을 넣는다.
- return문: 실행 중인 메서드를 종료하고 호출한 곳으로 되돌아간다. 메서드의 반환타입이 'void'가 아닌 경우, 구현부{} 안에 'return 반환값;'이 반드시 포함되어 있어야 한다. 이 문장은 작업을 수행한 결과인 반환값을 호출한 매서드로 전달하는데, 이 값의 타입은 반환타입과 일치하거나 적어도 자동 형변환이 가능한 것이어야 한다. return문은 단 하나의 값만 반환할 수 있다. (원래는 반환값의 유무에 관계없이 모든 메서드에 적어도 하나의 return문이 있어야 하지만 void일 경우 return문이 없어도 문제가 되지 않았던 이유는 컴파일러가 메서드의 마지막에 return;을 자동적으로 추가해주었기 때문이다. 하지만 반환타입이 void가 아닐 경우엔 반드시 return문이 있어야 하고 없으면 컴파일 에러 error: missing return statement가 발생한다).
- 반환값(return value): 주로 변수가 온다. 수식이 올 경우 수식을 계산한 결과가 반환된다. 반환타입이 일치할 경우 반환값으로 메서드를 호출해서 그 결과를 받아서 반환할 수도 있다. 간단한 메서드일 경우 if대신 조건연산자를 사용해서 반환값을 작성할 수도 있다.

```
// Example
static int abs(int x) {
    return x > 0 ? x : -x;
}
```

- 지역변수(local variable) :메서드 내에 선언된 지역 변수들은 그 메서드 내에서만 사용할 수 있어서 서로 다른 메서드라면 같은 이름의 변수를 선언해도 된다.
- 매개변수의 유효성 검사: 적절하지 않은 값이 매개변수를 통해 넘어온다면 매개변수의 값을 보정하던가, 보정하는 것이 불가능하다면 return문을 사용해서 작업을 중단하고 호출한 메서드로 되도록아가야 한다.

```
// Example
static float divide(int x, int y) {
    // 작업을 하기 전에 나누는 수 y가 0인지 확인
    if (y == 0) {
        System.out.println("0으로 나눌 수 없습니다.");
        return 0;
    }
    return x / (float)y;
}
```

▶ 메서드의 호출: 메서드를 호출할 때 메서드이름(값1, 값2,...) 식으로 하며, 괄호()안에 지정해준 값들을 인자(argument)또는 인수라고 하는데, 인자의 개수와 순서는 호출된 메서드에 선언된 매개변수와 일치해야 한다. 인자는 메서드가 호출되면서 매개변수에 대입되므로, 인자의 타입은 매개변수의 타입과 일치하거나 자동 형변환이 가능한 것이어야 한다 (ex. 인자: long, 매개변수: double; 인자: int, 매개변수: long)

☞ 메서드의 실행흐름: 같은 클래스 내에 메서드끼리는 참조변수를 사용하지 않고도 서로 호출이 가능하지만, static 메서드는 같은 클래스의 인스턴스 메서드를 호출할 수 없다. 먼저 해당 클래스의 인스턴스를 생성한 다음 참조변수를 통해서 인스턴스 메서드를 호출해야 한다. 먼저 메인메서드에서 메서드를 호출하면, 호출시 지정한 인자가 메서드의 매개변수에 각각 복사(대입)되고, 메서드의 괄호{}안에 있는 문장들이 순서대로 수행되며, 매서드의 모든 문장들이 실행되거나 return문을 만나면, 호출한 메서드(메인 메서드)로 되돌아와서 이후의 문장들을 실행한다. 메서드의 결과를 변수에 저장한다면 이 변수 역시 반환값과 같은 타입이거나 반환값이 자동 형변환되어 저장될 수 있는 타입이어야 한다.

Resource: 자바의 정석