

자료구조: 2022년 1학기 [강의]

객체 (Object)



© J.-H. Kang, CNU

강지훈

jhkang@cnu.ac.kr

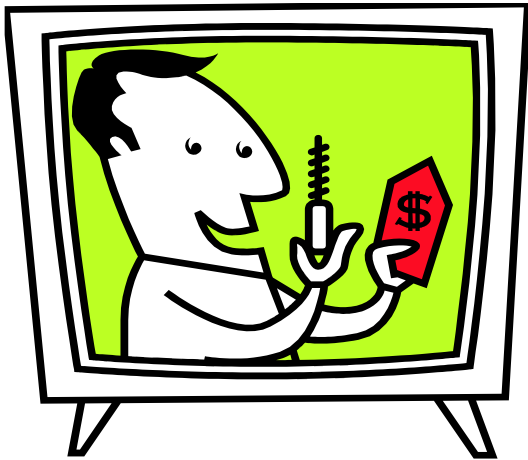
충남대학교 컴퓨터융합학부

객체의 사용자와 구현자



□ 객체 사용자는 어떻게 ?

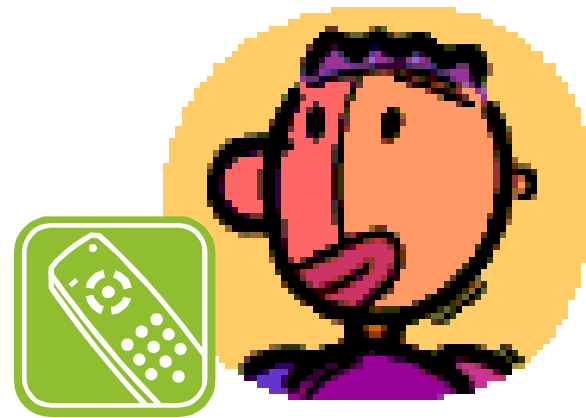
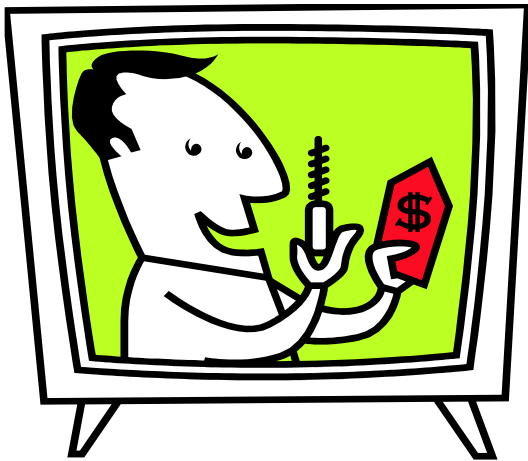
- 객체의 내부 구조나 속성을 알고 싶을까?
 - TV는 그 내부가 매우 복잡한 객체
 - TV 사용자는 TV 객체 내부의 복잡한 전자적/기계적 구조나 속성을 알고 싶을까?



□ 객체 사용자는 어떻게 ?

■ 사용자는 단지 TV 객체를 쉽게 사용하기를 원할 뿐!

- 전원 스위치로 켜고 끈다
- 볼륨 스위치로 소리 크기를 조절한다
- 채널 스위치로 원하는 채널을 선택한다



□ 객체 사용자는 어떻게 ?

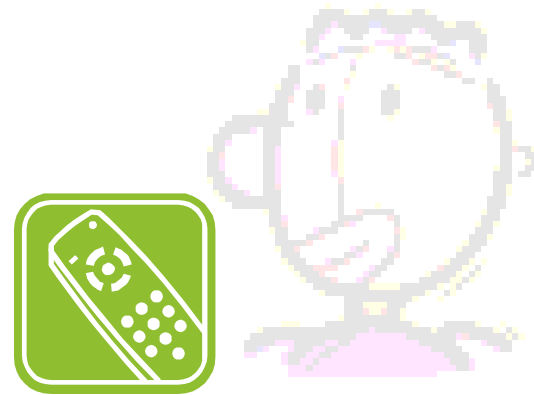
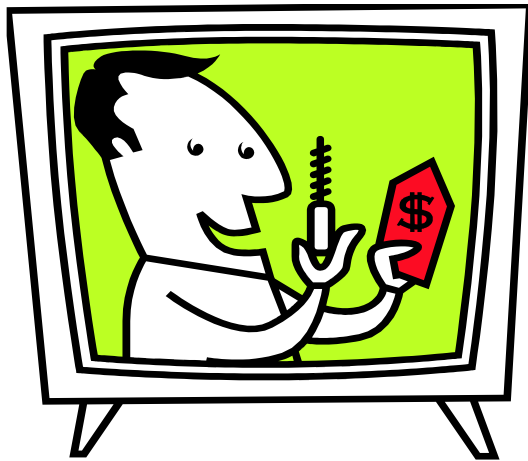
■ 리모컨의 역할은?

- 사용자에게 TV 객체의 내부를 알 필요 없이 편리하게 TV 시청을 하게 해 준다.
- 리모컨의 각각의 단추가 TV 를 쉽고 편리하게 사용하도록 만들어 준다.



□ 객체 구현자는 어떻게 ?

- 구현자는 TV 객체를 사용하기 쉽고 편하고, 성능 좋고, 싸게 만들어야 한다!
- TV 제조회사(구현자)는 사용자의 요구 사항에 맞도록 그 내부의 전자적/기계적 구조를 설계하고 만들어야 한다.
- 사용법이 바뀌지 않으면서, 지속적으로 성능을 혁신시킨다.



추상화 (Abstraction) 란?



□ 추상화 (Abstraction) 란?

■ 추상화 행위는:

- 구체적인 것을 감추는 것

■ 왜 감추는가?

- **사용자의 편의**를 위해서는 필수적!
 - ◆ 사용자에게는, TV를 어떻게 만들었는지 보다는,
 - ◆ 어떻게 하면 TV를 잘 사용할 수 있을 지가 더 중요!

□ 우리의 삶 자체가 추상화의 연속!

■ 우리의 삶을 잘 살펴 보라!

- 문명의 발전은 추상화의 과정

■ 나 혼자 무엇을 얼마나 할 수 있을까?

- 왜 협업이 중요한가?
- 상대방이 만들어서 내게 제공하는 결과가, 무엇을 가지고 어떻게 만들어졌는지를, 내가 자세히 알아야만 협업이 가능할까?

■ 지금 내게:

- 스마트폰이 없다면?
- 자바 프레임워크가 없다면?

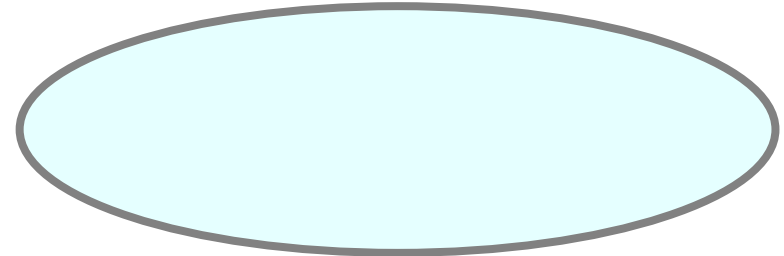
Java 에서의 객체



□ Java 에서의 객체의 정의

- 학생 객체를 만들어 사용하고 싶다

Student:



□ 객체를 정의할 때, 가장 먼저 생각할 점은?

■ 객체의 사용법:

- 객체에게 무슨 일을 시킬 것인가?

학번을 부여한다

학번을 얻어낸다

성적을 부여한다

성적을 얻어낸다

합불 판정을 얻어낸다

Student:



□ 객체는 언제나 리모컨으로만!

■ Student가 TV라면, 리모컨은?

학번을 부여한다

학번을 얻어낸다

성적을 부여한다

성적을 얻어낸다

합불 판정을 얻어낸다

Student:



□ 객체 사용법은 공개함수로!

■ 공개함수

(public functions / public methods)

- 리모컨 단추 하나가 한 개의 공개함수

```
void setStudentNo (String newStudentNo)
```

```
String studentNo ()
```

```
void setScore (int newScore)
```

```
int score ()
```

```
char grade ()
```

Student:



□ 객체는 Java 에서 어떻게 표현?

■ 일반 변수를 다시 생각해보자.

- `double d ;`

- 변수 `d` 역시 하나의 객체로 생각할 수 있다.

- 그 자료형이 `double` 이다.

자료형: 정보(데이터)를
보관할 방법이나 처리 방
법에 관한 설계도

- 직접 연관이 있는 연산
(변수 `d` 를 사용하여 할 수 있는 일):

 - ◆ `+, -, *, /` 과 같은 사칙연산

 - ◆ `>, <, ==, !=` 과 같은 비교연산

- 이러한 연산이 묵시적으로 (implicitly) 정의되어 있다.



□ 일반적인 객체의 표현은?

- kim 이라는 학생을 위한 학생정보는?
- 우리는 이렇게 하고 표현하고 싶다.

- **Student** kim ;
 - ◆ (cf.) double d ;

- 즉, 일반 변수처럼!

Student: kim

_studentNo: "20119999"
_score: 87
_grade: 'P'

- **kim** 은 하나의 객체 변수이다.
 - ◆ 객체는 Java 에서 변수로 표현할 수 있다.
- 그 자료형이 **Student** 이다.



□ Class: 객체의 자료형

- 그렇다면, 자료형 **Student** 는 Java 에서 어떻게 표현?
 - 다양한 객체의 형태가 미리 정의되어 존재할 수는 없다.
 - ◆ int, double, char 와 같이 가장 기본적인 것만 미리 존재한다.
 - 우리가 자료형을 만들어 (정의하여) 사용한다.

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    .....  
}
```

기본 자료형: 정수(int), 실수(float, double), 문자(char) 등은 간단한 형태의 정보 저장 방법이나 처리 방법에 관한 설계도.

추상 자료형: 정보 저장 방법이나 처리 방법의 설계도라는 기본 개념은 마찬가지이나, 그 방법이 단순하지 않다.

- 이러한 "객체를 위한 자료형"을 **class** 라고 한다.



□ 객체는 Class 로 정의한다

- class 는 하나의 자료형 (추상자료형)
 - 왜 추상 자료형이라고 할까?
- class 에는 “객체의 속성” 이 포함되게 된다.
 - 나중에 좀 더 자세히.....

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    .....  
}
```

□ 객체의 자료형은 class 이름으로

■ Class 에 이름을 부여하자.

- Student 객체들을 위한 Student class

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    .....  
}
```

□ Class 의 선언

- 그러므로, 다음과 같이 선언된다.

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    .....  
}
```

□ 객체의 속성은 **private** 으로

- 객체의 속성은 왜 **private** 으로 선언할까?
 - 속성은 TV 객체의 부품과 같은 것
 - TV 내부에 사용된 부품을, 사용자가 알아야만 TV 를 볼 수 (작동시킬 수) 있다면?

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    .....  
}
```

□ 완성된 class 인가?

■ 아직은.....

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
    ..... // 이곳에는?  
}
```

■ 객체는 해야 할 일이 있다.

□ 객체가 할 일은 어디에 어떻게?

- TV 객체의 리모컨 단추가 눌려졌을 때 해야 할 일:

```
public class Student {  
    private String _studentNo ;  
    private int _score ;  
    private char _grade ;  
    ..... // 이곳에 함수를 선언하고 구현한다  
}
```

- 어디에?
 - Class 안에 선언한다.
- 어떻게?
 - 함수 형태로 표현 한다.
 - 리모컨 단추 하나가 한 개의 함수.

□ 객체 변수 (Object Variable)

- 이제 우리는 일단은, 객체 변수를 만들어 사용할 수 있다.
- 일반 변수의 선언 방법과 마찬가지로.

```
public class Student {  
    private String  _studentNo ;  
    private int     _score ;  
    private char    _grade ;  
    // ..... // 일단은 주석으로  
}
```

```
Student kim ;
```


□ 객체 변수가 바로 객체?

```
public class Student {  
    private String _studentNo ;  
    private int    _score ;  
    private char   _grade ;  
    // .....  
}
```

```
Student kim ;
```

■ 객체 변수 kim 의 선언으로 그림처럼 될까?

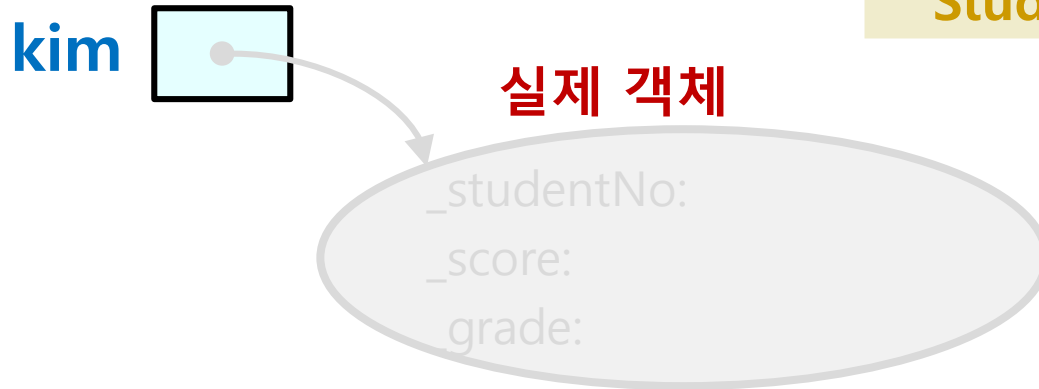
- 마치 일반 변수처럼.....
- 즉, 객체 변수의 선언으로
객체가 바로 (메모리에 실제로)
존재하게 될까?

Student: kim

_studentNo:
_score:
_grade:

□ 변수 선언으로 객체는 만들어지지 않는다!

- 객체 변수 kim 의 선언의 결과는...



```
public class Student {  
    private String _studentNo ;  
    private int    _score ;  
    private char   _grade ;  
    // .....  
}
```

```
Student kim ;
```

- 객체 변수 kim 은 단지 객체를 소유할 수 있을 뿐이다.
 - 객체는 아직 존재하지 않는다.
 - 객체 변수 kim 은 아직 소유한 객체가 없다.

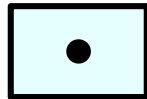
□ 객체를 생성시켜 보자 [1]

■ 객체 변수는 선언과 동시에 null 로.

- 즉, 현재 kim 은 아무 객체도 소유하고 있지 않다.

```
Student kim = null ;
```

kim



```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student() {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
    }  
  
    // ...  
}
```

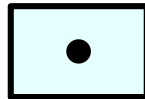
□ 객체를 생성시켜 보자 [2-1]

- 모든 객체는 **객체 생성자**와 함께 **"new"** 를 사용하여 생성시킨다.

```
Student kim = null ;  
kim = new Student() ;
```

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student() {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
    }  
    // ...  
}
```

kim



실제 객체

_studentNo: null
_score: 0
_grade: ' '

□ 객체를 생성시켜 보자 [2-2]

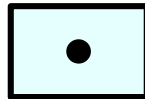
■ 객체 생성자

- 모든 객체에게 필수적인 리모컨 단추
- 이름은 Class 와 동일
- 객체를 생성하고, 초기 상태를 설정

```
Student kim = null ;  
kim = new Student() ;
```

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student() {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
    }  
    // ...  
}
```

kim



실제 객체

_studentNo: null
_score: 0
_grade: ' '

□ 객체를 생성시켜 보자 [3]

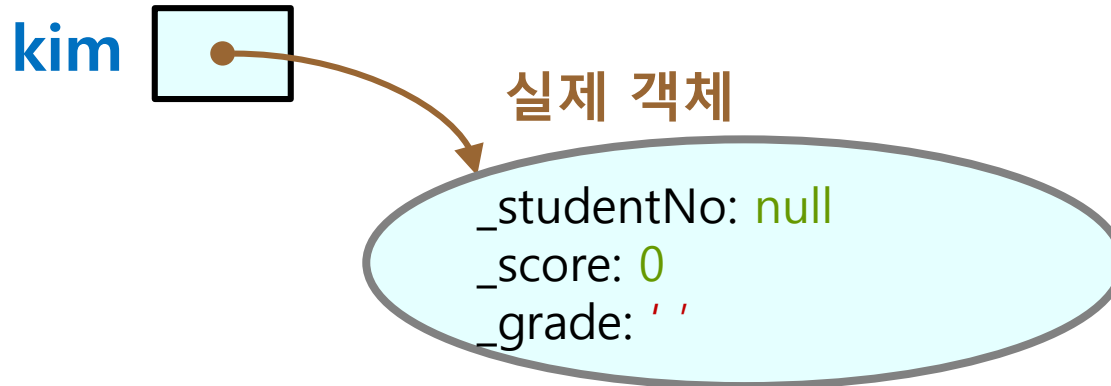
■ **new** 의 결과값은 객체의 소유권

- 메모리에서의 객체의 주소값
- 생성자 함수의 return 값

```
Student kim = null ;
kim = new Student() ;
```

```
public class Student {
    private String    _studentNo ;
    private int       _score ;
    private char      _grade ;

    // 생성자
    public Student() {
        this._studentNo = null ;
        this._score = 0 ;
        this._grade = ' ' ;
    }
    // ...
}
```



□ 객체의 생성은?

- Class는 스스로 객체 생성 방법을 가지고 있어야 한다.
 - 생성자 (Constructor): Class 안에 함수로 정의

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student () {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
  
    }  
    // ...  
}
```

반드시
class 이름과 동일

□ 객체의 생성자는 public

- 생성자는 사용자가 사용!
- 특별한 경우가 아니라면 보통은 public.

public:
객체를 외부에서
생성 해야 하므로

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student () {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
  
    }  
    // ...  
}
```



□ 객체의 생성자는 사용자에게 무엇을 제공하는가?

■ 객체를 생성하고 그 소유권을 제공한다.

● 소유권: (또는 사용권)

- ◆ 실제로는 메모리에 생성된 객체의 주소값

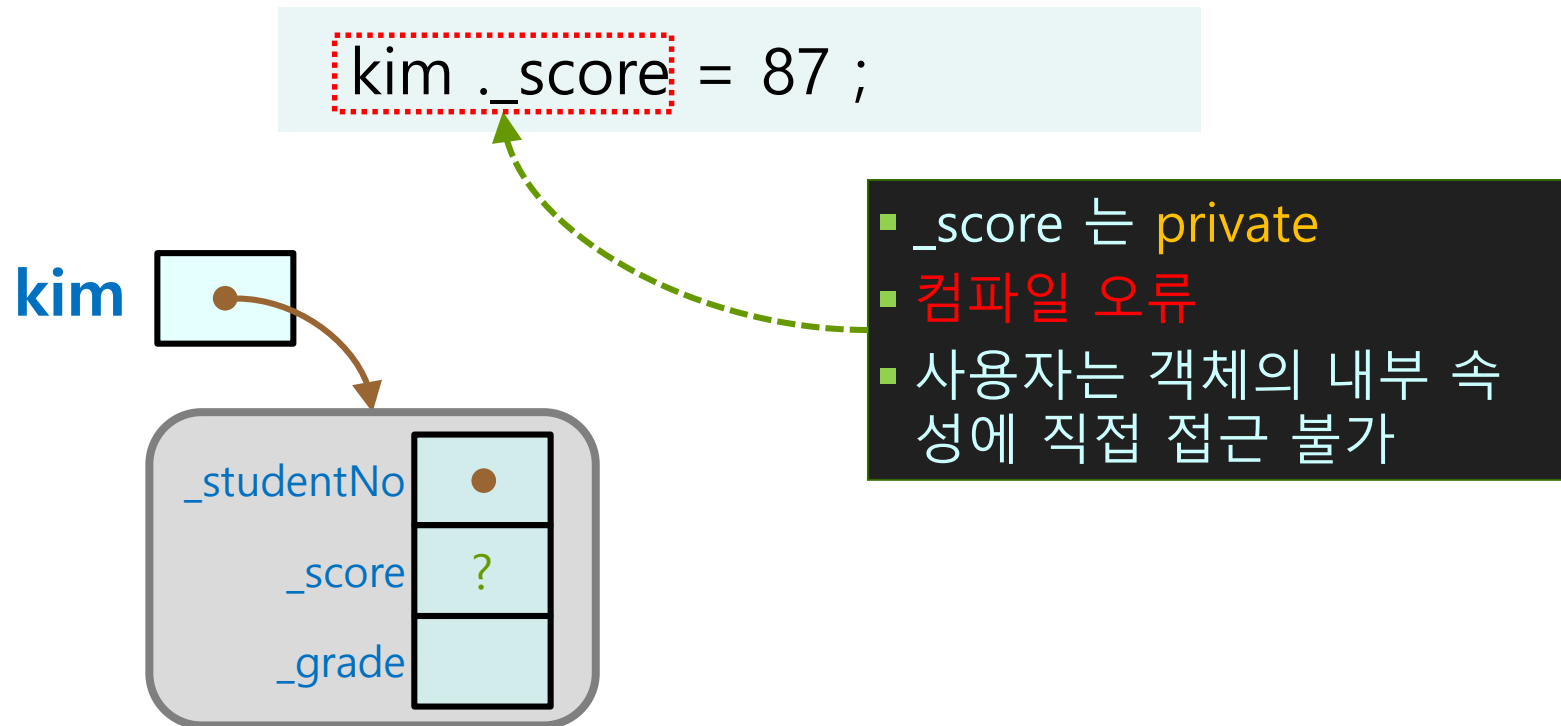
```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public Student () {  
        this._studentNo = null ;  
        this._score = 0 ;  
        this._grade = ' ' ;  
    }  
    // ...  
}
```

돌려주는 값은 있지만,
그 type (자료형)은 명
시하지 않는다

return 문도 없다

□ 객체의 사용자는 어떻게 객체를 사용?

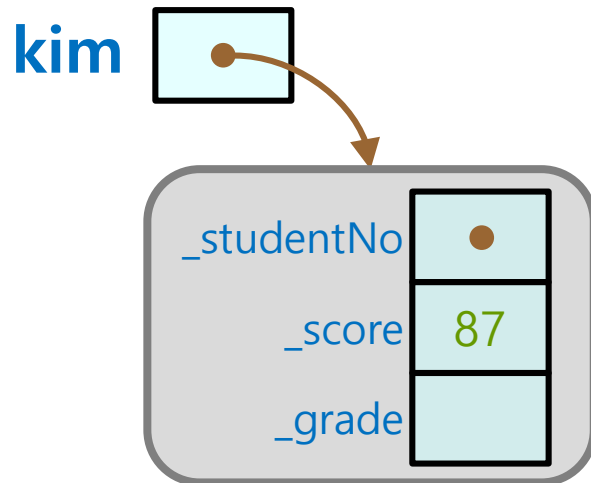
- 객체의 사용자는 객체의 **private** 속성을 알지 못한다.



□ 객체에게 일을 시킨다

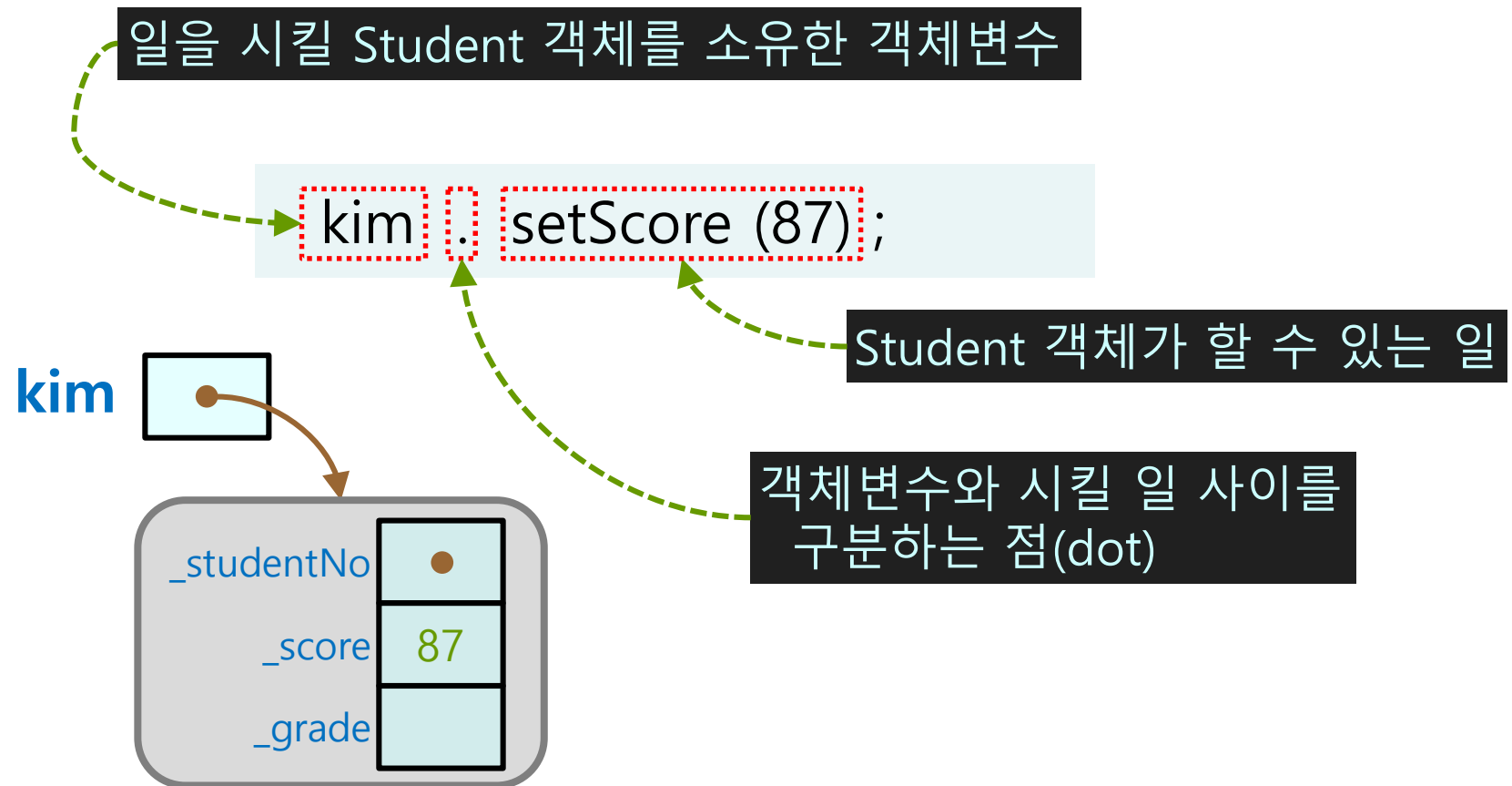
- 리모컨 단추 `setScore()` 를 누른다.
 - 객체의 사용자는 객체에 대해 직접 일을 하지 않고, 반드시 객체에게 시켜서 목적을 이룬다.
 - TV 리모콘을 생각해 보라.

```
kim . setScore (87) ;
```



□ 객체에게 일을 시킨다

- 사용자는 공개함수를 사용하여 객체에게 일을 시킨다.



객체의 Type 인 Class 의 구현 (설계)



□ Class 의 기본적인 구성

■ 속성 (attribute)

- 객체의 부품

■ 함수 (function)

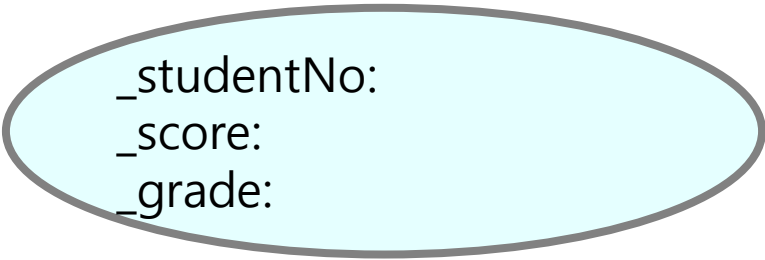
- 해야 할 일



□ 객체의 속성

- 학번 (_studentNo): 9 개 문자를 저장할 수 있는 문자열
- 성적 (_score): 정수형
- 학점 (_grade): 문자
 - 성적이 60 점 이상이면 학점은 'P' 의 값을, 60 점 미만이면 'F' 의 값을 갖는다.

Student:



_studentNo:
_score:
_grade:



□ 객체가 해야 할 일은?

- 학번의 set / get
- 성적의 set / get
- 성적 합불 평가
 - ◆ 성적을 판단하여, 통과/실패의 결과를 얻는다.
- 합불 평가의 get

Student: kim

_studentNo: "20119999"
_score: 87
_grade: 'P'

```
if (kim의 _score >= 60)
    kim의 _grade = 'P';
else
    kim의 _grade = 'F' ;
```


□ Class 안에서 객체 사용법의 정의

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public    Student() {  
        .....  
    }  
  
    public void setScore (int newScore) {  
        this._score = newScore ;  
    }  
  
    public int score () {  
        return this._score ;  
    }  
}
```

성적의 setter

성적의 getter

□ Class 안에서의 사용법의 정의

사용자의
사용법이므로
반드시 public

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public    Student() {  
        .....  
    }  
  
    public void setScore (int newScore)  
    {  
        ..... // 여기에 실제 할 일을  
    }  
  
    public int score ()  
    {  
        ..... // 여기에 실제 할 일을  
    }  
}
```



□ 객체가 할 일의 구현

```
public class Student {  
    private String    _studentNo ;  
    private int       _score ;  
    private char      _grade ;  
  
    // 생성자  
    public    Student() {  
        .....  
    }  
  
    public void setScore (int newScore)  
    {  
        this._score = newScore ;  
    }  
  
    public int score ()  
    {  
        return this._score ;  
    }  
}
```

각 사용법에 대해,
객체가 실제로 할 일

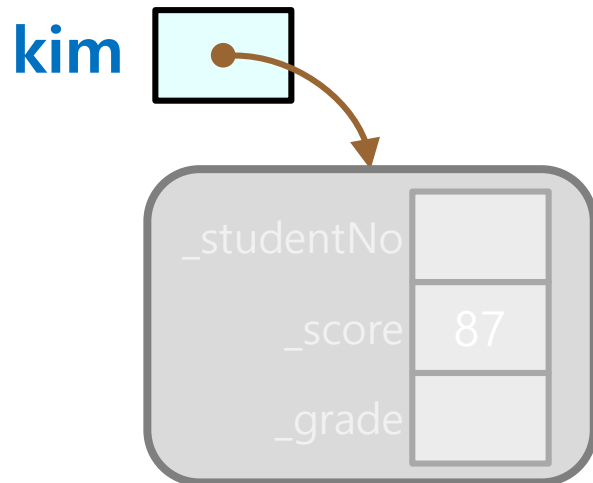
this._score = newScore ;

return this._score ;

□ Class 안에서 속성의 사용법

- 공개함수가, 자신에게 일을 시킨 (사용자가 지정한) 객체를 인지하는 방법은?

```
public void setScore (int newScore)
{
    this ._score = newScore ;
}
```



`kim . setScore (87) ;`

사용자가 객체 kim 에게
"성적을 부여하는 일"을 시켰다

□ Class 안에서 속성의 사용법

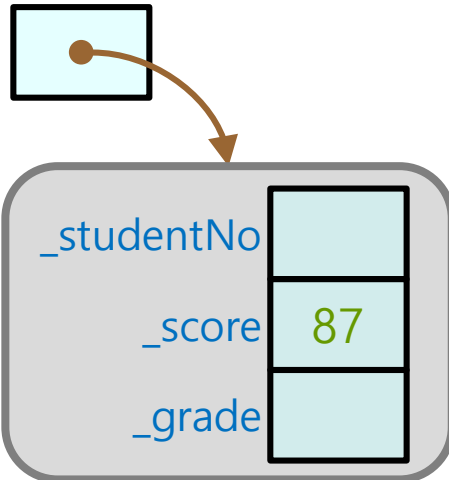
■ "this"

일을 하도록 지시 받은 객체 자신을 가리킨다

"this" 표현은 class 내부의 instance method (또는 생성자) 안에서만.

```
public void setScore (int newScore)
{
    this._score = newScore ;
}
```

kim



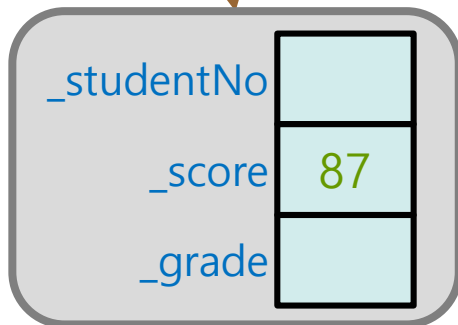
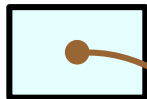
```
kim . setScore (87) ;
```

□ Class 안에서 속성의 사용법

■ this._score

```
public void setScore (int newScore )
{
    this._score = newScore ;
}
```

kim



일을 하도록 명령을 받은 객체가 가지고 있는 속성의 이름

kim . setScore (87) ;

Java 프로그램 코딩 규칙



□ 공통 사항

■ 변수 이름과 함수 이름

- 소문자로 시작한다
- 여러 단어로 구성될 경우에는 두 번째 이후의 단어는 대문자로 시작한다
 - ◆ `int numberOfStudents ;`

■ 상수

- 전체를 대문자로 한다
- 여러 단어로 구성될 경우에는 단어 사이에 '_' 를 끼워 넣는다.

■ Class 이름

- 대문자로 시작한다

□ Class 내부에서

Class 이름은
대문자로 시작

함수 이름은
소문자로 시작

class 내부에 선언
된 객체의 속성
이나 함수에는 앞
에 "this."를 붙인
다

```
public class Student {
    private String
    private int
    private char

    // 생성자
    public Student () {
        .....
    }

    public void setScore (int newScore)
    {
        this._score = newScore ;
    }

    public int score ()
    {
        return this._score ;
    }
}
```

객체의 속성 이름은
'_', 그리고
이어서 소문자로 시
작

Getter 의 이름은 얻
고자 하는 대상의 이
름으로 한다.
Setter의 이름은
getter 이름에 "set"
을 붙인다 .

End of "Object"



