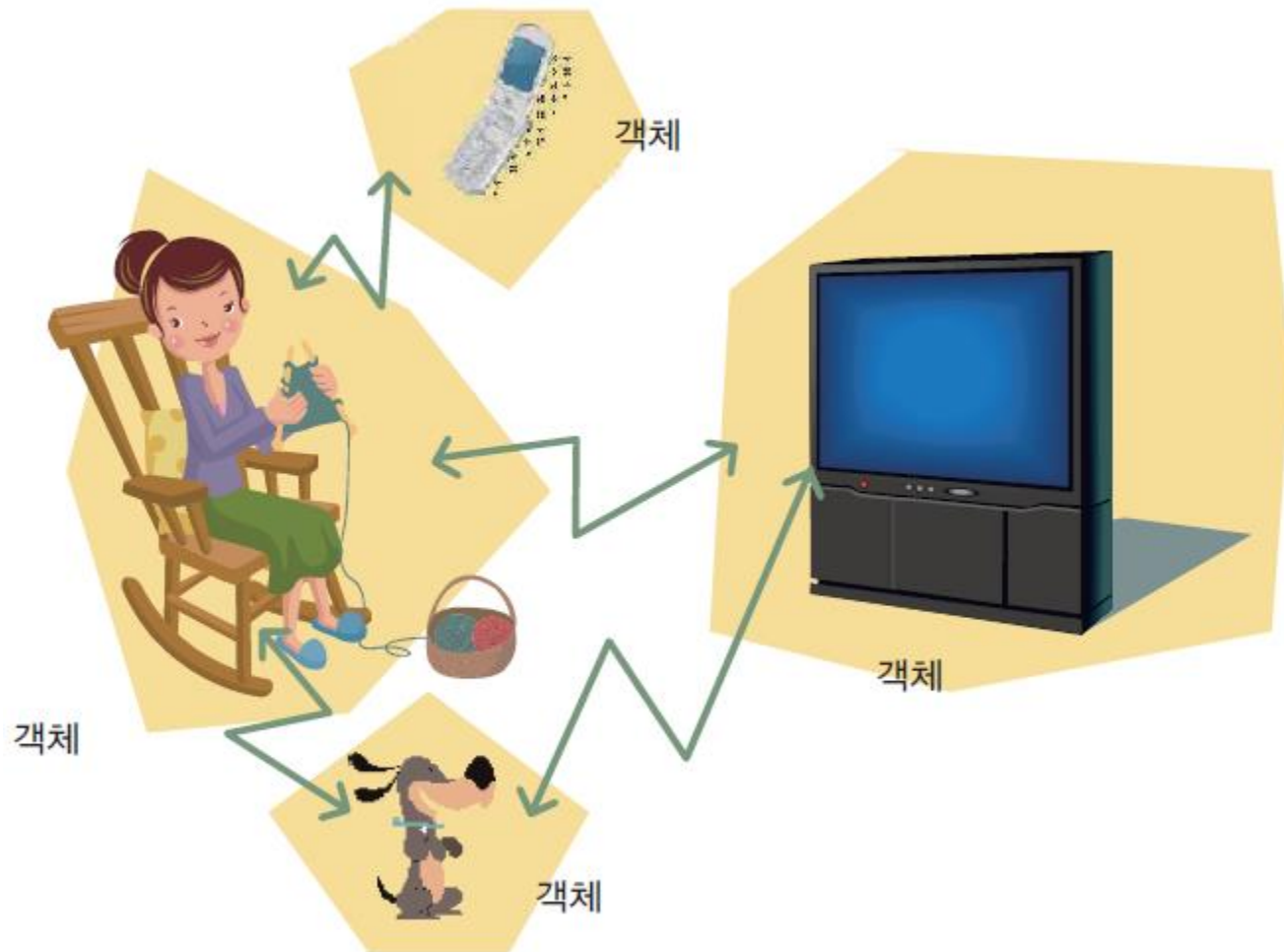


JAVA 프로그래밍

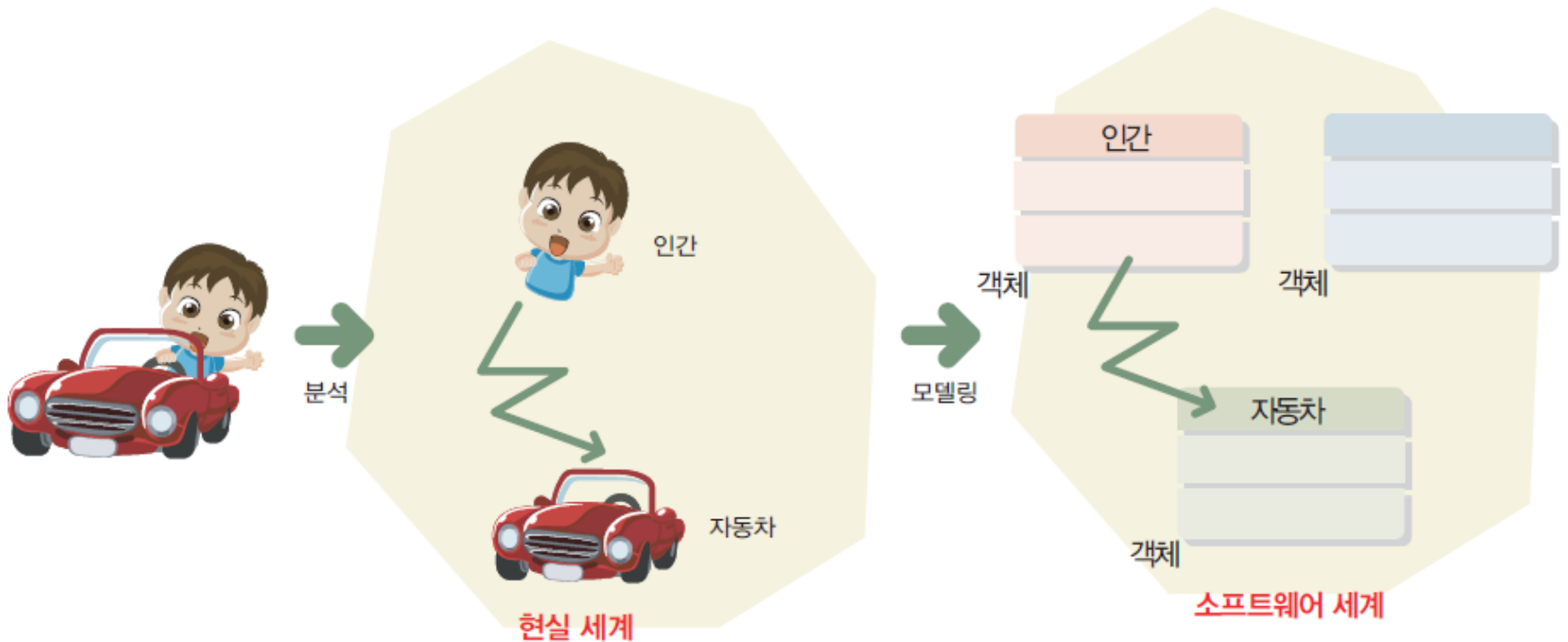
클래스와 객체 1

실제 세계는 객체로 이루어진다.



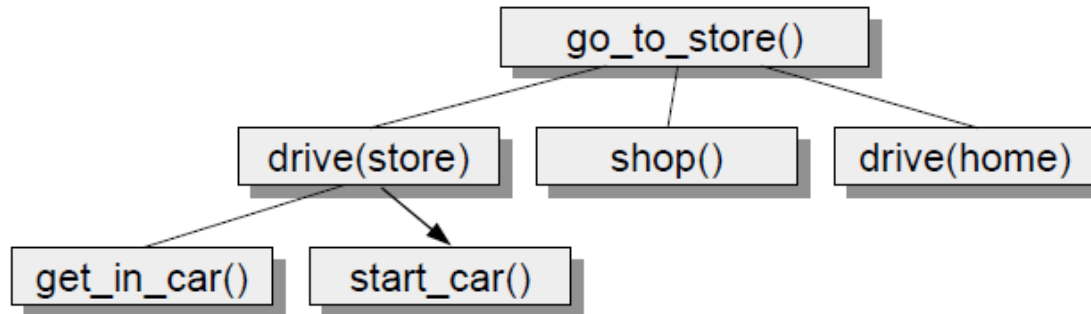
객체 지향이란?

- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법
- 현실 세계의 객체를 소프트웨어 객체로 설계하는 것을 객체 모델링(Object Modeling)이라고 한다.



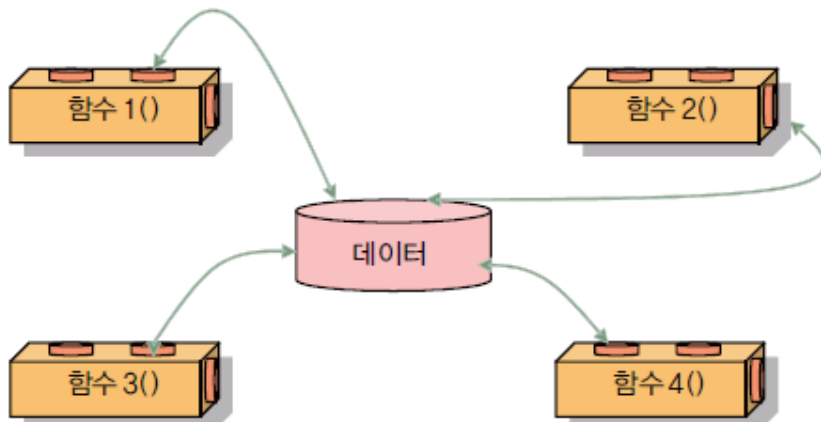
절차 지향과 객체 지향

- 절차 지향 프로그래밍(procedural programming)
 - 일어나는 일을 시간 순으로 프로그래밍 하는 방법

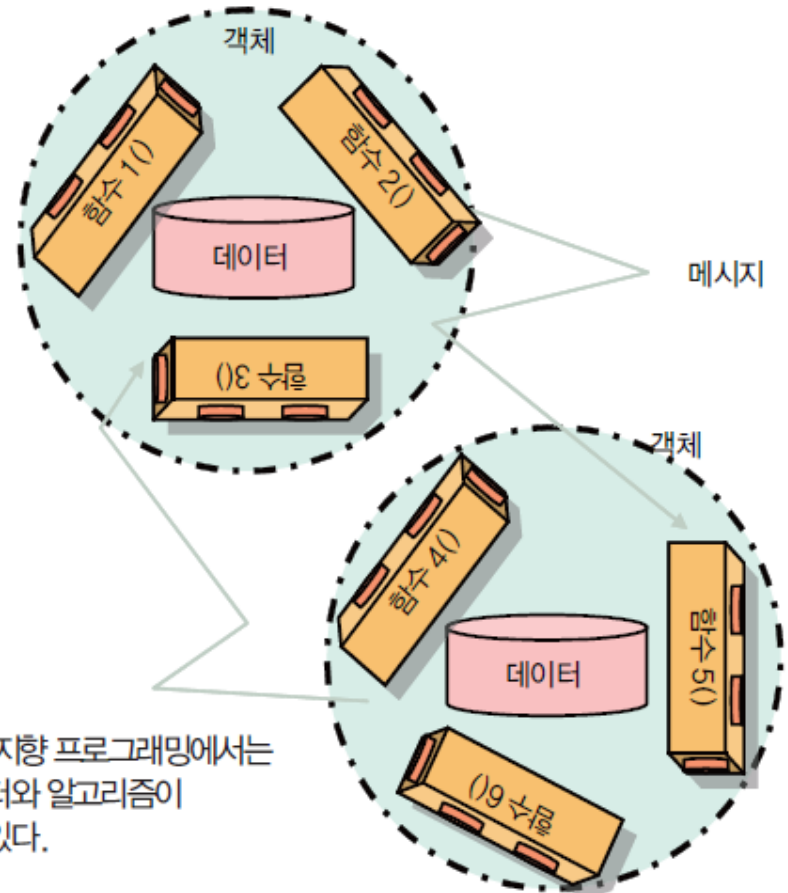


- 객체 지향 프로그래밍(Object-Oriented Programming)
 - 객체를 정의하고 객체 간 협력을 프로그래밍 하는 방법

절차 지향과 객체 지향



절차 지향 프로그래밍에서는
데이터와 알고리즘이
뭉쳐있지 않다.



객체 지향 프로그래밍에서는
데이터와 알고리즘이
뭉쳐있다.

- ✓ 객체들 사이의 상호작용 수단은 “메서드”이다.
- ✓ 객체가 다른 객체의 기능을 이용하는 것이 바로 ‘메서드 호출’이다.

생활 속의 객체 예

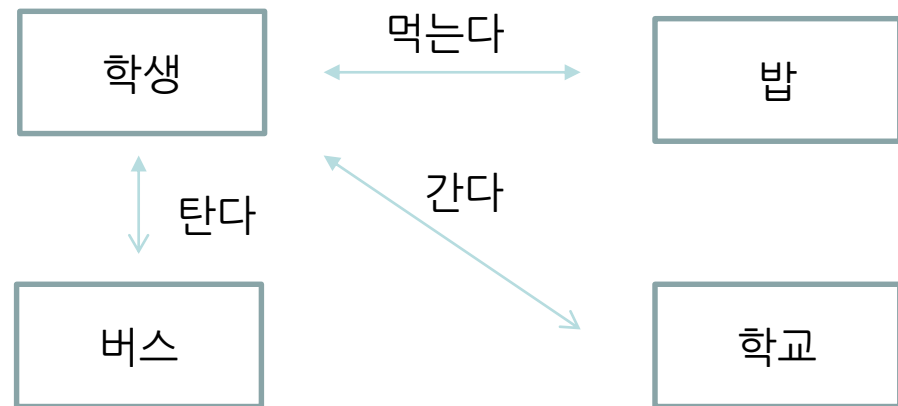
- 학교 가는 과정에 대한 절차적 프로그래밍

일어난다 → 씻는다 → 밥을 먹는다 → 버스를 탄다 → 요금을 지불한다
→ 학교에 도착한다.

- 시간의 흐름에 따른 프로그래밍

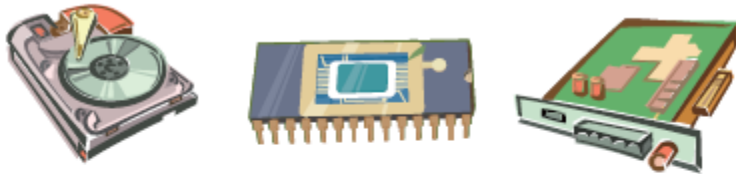
- 학교 가는 과정에 대한 객체 지향 프로그래밍

- 객체를 정의
- 객체의 기능 구현
- 객체 사이의 협력 구현

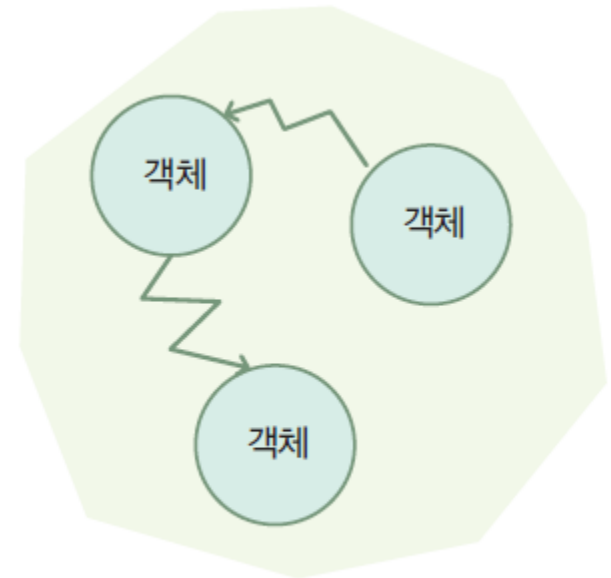


객체 지향 방법

- 객체 지향으로 소프트웨어를 작성하는 것은 컴퓨터 하드웨어 부품을 구입하여 컴퓨터를 조립하는 것과 비슷하다.



부품을 조립하여 제품을 만들듯이
객체들을 조립하여 소프트웨어를 만든다.



소프트웨어

객체 지향 언어의 특징

※ 기존의 프로그래밍 언어와 크게 다르지 않다.

- 기존의 프로그래밍 언어에 몇 가지 규칙을 추가한 것일 뿐이다.
- 3가지 중요 관점 : **재사용성, 유지보수, 중복의 제거**

※ 코드의 재사용성이 높다.

- 클래스를 정의하고 상속이라는 기능을 통해 기존에 작성된 코드를 이용하여 새로운 코드를 쉽게 작성할 수 있다.

※ 코드의 관리가 쉬워졌다.

- 기존의 절차적 언어에서는 프로그램의 일부를 변경할 때 변경된 코드와 관련된 부분도 일일이 수정해야 하지만, 객체지향 언어에서는 코드 간의 관계를 맺어줄 수 있기 때문에 코드가 변경되면 관련된 코드들도 자동적으로 변경되는 것과 같은 결과를 얻을 수 있어서 코드를 관리하는 것이 쉽다.

※ 신뢰성이 높은 프로그램의 개발을 가능하게 한다.

- 접근 제어자와 메서드를 통해 중요한 데이터가 외부에서 임의로 변경되는 것으로부터 보호하고, 데이터가 적절하지 않은 값이 되는 것을 막을 수 있다.

예를 들어, 비밀번호와 같이 중요한 데이터를 외부에서 접근하지 못하게 하거나 날짜의 월 데이터가 13월이 되는 것 등을 막을 수 있다.

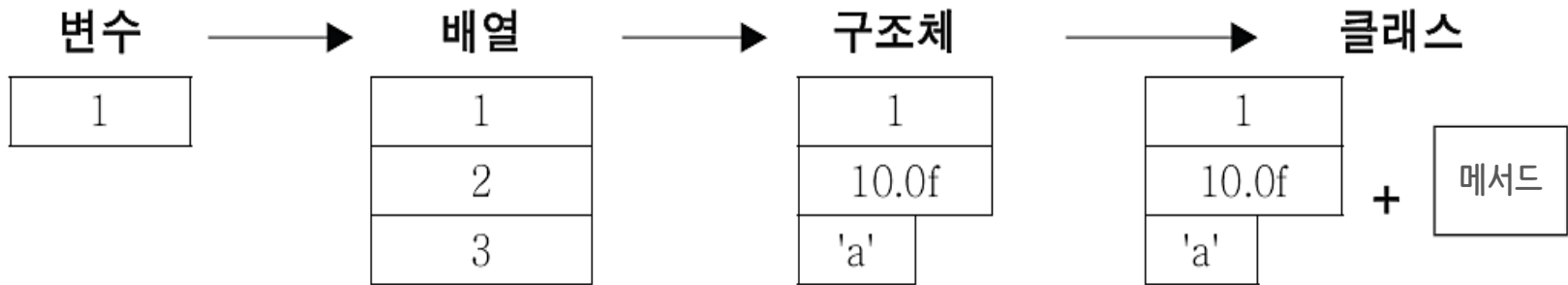
클래스(class)와 객체(object)

- **클래스** : 객체를 정의해 놓은 것으로 객체를 생성하는데 사용된다.
- **객체** : 실제로 존재하는 사물 또는 개념으로 객체의 속성과 기능에 따라 다르다.
- 클래스를 정의하고 클래스를 통해 객체를 생성하는 이유는 설계도를 통해서 제품을 만드는 이유와 같다. 설계도만 정확하게 잘 만들어 놓으면 제품을 만들 때마다 매번 고민할 필요 없이 설계도에서 정해진 규칙대로만 만들면 된다.
마찬가지로 클래스 하나만 잘 만들어 놓으면 많은 수의 객체를 만드는 것도 고민 없이 간단히 처리할 수 있다.
- 객체는 서로 독립적이다. 같은 클래스로부터 생성된 객체일지라도 서로 독립적이어서 어느 한 객체의 변화가 다른 객체에 영향을 주지 않는다.

클래스	객체
제품 설계도	제품
자동차 설계도	자동차
와플 기계(틀)	와플

클래스의 정의

- 클래스 : 데이터와 메서드의 결합
- 데이터 저장형태의 발전과정 : 변수는 하나의 값만 저장 → 같은 타입의 여러 데이터를 저장할 수 있는 배열 도입 → 타입은 다르지만 서로 관련된 데이터를 하나로 묶어 저장할 수 있는 구조체 도입 → 서로 관련된 데이터와 메서드를 결합한 클래스 도입



- ※ 변수 : 하나의 데이터를 저장할 수 있는 공간
- ※ 배열 : 같은 타입의 여러 데이터를 저장할 수 있는 공간
- ※ 구조체 : 타입에 관계없이 서로 관련된 데이터들을 저장할 수 있는 공간
- ※ 클래스 : 데이터와 메서드의 결합(구조체+메서드)

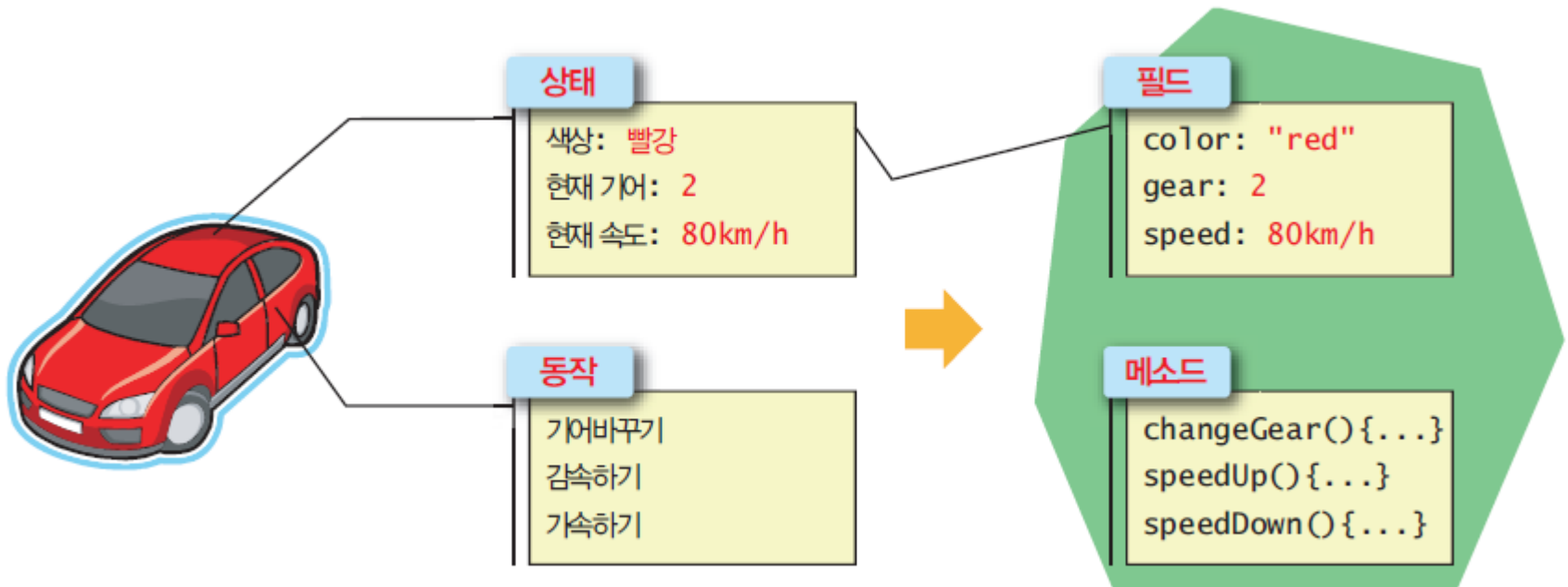
클래스(class)

- 클래스(class) : 객체를 만드는 설계도
- 클래스로부터 만들어진 각각의 객체를 해당 클래스의 **인스턴스(instance)**라고 한다.



객체(object)

- 객체(object)는 상태와 동작을 가진다.
 - 객체의 상태(state)는 객체의 특정 값(**속성**) → 멤버(인스턴스) 필드
 - 객체의 동작(behavior)은 객체가 취할 수 있는 **기능** → 멤버 메서드
- (외부로 부터 매개 값을 받을 수도 있고, 실행 후 값을 리턴 할 수도 있다. 객체 간의 데이터 전달 수단으로 사용)



객체의 구성 요소

- 객체의 구성 요소 : 속성(상태)과 기능(동작)으로 이루어져 있다.
- 객체는 속성과 기능의 집합이며, 속성과 기능을 객체의 **멤버**(member, 구성 요소)라고 한다.
- 클래스를 정의할 때 속성(state)은 변수로, 기능(behavior)은 메서드로 정의한다.

속성	크기, 길이, 높이, 색상, 볼륨, 채널 등
기능	켜기, 끄기, 볼륨 높이기, 볼륨 낮추기, 채널 높이기 등

필드

메서드

```
class TV {
```

```
String color;           // 색상  
boolean power;         // 전원상태(on/off)  
int channel;           // 채널
```

```
void power() { power = !power; }  
    // 전원 켜기, 끄기  
void channelUp { channel++; }  
    // 채널 높이기  
void channelDown { channel--; }  
    // 채널 낮추기
```

```
}
```

클래스의 구조

```
class 클래스이름 {  
    // 필드 정의 멤버 변수  
    int field1;  
    ...  
  
    // 메소드 정의  
    void method1(parameter) { ... }  
    ...  
}
```

클래스 이름

필드 정의:
객체의 속성을 나타낸다.

메소드 정의:
객체의 동작을 나타낸다.

클래스의 예 : 자동차

```
class Car {
```

```
String color; // 모델  
int speed;    // 현재 속도  
int gear;     // 현재 기어 단수
```

필드 정의:
객체의 속성을 나타낸다.

메소드 정의:
객체의 동작을 나타낸다.

```
void print() {  
    System.out.println("( " + color + ", " + speed + ", " + gear + " )");  
}
```

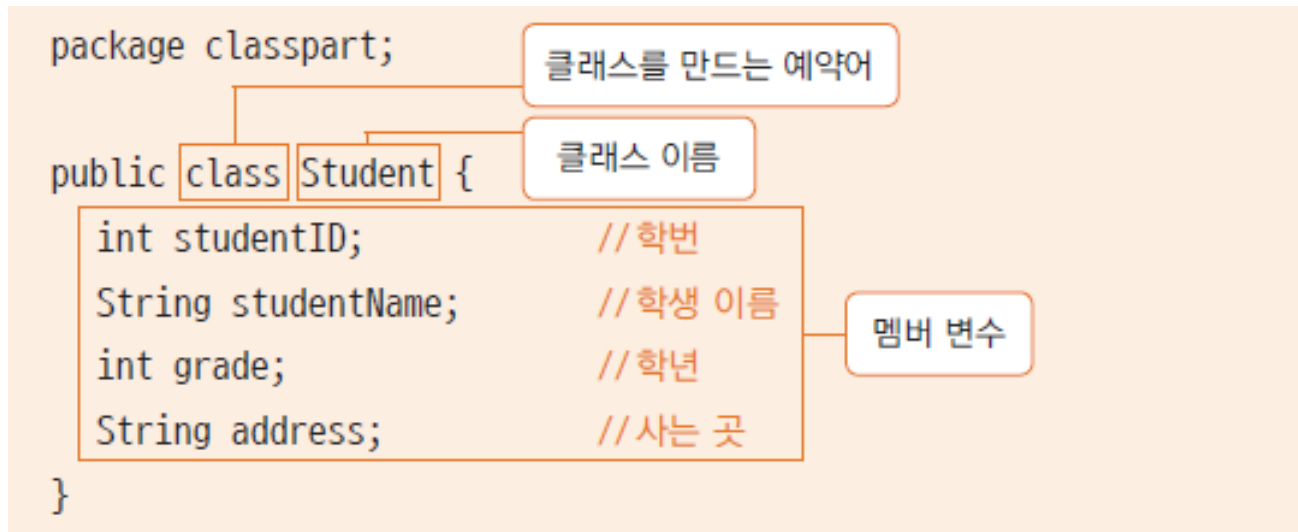
클래스 정의하기

- class는 대부분 대문자로 시작

하나의 java 파일에는 하나의 클래스를 두는 것이 원칙이나,
여러 개의 클래스가 있는 경우 public 클래스는 단 하나이며,
public 클래스와 java 파일의 이름은 동일해야 함.

- 자바의 모든 코드는 class 내부에 위치

학생 클래스 만들기



속성	자료형	변수 이름	설명
학번	int	studentID	학번은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
이름	String	studentName	학생 이름은 문자로 되어 있습니다. 그런데 이름은 A 같은 하나의 문자가 아니라 여러 개의 문자로 이루어진 문자열로 표현합니다. 문자열은 자바에서 제공하는 String 클래스를 사용합니다.
학년	int	grade	학년은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
사는 곳	String	address	문자열을 나타내기 위해 String을 사용합니다.

클래스의 속성

- 클래스의 특징을 나타냄
- property, attribute 라고도 함
- 자료형을 이용하여 멤버 변수로 선언

속성	자료형	변수 이름	설명
학번	int	studentID	학번은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
이름	String	studentName	학생 이름은 문자로 되어 있습니다. 그런데 이름은 A 같은 하나의 문자가 아니라 여러 개의 문자로 이루어진 문자열로 표현합니다. 문자열은 자바에서 제공하는 String 클래스를 사용합니다.
학년	int	grade	학년은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
사는 곳	String	address	문자열을 나타내기 위해 String을 사용합니다.

클래스의 기능

- 메서드(method)로 구현
- 멤버 함수(member function) 라고도 함
- 객체가 수행하는 기능을 구현

```
package classpart;
```

```
public class Student {
```

```
    int studentID;
```

```
    String studentName;
```

```
    int grade;
```

```
    String address;
```

메서드 추가

```
    public void showStudentInfo( ) {
```

```
        System.out.println(studentName + ", " + address); //이름, 주소 출력
```

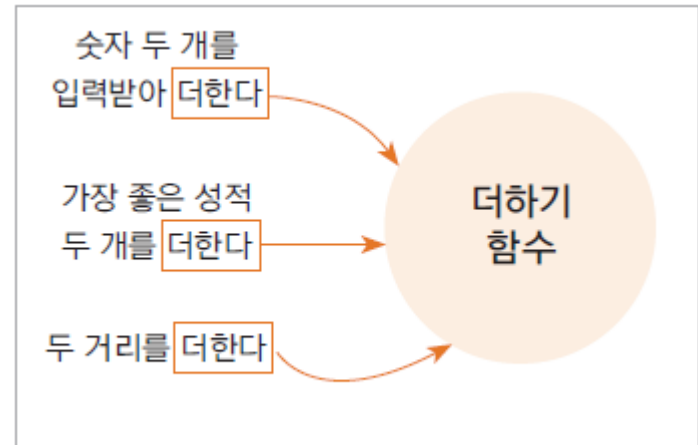
```
    }
```

```
}
```

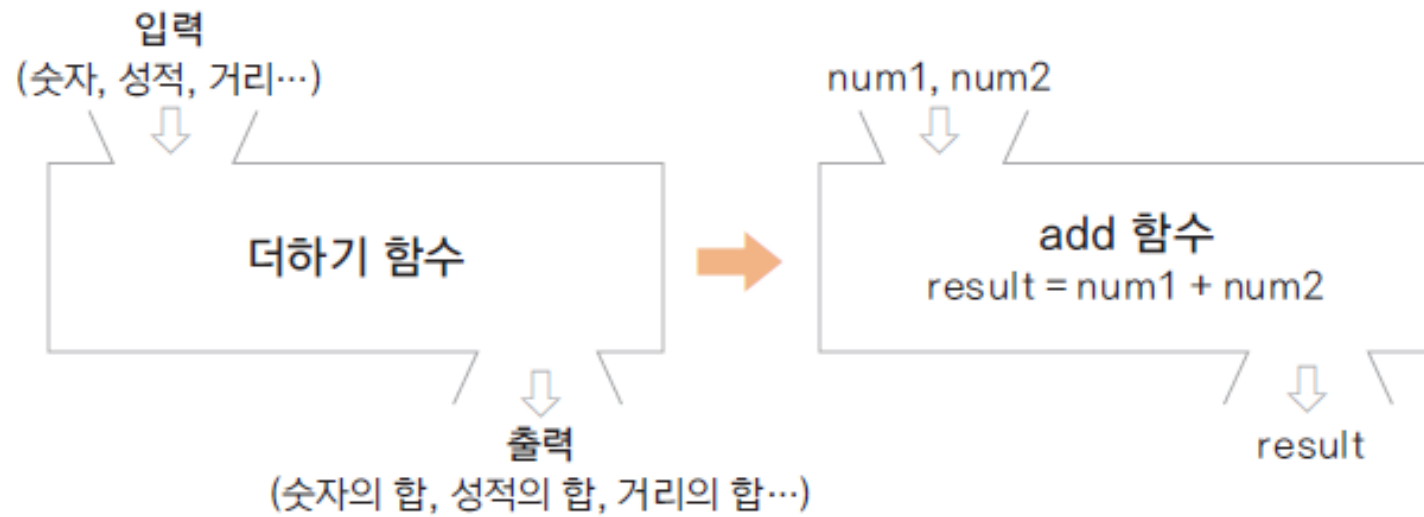
메서드

- 함수의 일종
- 객체의 기능을 제공하기 위해 클래스 내부에 구현되는 함수

- 함수란 ?
 - 하나의 기능을 수행하는 일련의 코드
 - 중복되는 기능은 함수로 구현하여 함수를 호출하여 사용함

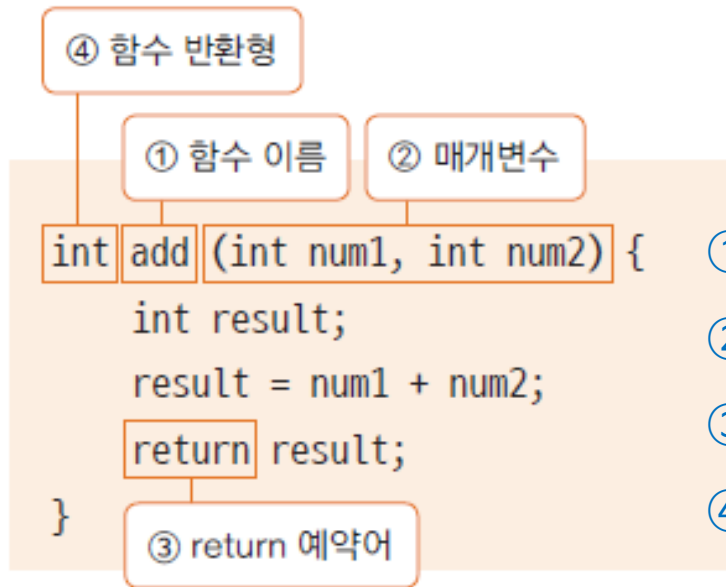


함수



함수 정의하기

- 함수를 코드로 구현
- 함수의 이름, 매개변수, 반환 값을 선언하고 코드를 구현함



- ① 함수 이름 : 함수의 기능과 관련하여 명명
- ② 매개 변수 : 함수의 수행을 위해 필요한 변수
- ③ return : 함수 수행 결과를 반환하기 위한 예약어
- ④ 함수 반환형 : 반환 값의 자료형을 나타냄

반환 값이 없는 경우 void라고 씀

함수 구현하고 호출하기

```
package classpart;

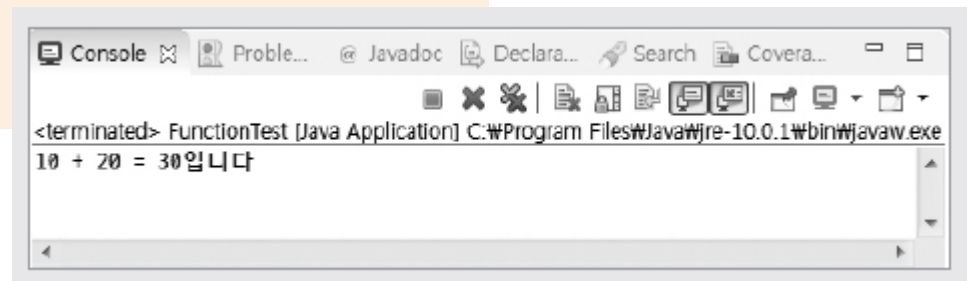
public class FunctionTest {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;

        int sum = add(num1, num2);
        System.out.println(num1 + " + " + num2 + " = " + sum + "입니다");
    }

    public static int add(int n1, int n2) {
        int result = n1 + n2;
        return result; // 결과 값 반환
    }
}
```

add() 함수 호출

add() 함수



Quiz

<FunctionTest.java>

사칙 연산을 수행하는 함수를 모두 구현하고 결과 값을 출력해 보세요.

(Hint)

addNum(덧셈), subtract(뺄셈), times(곱셈), divide(나눗셈) 메서드 이름으로 사용

[실행 결과]

$$10 + 20 = 30$$

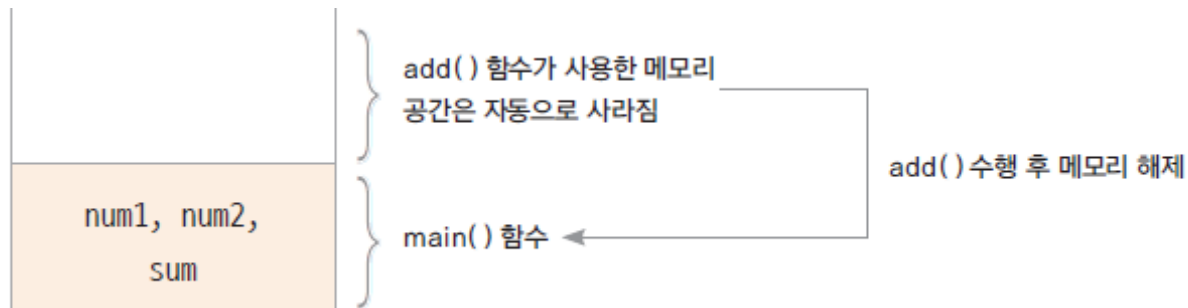
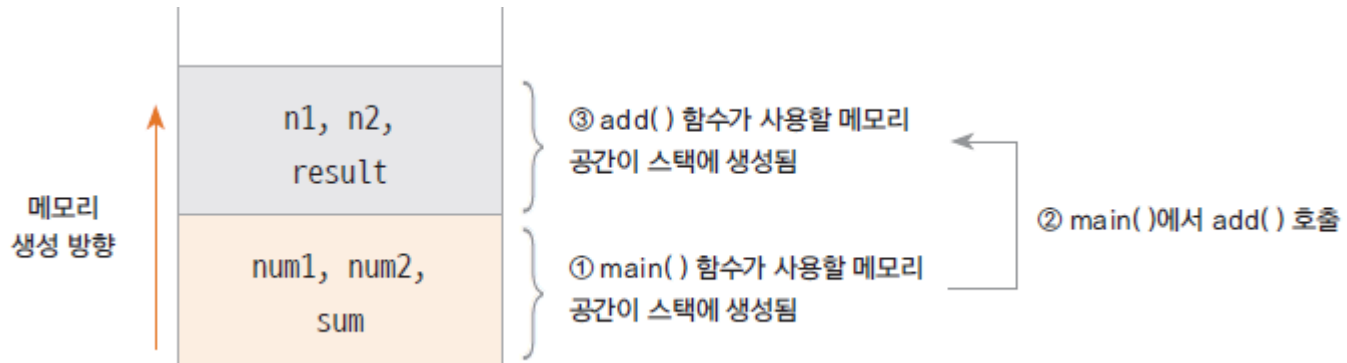
$$10 - 20 = -10$$

$$10 * 20 = 200$$

$$10 / 20 = 0.5$$

함수와 스택 메모리

- 함수가 호출될 때 사용하는 메모리 → 스택(stack)
- 함수의 기능 수행이 끝나면 자동으로 반환되는 메모리
- 함수 호출과 스택 메모리 구조



클래스에 메서드 구현하기

- 클래스의 메서드는 멤버 변수를 사용하여 기능 구현(멤버 함수)
- 학생의 이름을 반환하는 메서드

```
package classpart;
```

```
public class Student {
```

```
    int studentID;
```

```
    String studentName;
```

```
    int grade;
```

```
    String address;
```

```
    public String getStudentName( ) {
```

```
        return studentName;
```

```
    }
```

```
}
```

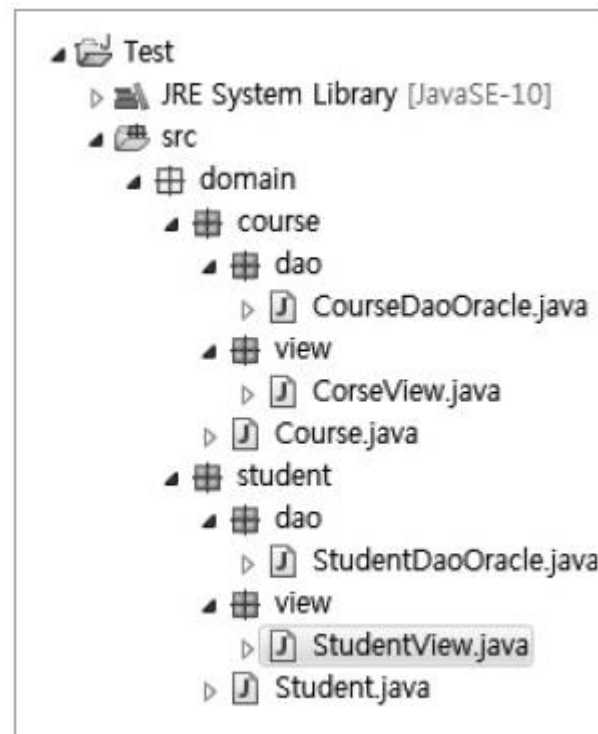
학생의 이름을 반환하는
메서드

❖ 메서드 이름은 해당 클래스를 사용하는 코드의 입장에서 짓는 것이 좋음.

패키지

- 소스의 묶음 카테고리
- 계층구조로 구성되며 소문자로 명명
- 패키지를 정의 하는 것은 소스 코드를 어떻게 구분하여 관리할 것인지를 정의 하는 것

기본 클래스와
각 기능을 하는 클래스들을
패키지로 구분하여 관리 하는 예



naming convention

- 클래스 이름은 대문자로 시작
- 패키지의 이름은 소문자로 작성
- 메서드나 변수의 이름은 소문자로 시작하고 단어마다 대문자로 써 줌으로써 가독성을 높임(camel notation : 낙타 표기법)
- 반드시 지켜야 하는 사항은 아니지만, 일종의 관례

객체를 생성하는 방법

클래스형 변수이름 = **new** 생성자();

Car myCar = **new** Car();



객체 생성 코드

```
class Car {  
    // 필드 정의  
    String color;    // 색상  
    int speed;       // 현재 속도  
    int gear;        // 현재 기어  
    void print() {  
        System.out.println("( " + color + ", " + speed + ", " + gear + " )");  
    }  
}  
  
class CarTest {  
    public static void main(String[] args) {  
        Car myCar = new Car();  
        myCar.color = "red";  
        myCar.speed = 0;  
        myCar.gear = 1;  
        myCar.print();  
    }  
}
```

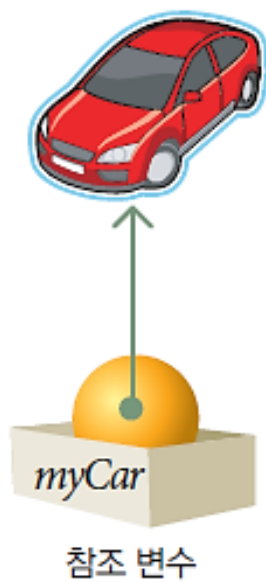


클래스

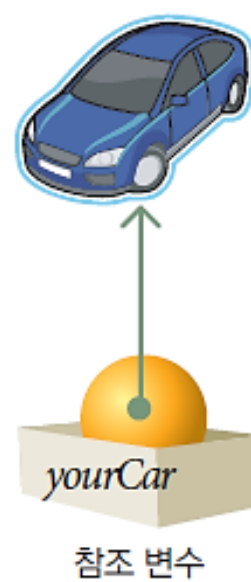


객체

객체를 하나 더 생성하는 코드



color	<input type="text" value="red"/>
speed	<input type="text" value="0"/>
gear	<input type="text" value="1"/>
print()	<input type="text" value=""/>



color	<input type="text" value="blue"/>
speed	<input type="text" value="60"/>
gear	<input type="text" value="3"/>
print()	<input type="text" value=""/>



클래스

```
01 class Car {  
02     // 필드 정의  
03     String color;    // 색상  
04     int speed;       // 현재 속도  
05     int gear;        // 현재 기어  
06     void print() {  
07         System.out.println("( " + color + ", " + speed + ", " + gear + " )");  
08     }  
09 }
```



객체

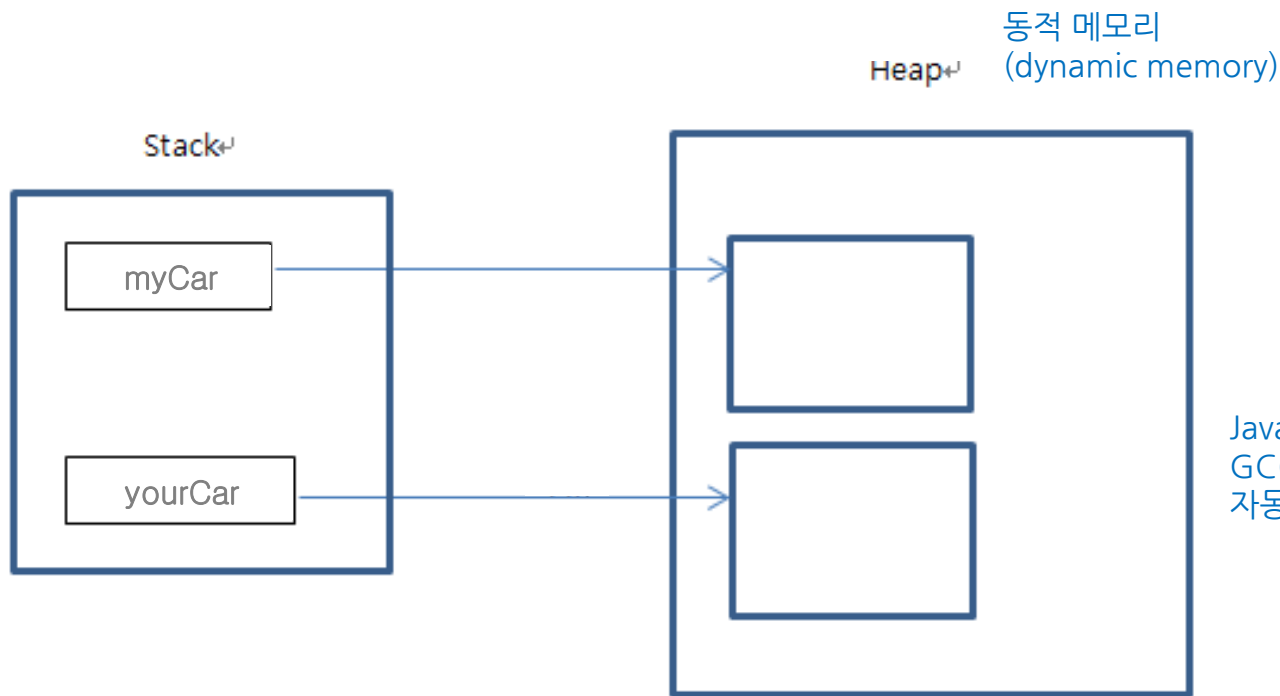
```
10 class CarTest {  
11     public static void main(String[] args) {  
12  
13         Car myCar = new Car();    // 객체 생성  
14         myCar.color = "red";      // 객체의 필드 변경  
15         myCar.speed = 0;          // 객체의 필드 변경  
16         myCar.gear = 1;           // 객체의 필드 변경  
17         myCar.print();            // 객체의 메소드 호출  
18  
19         Car yourCar = new Car();  // 객체 생성  
20         yourCar.color = "blue";    // 객체의 필드 변경  
21         yourCar.speed = 60;        // 객체의 필드 변경  
22         yourCar.gear = 3;          // 객체의 필드 변경  
23         yourCar.print();           // 객체의 메소드 호출  
24     }  
25 }
```



객체

인스턴스와 힙(heap) 메모리

- 하나의 클래스 코드로부터 여러 개의 인스턴스를 생성
- 인스턴스는 힙(Heap) 메모리에 생성됨
- 각각의 인스턴스는 다른 메모리에 다른 값을 가짐

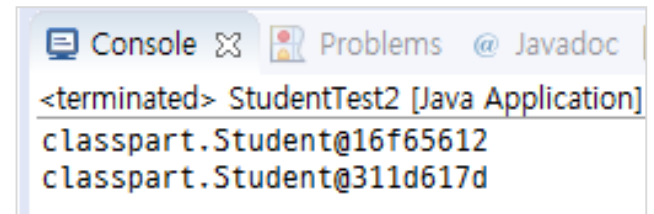


Java에서는 사용이 끝나면
GC(Garbage Collector)가
자동으로 메모리를 해제해 줌.

참조 변수와 참조 값

- 참조 변수 : 인스턴스 생성시 선언하는 변수
- 참조 값 : 인스턴스가 생성되는 힙(heap) 메모리 주소
- 참조 변수를 이용한 참조 값 확인 코드

```
public class StudentTest2 {  
  
    public static void main(String[] args) {  
  
        Student student1 = new Student();  
        student1.studentName = "홍길동";  
  
        Student student2 = new Student();  
        student2.studentName = "이순신";  
  
        System.out.println(student1);  
        System.out.println(student2);  
    }  
}
```



Console Problems Javadoc

<terminated> StudentTest2 [Java Application]
classpart.Student@16f65612
classpart.Student@311d617d

클래스, 인스턴스, 참조 변수, 참조 값

용어	설명
객체	객체 지향 프로그램의 대상, 생성된 인스턴스
클래스	객체를 프로그래밍하기 위해 코드로 만든 상태
인스턴스	클래스가 메모리에 생성된 상태
멤버 변수	클래스의 속성, 특성
메서드	멤버 변수를 이용하여 클래스의 기능을 구현
참조 변수	메모리에 생성된 인스턴스를 가리키는 변수
참조 값	생성된 인스턴스의 메모리 주소 값

Quiz

<BoxTest.java>

[클래스 이름] Box

[클래스의 필드]

width(너비), height(높이), length(길이)

[멤버 메서드] void print()

상자의 크기 계산식 = 너비*높이*길이

[실행 결과]

상자 너비 : 20cm, 상자 높이 : 30cm, 상자 길이 : 10cm

상자의 크기 : 6000cm³

Quiz

<StudentTest.java>

[클래스 이름] Student

[클래스의 필드]

no(학번), name(이름), kor(국어), math(수학), eng(영어), sum(총점), avg(평균)

[멤버 메서드]

```
void print() {  
    총점 계산식;  
    평균 계산식;  
    ...  
}
```

[실행 결과]

학생의 총점 : 278

학생의 평균 : 92.7

Quiz

<MovieTest.java>

[클래스 이름] Movie

[클래스의 필드]

title : 영화 제목

name : 감독 이름

year : 발표된 연도

score : 평가점수

[멤버 메서드] void print()

[실행 결과]

오락성 점수 입력(1~100) : 85

작품성 점수 입력(1~100) : 93

예술성 점수 입력(1~100) : 79

영화 제목: 허삼관, 감독 이름: 하정우, 발표 연도: 2015, 영화 평점: 85.7

생성자(constructor)

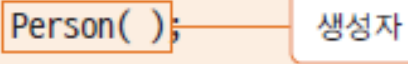
- 인스턴스 생성 시, new 키워드와 함께 사용했던 생성자

```
package constructor;

public class Person {
    String name;
    float height;
    float weight;
}

package constructor;

public class PersonTest {
    public static void main(String[ ] args) {
        Person personLee = new Person( );
    }
}
```

A diagram consisting of a rectangular box with a red border containing the text 'Person()'. A horizontal line extends from the right side of this box to another rectangular box with a red border containing the text '생성자' (constructor).

생성자(constructor)

- 생성자 기본 문법

```
<modifiers> <class_name>([<argument_list>]) {  
    [<statements>]  
}
```

- 생성자는 인스턴스를 초기화 할 때의 명령어 집합
- 생성자의 이름은 그 클래스의 이름과 같음
- 생성자는 메서드가 아님. 상속되지 않으며, 리턴 값은 없음

디폴트 생성자 (default constructor)

- 하나의 클래스에는 반드시 적어도 하나 이상의 Constructor가 존재
- 프로그래머가 Constructor를 기술하지 않으면,
Default Constructor가 자동으로 생김 (컴파일러가 자동으로 코드에 넣어 줌)
 - Default Constructor는 매개변수가 없음
 - Default Constructor는 구현부가 없음
- 만약 클래스에 매개변수가 있는 생성자를 추가하면,
Default Constructor는 자동으로 제공되지 않음

생성자 오버로드 (constructor overload)

- 필요에 의해 생성자를 추가 하는 경우, 여러 개의 생성자가 하나의 클래스에 있음
(**overload**) → 매개변수가 다른 생성자를 여러 개 만들 수 있음

```
package constructor;
```

```
public class Person {
```

```
    String name;
```

```
    float height;
```

```
    float weight;
```

```
    public Person( ) { }
```

디폴트 생성자

```
    public Person(String pname) {
```

```
        name = pname;
```

```
    }
```

이름을 매개변수로 입력받는 생성자

```
    public Person(String pname, float pheight, float pweight) {
```

```
        name = pname;
```

```
        height = pheight;
```

```
        weight = pweight;
```

```
    }
```

이름, 키, 몸무게를
매개변수로 입력
받는 생성자

```
}
```

Quiz

<PersonTest.java>

Person 클래스에 이미 추가된 생성자 외에 이름, 키, 몸무게를 매개변수로 받는 생성자를 추가해 보세요.

그리고 추가된 생성자를 사용해 인스턴스를 생성해 보세요.

[실행 결과]

이순신님의 키는 175.3cm이고, 몸무게는 75.4kg입니다.

참조 자료형 (reference data type)

- 변수의 자료형



- 클래스 형으로 선언하는 자료형

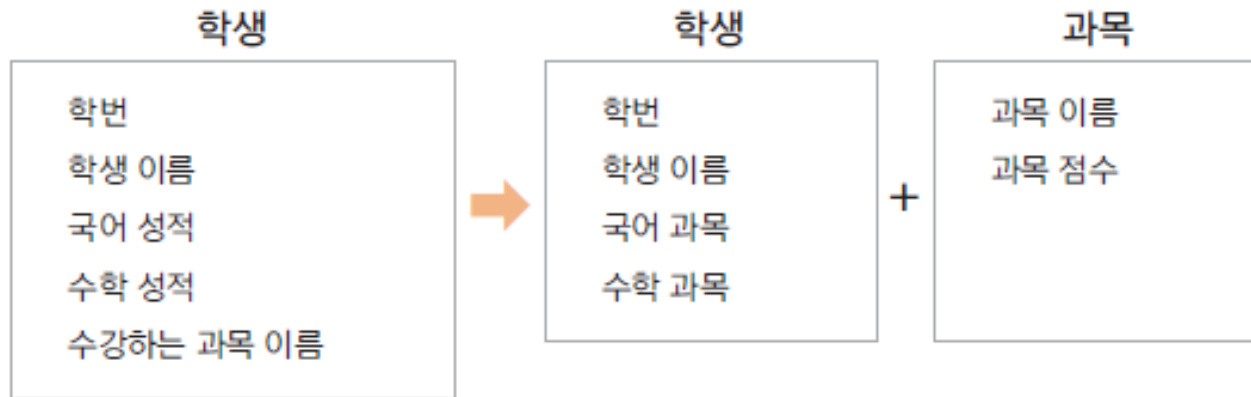
참조 자료형의 예

학생의 속성 중 과목에 대한 부분

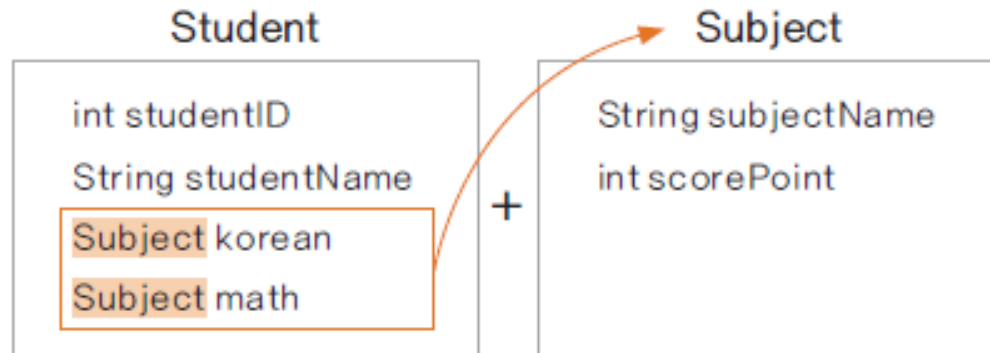
과목에 대한 각 속성을 학생 클래스에 정의하지 않고

과목이라는 클래스로 분리해서 사용

이 때 과목은 참조 자료형으로 선언



참조 자료형의 예



```
public class Student3 {
    int studentID;
    String studentName;
    Subject korean;
    Subject math;
}
```

Subject형을 사용하여 선언

```
public class Subject {
    String SubjectName;
    int scorePoint;
}
```

Quiz

아래와 같이 클래스를 만들고 결과를 출력해 보세요.

나이가 40살, 이름이 James라는 남자가 있습니다.
이 남자는 결혼을 했고, 자식이 셋 있습니다.

(Hint)

클래스 이름은 보편적인 것으로 만드세요.(Man 또는 Person)

클래스에서 사용할 멤버 변수와 각 멤버 변수에 맞는 자료형을 선언해서 사용하세요.

[출력 결과]

나이 : 40

이름 : James

결혼 여부 : true

자녀 수 : 3

Quiz

쇼핑몰에 주문이 들어온 주문 내용은 다음과 같습니다.

주문 번호 : 201803120001

주문자 아이디 : abc123

주문 날짜 : 2018년 3월12일

주문자 이름 : 홍길순

주문 상품 번호 : PD0345-12

배송 주소 : 서울시 영등포구 여의도동 20번지

위 주문에 대한 클래스를 만들고 주문 내용을 인스턴스로 생성한 후,
위와 같은 형식으로 주문 내용을 출력해 보세요.