객체와 클래스
생성자와 멤버변수

강사 : 강병준

객체지향언어의 역사

- 과학, 군사적 모의실험(simulation)을 위해 컴퓨터를 이용한 가상세계를 구현하려는 노력으로부터 객체지향이론이 시작됨

- 1960년대 최초의 객체지향언어 Simula탄생

- 1980년대 절차방식의 프로그래밍의 한계를 객체지향방식으로 극복하려고 노력함.(C++, Smalltalk과 같은 보다 발전된 객체지향언어가 탄생)

- 1995년 말 Java탄생. 객체지향언어가 프로그래밍 언어의 주류가 됨.

객체지향과 절차지향

객체 지향(Object-Oriented) 대 절차지향 (Procedural-Oriented)

- 절차지향(구조적 프로그래밍) : 데이터 구조와 그 데이터를 변화 시키는 알고리즘으로 구성

- 객체지향 : 객체들이 메시지(message)를 통하여 통신함으로써 원하는 결과를 얻는다. 각 객체는 고유의 데이터와 데이터를 처리할 수 있는 메소드로 구성

객체(Object)

▶객체란?

정수,실수 또는 문자 등과 같이 단순한 테이터나 자동차, 비행기 등과 같은 복잡한 사물뿐만 아니라 기업에의 공헌도, 이성간의 사랑 등의 추상적 개념에 이르기까지, 인간이 하나의 개념으로 파악하는 모든 것들을 일컫는 것으로 일반적으로 인간에게 인지되는 개념적인 한 단위를 의미한다.

▶프로그래밍에서의 객체란?

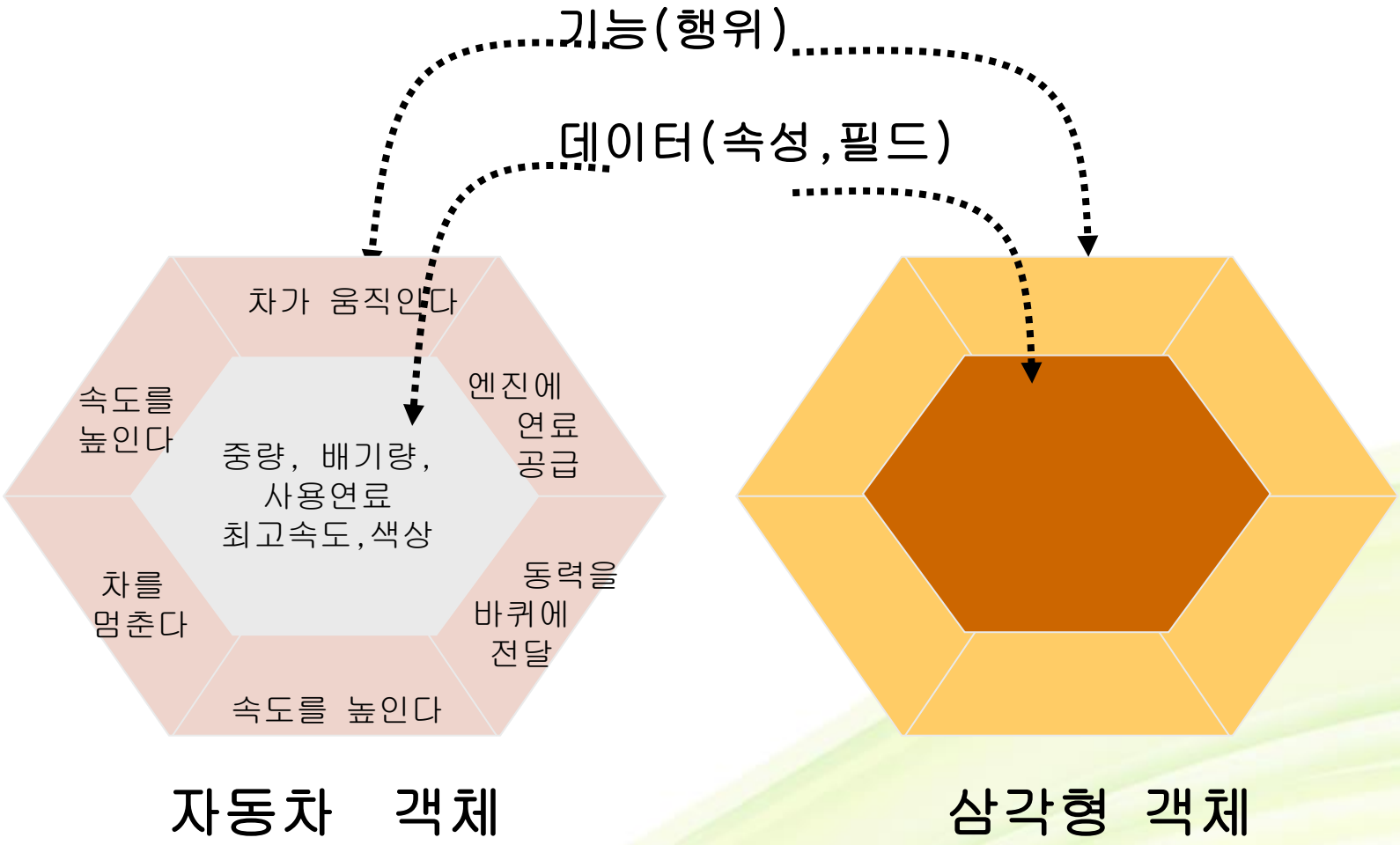
상태와 행위의 집합체이다…즉 속성(필드,데이터) 와

메소드(함수,행위)의 집합체이다.

속성(필드) => 객체의 정적특성으로 객체가 가지고 있는 정보를 보관하는 기억장소로 사용되는 데이터 영역

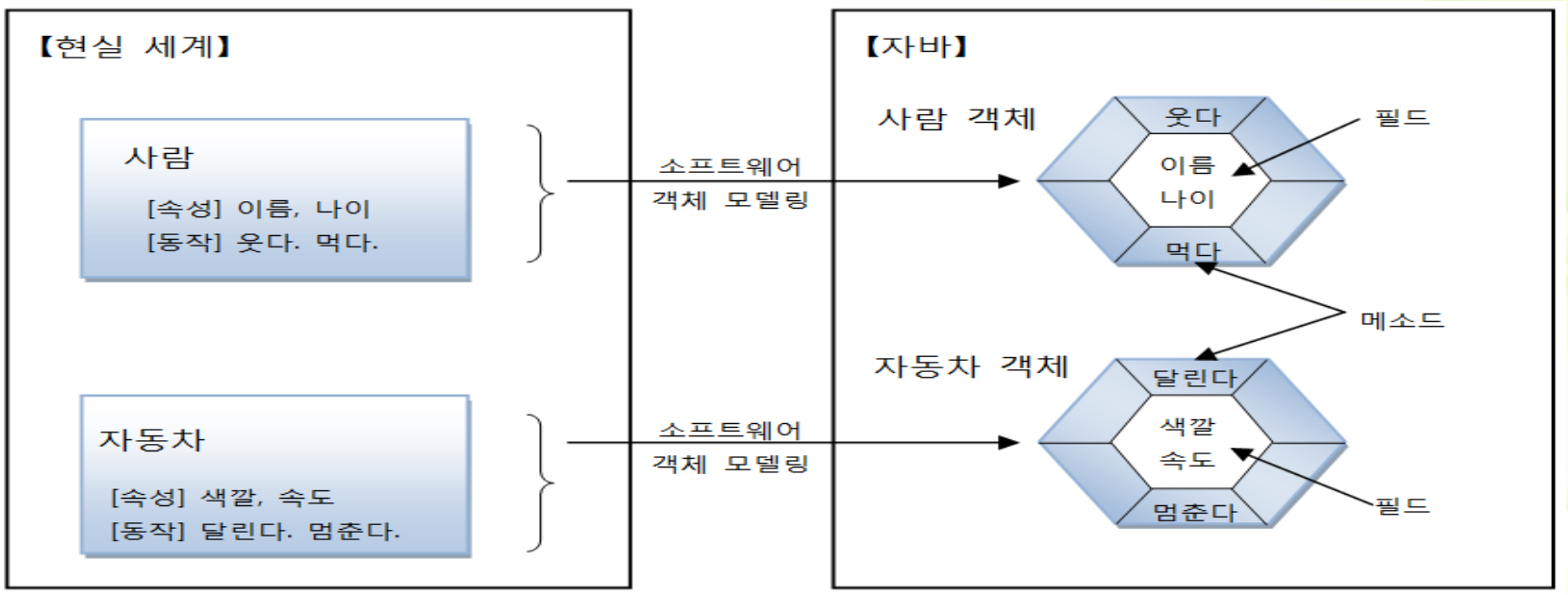
메소드 => 객체의 동적특성으로서 객체의 데이터를 엑세스하거나 또는 객체가 가지고 있는 속성을 변경하는 일을 한다.

객체의 예


기능(행위)
데이터(속성,필드)
차가 움직인다
엔진에 연료 공급
동력을 바퀴에 전달
속도를 높인다
차를 멈춘다
속도를 높인다
중량, 배기량, 사용연료 최고속도,색상
자동차 객체
삼각형 객체

객체 지향 프로그래밍

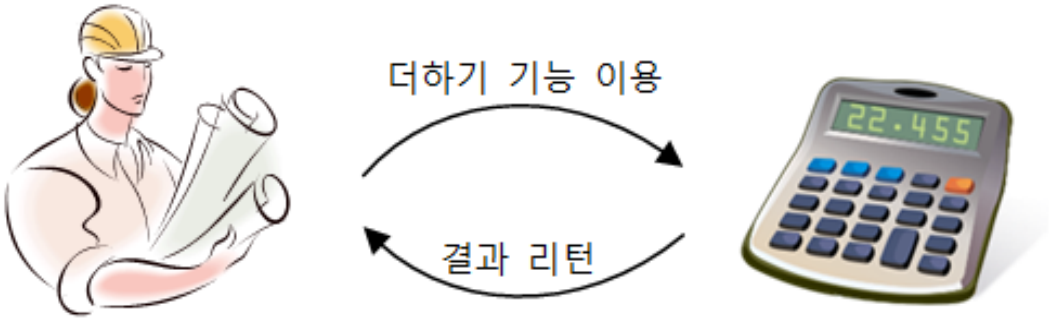
- 객체 지향 프로그래밍
  - OOP: Object Oriented Programming
  - 부품 객체를 먼저 만들고 이것들을 하나씩 조립해 완성된 프로그램을 만드는 기법
- 객체(Object)란?
  - 물리적으로 존재하는 것 (자동차, 책, 사람)
  - 추상적인 것(회사, 날짜) 중에서 자신의 속성과 동작을 가지는 모든 것
  - 객체는 필드(속성) 과 메소드(동작)로 구성된 자바 객체로 모델링 가능

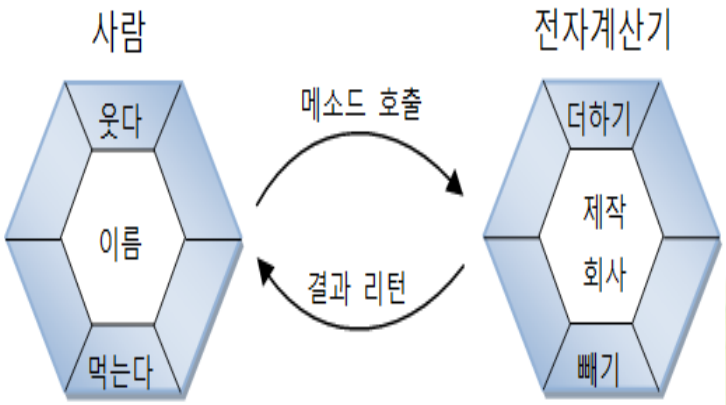


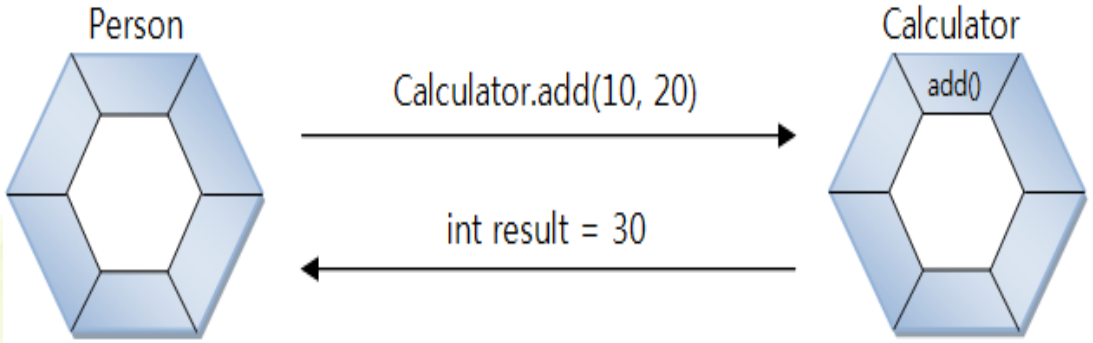
객체 지향 프로그래밍

❖ 객체의 상호 작용

- 객체들은 서로 간에 기능(동작)을 이용하고 데이터를 주고 받음



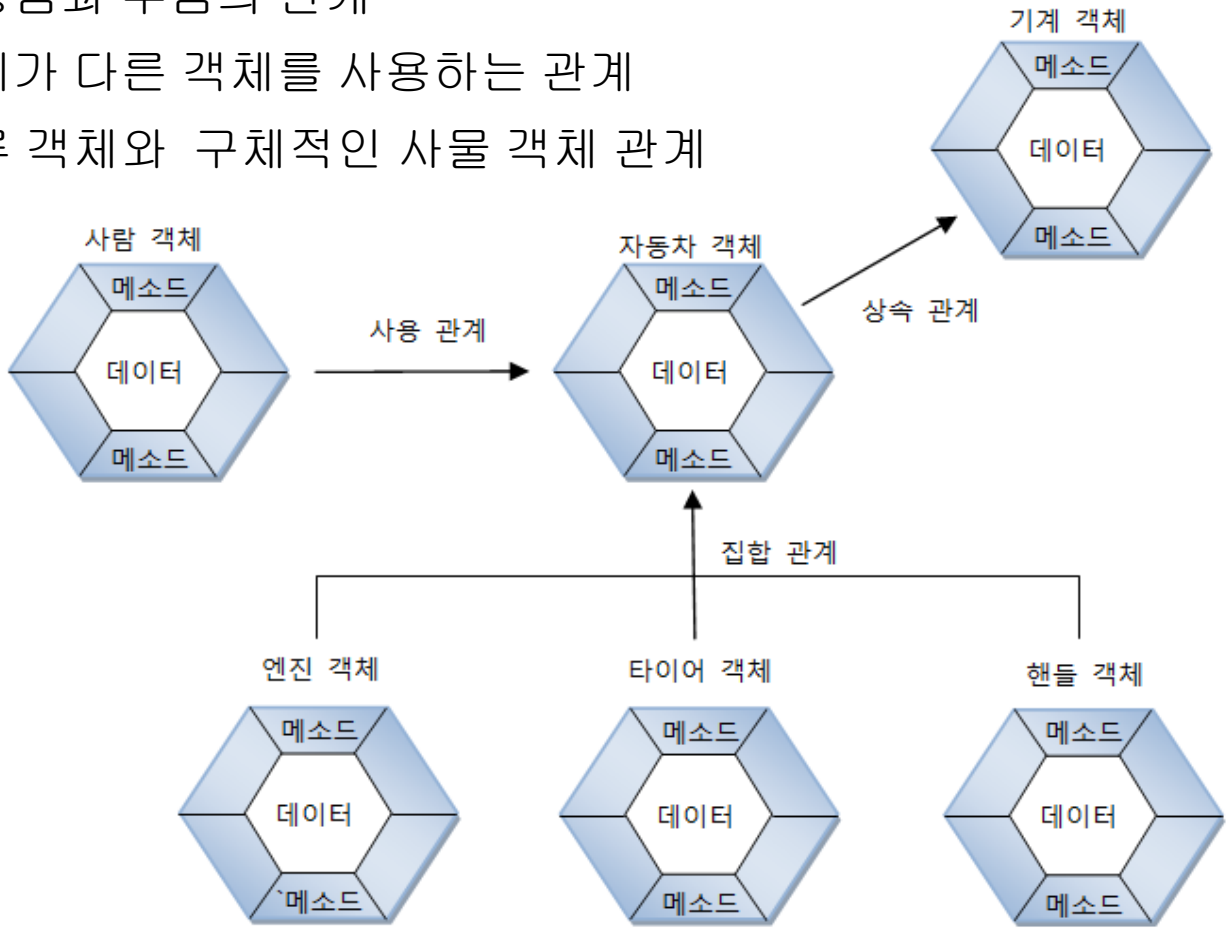




객체 지향 프로그래밍

객체간의 관계

- 객체 지향 프로그램에서는 객체는 다른 객체와 관계를 맺음
- 관계의 종류
  - 집합 관계: 완성품과 부품의 관계
  - 사용 관계: 객체가 다른 객체를 사용하는 관계
  - 상속 관계: 종류 객체와 구체적인 사물 객체 관계



클래스(Class)

▶클래스란?

서로 공통되는 구조를 가지고 있는 객체를 모아 이 객체들이 가지고 있는 데이터 영역의 구조와 각각의 객체가 수행할수 있는 메소드들을 정의한 객체를 의미한다. 한마디로 일반화된 속성과 메소드로 객체를 기술한 것을 클래스라고 한다..

객체는 항상 클래스로부터 생성된다. 즉 클래스는 객체를 생성하는 형판(template)

클래스는 두개의 구성요소(member)인 자료구조(필드)와 연산(메소드)을 가진다

클래스로부터 생성된 객체를 instance라 한다.

객체 = instance

정보처리의 주체는 클래스가 아니라 객체이다

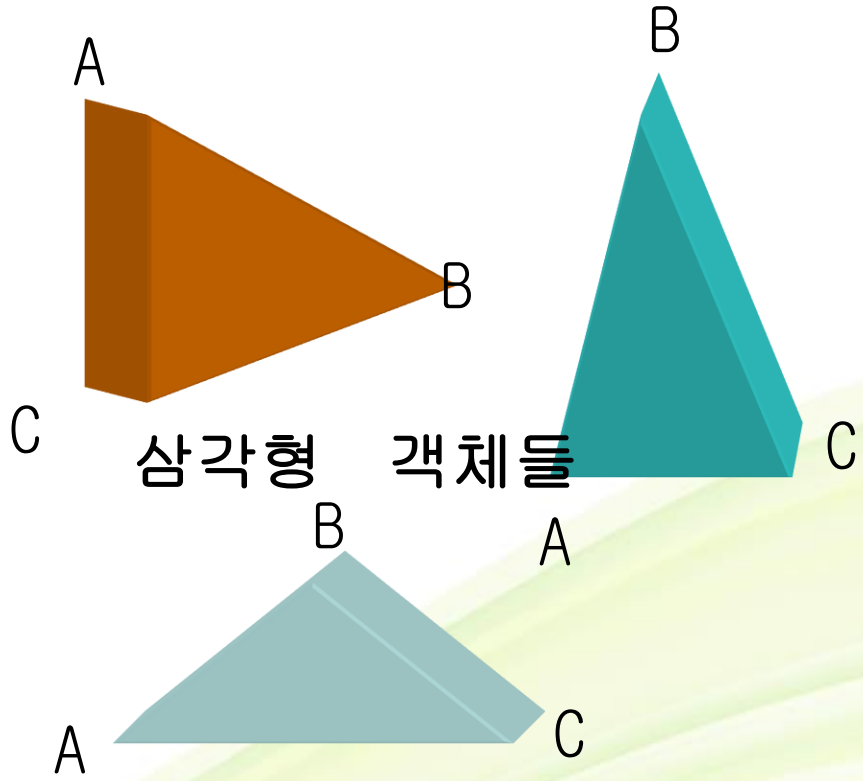
객체지향 프로그래밍의 시작은 클래스의 생성이다

클래스의 예

삼각형 클래스와 객체

▶속성
- 점 A
- 점 B
- 점C
- 색깔
- 패턴

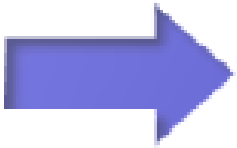
▶메소드
- 그리기
- 색바꾸기
- 옮기기



삼각형 객체들

클래스와 객체

클래스(설계도)



객체(건물)

클래스와 객체의 정의와 용도

- 클래스의 정의 – 클래스란 객체를 정의해 놓은 것이다.
- 클래스의 용도 – 클래스는 객체를 생성하는데 사용된다.

- 객체의 정의 – 실제로 존재하는 것. 사물 또는 개념.
- 객체의 용도 – 객체의 속성과 기능에 따라 다름.

클래스	객체
제품 설계도	제품
TV설계도	TV
붕어빵기계(틀의미)	붕어빵

객체와 인스턴스

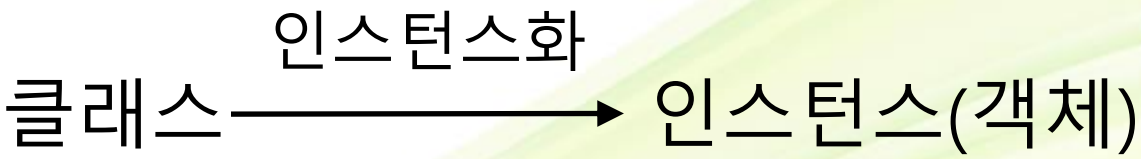
▶ 객체 ≒ 인스턴스

- 객체(object)는 인스턴스(instance)를 포함하는 일반적인 의미

책상은 인스턴스다. 책상은 객체다.	책상은 책상 클래스의 객체다. 책상은 책상 클래스의 인스턴스다.

▶ 인스턴스화(instantiate, 인스턴스化)

- 클래스로부터 인스턴스를 생성하는 것.



객체의 구성요소 – 속성과 기능

▶ 객체는 속성과 기능으로 이루어져 있다.

- 객체는 속성과 기능의 집합이며, 속성과 기능을 객체의 멤버(member, 구성요소)라고 한다.

▶ 속성은 변수로, 기능은 메서드로 정의한다.

- 클래스를 정의할 때 객체의 속성은 변수로, 기능은 메서드로 정의한다.

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

변수

메서드

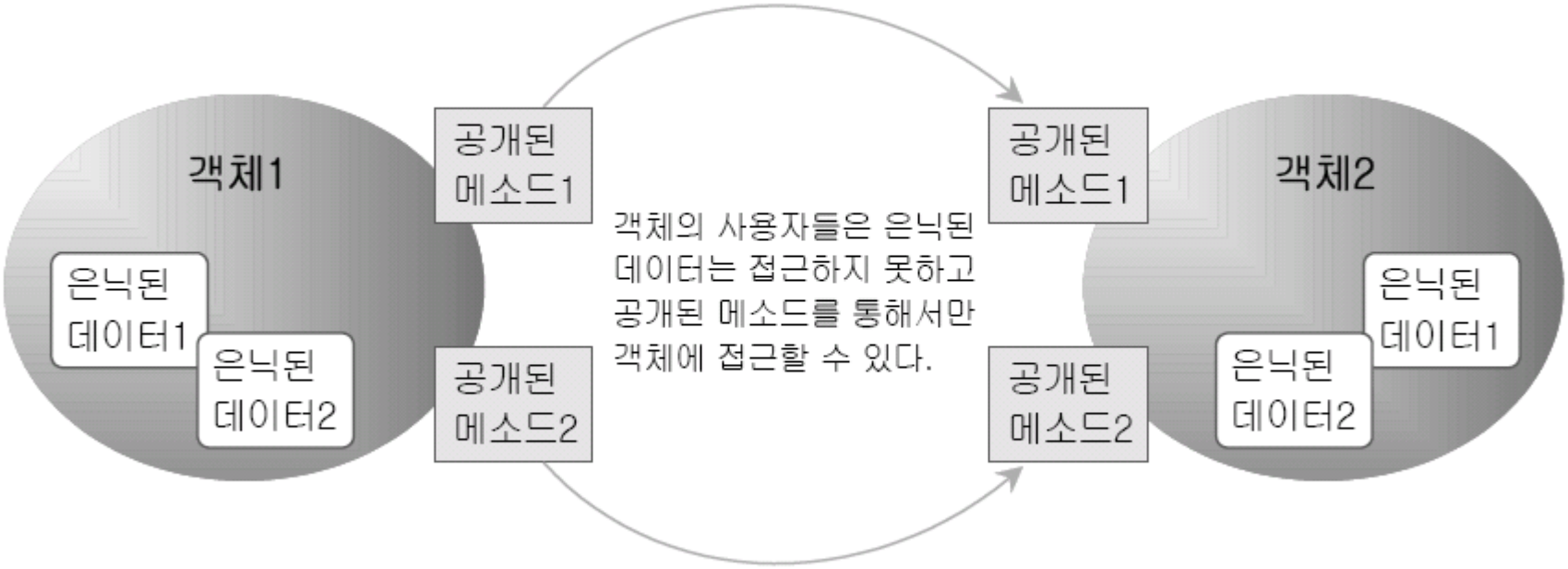
```
class Tv {
    String color;   // 색깔
    boolean power; //
        전원상태(on/off)
    int channel;    // 채널

    void power() { power = !power; } //
        전원on/off
    void channelUp( channel++;)       //
        채널 높이기      }
    void channelDown {channel--;}     //
        채널 낮추기
```

클래스의 설계

객체 지향 프로그래밍의 특징

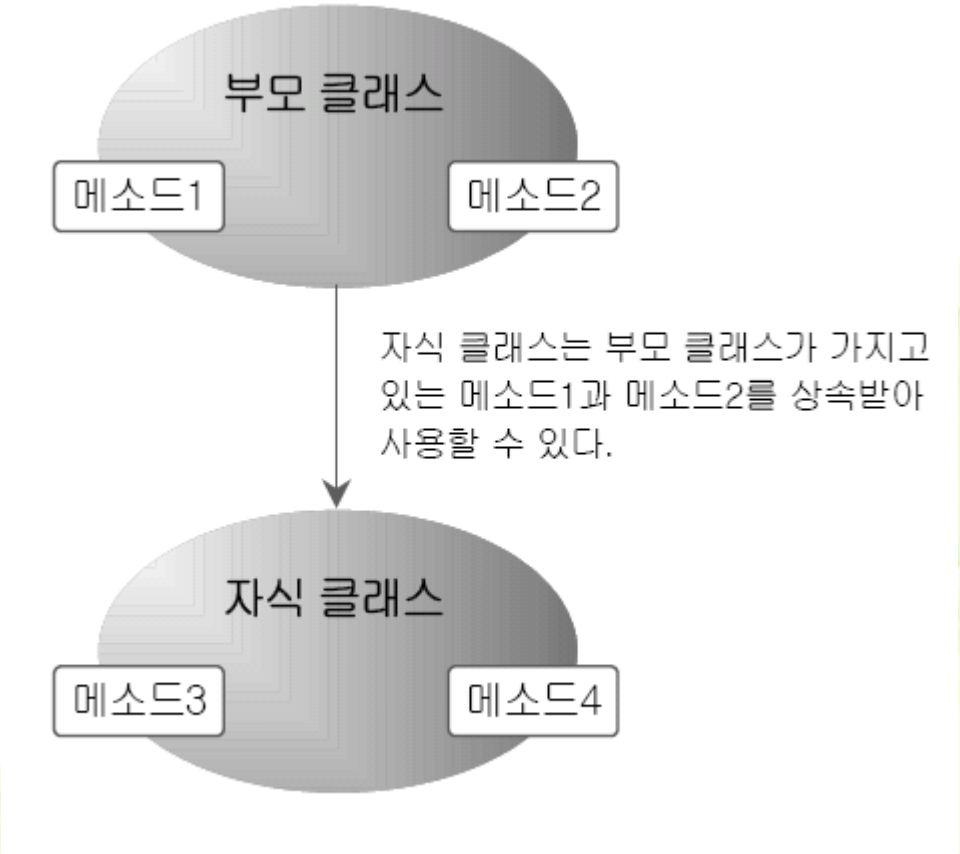
- (1) 캡슐화와 데이터 은닉



클래스의 설계

객체 지향 프로그래밍의 특징

- (2) 다형성과 메소드 오버로딩
- (3) 상속성



클래스의 예

```
Class Triangle  {
        //속성들(필드들)
        Point A;
        Point B;
        Point C;
        Color color;
        Image pattern;

        //메소드들
        void draw(){
                //실제 그리는 부분
        }
        void move(){
                //각 점 위치를 변경시키는 부분
        }
        void serColor(){
                //색깔을 변경시키는 부분
        }
}
```

클래스의 구성

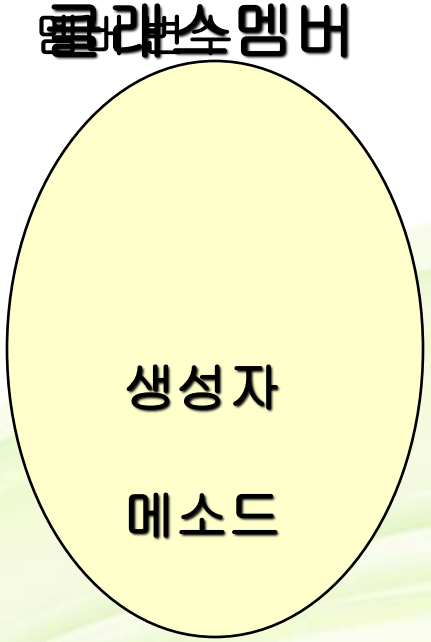
클래스의 일반구조

```
class Class-name {  //클래스 헤더 부분

  type1 varName1 = value1;   ……..
  typeN varNameN = valueN;

  Class-name() {
    //생성되는 객체의 초기화 과정을 기술
   …….. }
  Class-name(argsN) {
   …….. }

  mtype mName1(margs1)  { …….
    //메소드 기술
  }
  mtype mNameN(margsN)  {…….
  }
}
```



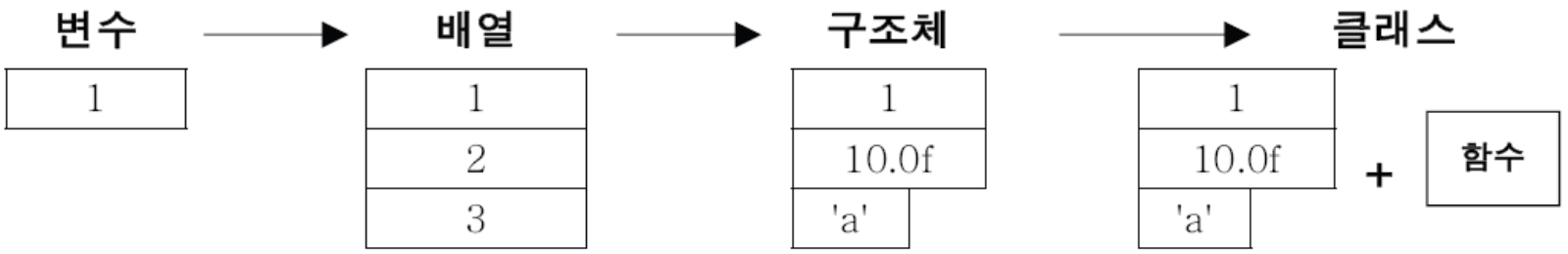
클래스의 일반구조-예

```
class SampleClass {   // 클래스 헤더부분
  int a;
  int b;     // 멤버 변수 부분
  int c;

  public SampleClass()  {
   // 생성자 부분. 이름이 클래스 명과 같다
      a = x;
      b = y;
      c = z;
  }
  public int sum() {   // 메소드 부분
      int d;
      d = a + b + c;
   }
}
```

클래스의 또 다른 정의

클래스 – 데이터와 함수의 결합



[그림6-3] 데이터 저장개념의 발전과정

- 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 함수의 결합(구조체+함수)

클래스의 선언

```
[접근 제어자] class 클래스명 [extends  상위 클래스명]
                                    [implements  인터페이스명] {
  .........    //클래스 멤버 부분
}
```

▶접근제어자 : 다른 클래스가 이 클래스를 참조할 때의 제한 사항을 지정하는 것이다.

▶class : 클래스 정의를 시작하는 키워드

▶클래스명 : 정의될 클래스의 이름을 지정

▶extends 와 implements : 클래스의 확장과 관련된 예약어

접근 제어자

▶public : 해당 클래스의 필드와 메소드의 사용을 다른 모든 클래스에 허용 그리고 이들은 서브클래스로 상속된다.

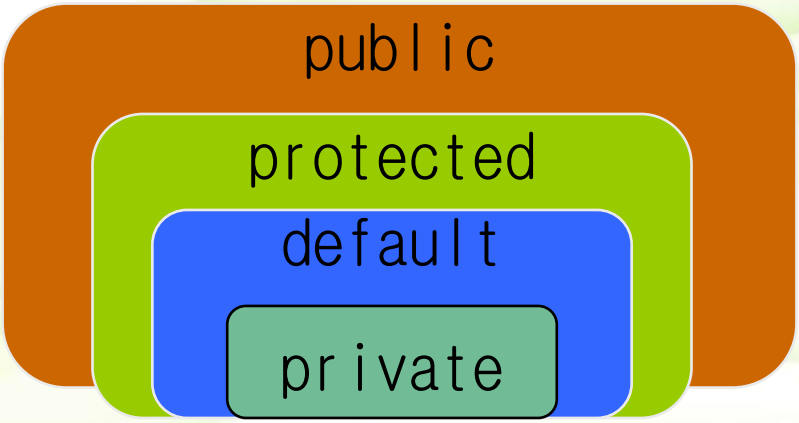
▶protected : 클래스의 멤버를 클래스 자신과 이 클래스로부터 상속받은 서브클래스에만 접근을 허용

▶private : 해당 클래스만이 이 멤버를 사용할수 있다. 외부객체에서는 절대로 접근을 할 수 없다.

▶default : 접근제어자를 명시하지 않은 경우의 디폴트 접근제어자이다. 같은 패키지내의 클래스들은 public 권한을 갖고 접근가능하다.

▶final : 서브클래스를 가질수 없는 클래스

▶abstract : 추상 메소드를 가지는 추상 클래스를 의미



클래스와 객체

객체지향 프로그램은 3단계로 진행된다.

자동차 객체를 예를 들어 설명하면 추상화 작업은 프로그램에 적합하게 자동차 객체를 설계하는 것이다.

클래스와 객체

형식	public class 클래스이름{
	  접근_지정자 자료형 필드;
	  접근_지정자 자료형 메소드( ){
	  :
	  :
	  }
	};

클래스와 객체

new 연산자가 힙 영역에 메모리할당 할당 후 되돌려 주는 주소는 레퍼런스 변수에 저장된다.

```
Car car01 = new Car( );
```

❶ 레퍼런스 변수 선언 ❷ 인스턴스 생성

레퍼런스 변수는 객체에 대한 참조(주소)를 갖게 된다.

레퍼런스 변수와 객체

new 연산자 다음에 클래스 명(Point)을 기술하면

- 실질적인 좌표 값(x, y)을 저장할 수 있는 기억 공간이 생성된다.

레퍼런스 변수의 역할

- new에 의해서 할당되는 기억 공간은 힙 영역인데 이곳은 실질적인 값을 저장할 수 있지만, 힙 영역은 직접 접근할 수 없기에 따로 레퍼런스 변수를 두고 접근한다.

Car car01 = **new Car( );** ➡

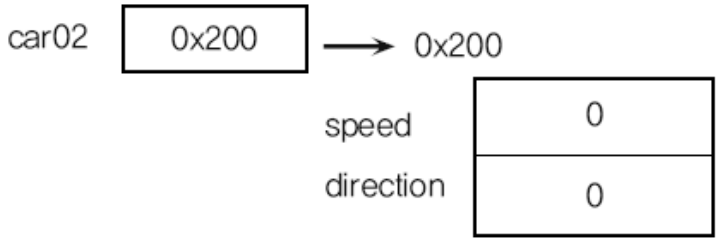
speed	0
direction	0

클래스와 객체

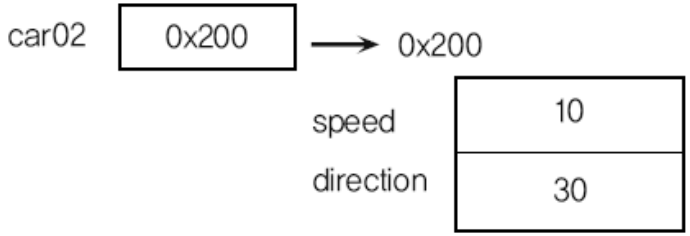
닷(.) 멤버 참조 연산자로 필드에 접근

```
car02 = new Car( );
```

레퍼런스 변수 인스턴스 생성




```
car02.speed = 10;
car02.direction = 30;
```



객체의 선언과 생성

레퍼런스 변수, 클래스객체

- 객체의 선언

  클래스명 객체참조변수(변수이름);

- 객체의 생성

  객체참조변수(변수이름) = new 클래스명();

- 객체의 선언과 생성

  클래스명 객체참조변수(변수이름) = new 클래스명();

```
class Box {
  int a;
  int b;
}
class MyBox {
.............
Box mybox1;
mybox1 = new Box();   // 또는 Box mybox1 = new Box();
.............
}
```

객체의 사용

객체참조변수는 해당 객체의 멤버변수와 메소드를 사용할수 있다.

사용법) 객체참조변수.메소드이름( );

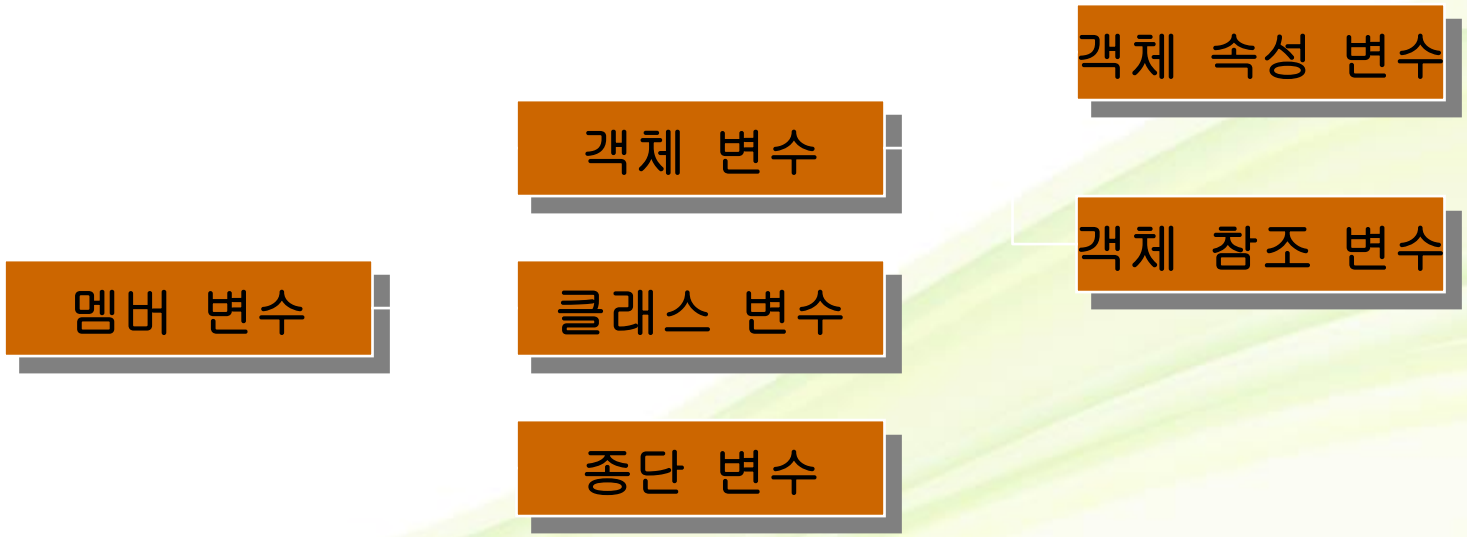
객체참조변수.멤버변수이름;

```
class Test{
   int max,min,sum;            ← 멤버변수
   void sum( ){
      sum=max+min;
   }
   public static void main(String[ ] arg){
      Test mytest = new Test();
      mytest.max=10;
      mytest.min=20;
      mytest.sum();
      System.out.println("두수의 합은="+mytest.sum);
   }
}
```

멤버변수

- 멤버 변수는 클래스내의 메소드 밖에서 선언된 모든 것을 의미한다
- 멤버 변수는 객체가 가질 수 있는 속성들을 나타낸다

- 멤버 변수의 구분



멤버변수 선언

▶멤버 변수 선언

[접근제어자] [static/final] 데이터형 변수명;

☞static : 클래스 변수, final : 종단 변수

☞static과 final이 붙지 않은 변수 : 객체변수(객체 속성변수 , 객체참조변수)

(1) 객체 변수(객체 참조변수와 객체 속성변수)

☞객체 변수 : 객체가 가질 수 있는 특성을 표현

☞객체 속성 변수 : 객체가 가질 수 있는 속성을 나타내는 값으로서 기본 자료형의 값들로 구성

☞객체 참조 변수 : 객체를 지정하는 변수. 자바에서는 기본 자료형을 제외한 모든 요소들을 객체로 취급

☞사용자는 객체를 생성한 다음 그 객체에 접근 하기 위해서는 객체 참조 변수를 통하여 그 객체의 멤버들에 접근할 수 있다.

멤버 변수

(1) 객체 참조변수와 객체 속성변수

```
class Box {
  int width;    // 객체 속성 변수 width
  int height;   // 객체 속성 변수 height
  int depth;    // 객체 속성 변수 depth
}
class MyBox {
  int vol;          // 객체 속성 변수 vol
  Box mybox1;       // 객체 참조 변수 mybox1
  Box mybox2;       // 객체 참조 변수 mybox2
  String boxname;   // 객체 참조 변수 boxname
                    // 자바에서 문자열은 객체로 취급한다
  mybox1 = new Box();
  mybox2 = new Box();
  ............
 }
```

멤버 변수 - 객체 참조변수와 객체 속성변수

객체 속성 변수 : 변수의 값이 복사되어 전달

```
int my_count1 = 100;
```

100
my_count1

```
int my_count2 = my_count1;
```

100
my_count2

객체 참조 변수 : 객체에 대한 주소가 복사되어 전달되므로
결국 같은 객체를 가르키게 된다

```
Box mybox1 = new Box();
```

객체의 주소
mybox1

```
Box mybox2 = mybox1;
```

객체의 주소
mybox2

width
height
depth

```
class A {
    public int x = 10;
    public int y = 20;
    int add() {
        return (x+y);
    }
}

class B {
    public static void main(String[] args) {
        A a = new A();

        System.out.println("x = " + a.x);
        System.out.println("y = " + a.y);
        System.out.println("Sum = " + a.add());
    }
}
```

```
class Grade {
    int kor  = 60; // 객체 속성 변수
    int eng  = 60; // 객체 속성 변수
    int math = 60; // 객체 속성 변수
}
class Result {
    public static void main(String[] args) {
        int total;
        double avg;
        Grade grade1 = new Grade();
        Grade grade2 = new Grade();
        grade1.kor = 70;
        grade2.eng = 90;
        total = grade1.kor + grade2.eng + grade2.math;
        avg = total / 3.0;
        System.out.println("과목 평균 = " + avg);
    }
}
```

```
package magic ;
class Package {
        int i = 0 ;
        int j = 10 ;
        int add(){
                return (i+j);
        }
}


class EXE{
        public static void main(String [] args){
                magic.Package p = new magic.Package();
                System.out.println( p.add());
        }
}
```

선언위치에 따른 변수의 종류

- 인스턴스변수(instance variable)
  - 각 인스턴스의 개별적인 저장공간. 인스턴스마다 다른 값 저장가능
  - 인스턴스 생성 후, '참조변수.인스턴스변수명'으로 접근
  - 인스턴스를 생성할 때 생성되고, 참조변수가 없을 때 가비지컬렉터에 의해자동제거됨
- 클래스변수(class variable)
  - 같은 클래스의 모든 인스턴스들이 공유하는 변수
  - 인스턴스 생성없이 '클래스이름.클래스변수명'으로 접근
  - 클래스가 로딩될 때 생성되고 프로그램이 종료될 때 소멸
- 지역변수(local variable)
  - 메서드 내에 선언되며, 메서드의 종료와 함께 소멸
  - 조건문, 반복문의 블럭{} 내에 선언된 지역변수는 블럭을 벗어나면 소멸

변수

선언위치에 따른 변수의 종류

"변수의 선언위치가 변수의 종류와 범위(scope)을 결정한다."

```
class Variables {
    int      instanceVar; // 인스턴스변수      클래스 영역
    static   int    staticVar; // 클래스변수
    void method( ) {
        int        localVar;   // 지역변수      메소드 영역
    }
}
```

변수의 종류	선언위치	생성시기
인스턴스변수	클래스 영역	인스턴스 생성 시
클래스변수		클래스가 메모리에 올라갈 때
지역변수	메소드 영역	해당 메소드가 호출될 때

변수

선언위치에 따른 변수의 종류

▶ 인스턴스변수(instance variable)

- 인스턴스 생성 후, **'참조변수.인스턴스변수명'**으로 접근
  - 생성 시기 : 인스턴스가 생성될 때

```
Variables  var = new Variable();          // 인스턴스 생성
var.instanceVar = 10    // 참조변수.인스턴스변수명
```

Class Variables

```
class Variables {
    int      instanceVar; // 인스턴스변수      클래스 영역
    static   int    staticVar; // 클래스변수
    void method( ) {
        int         localVar;   // 지역변수      메소드 영역
    }
}
```

변수

선언위치에 따른 변수의 종류

- 클래스변수(class variable)
  - 동일 클래스 내의 모든 인스턴스들이 공유하는 변수
  - 인스턴스 생성없이 **'클래스이름.클래스변수명'**으로 접근
  - 생성과 소멸 : 클래스가 로딩될 때 생성, 프로그램 종료될 때 소멸

```
Variables.staticVar = 10;        //  클래스이름.클래스변수명
```

```
class Variables {
    int      instanceVar; // 인스턴스변수      클래스 영역
    static   intstaticVar; // 클래스변수
    void method( ) {
      int       localVar;   // 지역변수      메소드 영역
    }
}
```

변수

선언위치에 따른 변수의 종류

▶ 지역변수(local variable)

- 메소드 내에 선언되며, 메소드가 실행될 때 생성되고, 종료와 함께 소멸
- 조건문, 반복문의 블럭{} 내에 선언된 지역변수는 블럭을 벗어나면 소멸

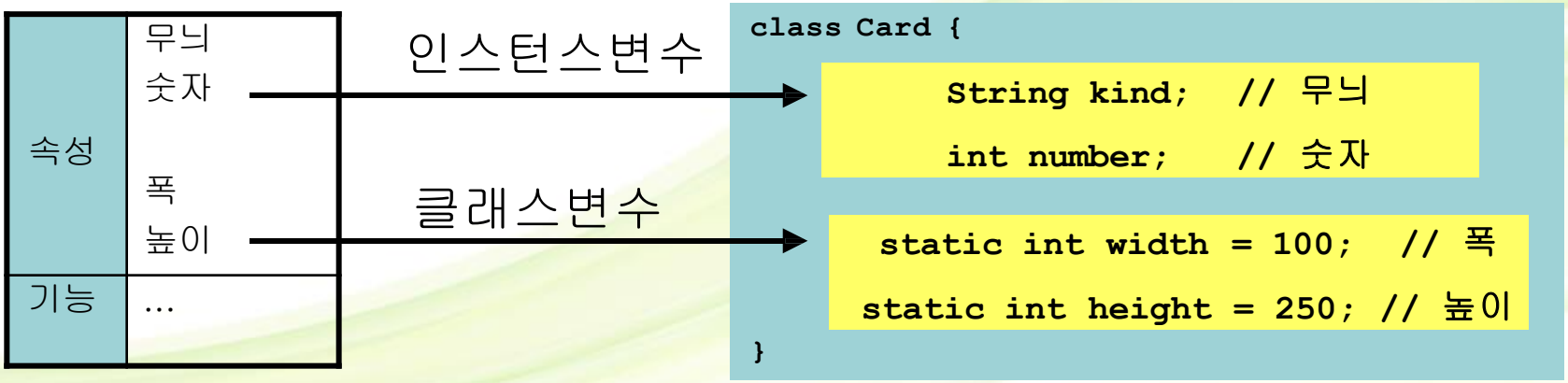
```
Variables   var = new Variable();          // 인스턴스 생성
var.methed()     // 참조변수.메소드명
```

```
class Variables {
    int      instanceVar; // 인스턴스변수      클래스 영역
    static   intstaticVar; // 클래스변수
    void method( ) {
      int       localVar;   // 지역변수      메소드 영역
    }
}
```

변수

클래스변수와 인스턴스변수

- 인스턴스변수 : 인스턴스가 생성될 때마다 생성, 인스턴스 마다 각기 다른 저장 공간 생성 - 동적변수
- 클래스변수 : 클래스가 로딩 될 때 한 번만 생성, 하나의 저장공간을 공유, 공통된 값을 갖는 경우 - 정적변수



c1 : Heart,7 크기 : 100,250

c2 : Heart,7 크기 : 50,80

```
public class CardTest {
     public static void main(String[] args) {
       Card  c1 = new Card();    // 객체1 생성
       c1.kind = "Heart";        // 인스턴스 변수 접근
       c1.number = 7;
       Card  c2 = new Card();    // 객체2 생성
       c2.kind = "Spade";        // 인스턴스 변수 접근
       c2.number = 4;
            System.out.print("c1 : " + c1.kind + ",” + c1.number);
       System.out.println(" 크기 : " + Card.width+ "," + Card.height); // 클래스 접근
        c1.width = 50;           // 참조변수로 클래스 변수 접근
       c2.height = 80;
       System.out.print("c2 : " +   c1.kind +","+c1.number);
       System.out.println(" 크기 : " + c2.width+","+c2.height);
    }
      }
class  Card{
    String kind;                 // 인스턴스 변수
     int number;
    static int width = 100;      // 클래스 변수
    static int height = 250;
}
```

멤버 변수 - 클래스 변수

▶자바에서의 변수는 다음과 같은 형식으로 선언된다

[public/private/protected] [static/final] 데이터형 변수이름;

☞클래스 변수는 static을 붙여 선언한다

☞클래스 변수는 전역변수의 개념을 가진다.

▶클래스 변수의 용도

☞객체 변수(객체참조,객체속성)는 객체가 생성될 때마다 각 객체에 변수들이 생성되지만, 클래스 변수는 클래스로부터 생성된 객체들의 수와 상관없이 하나만 생성

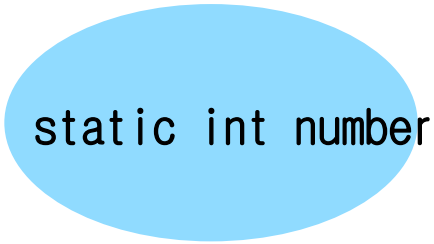
☞한 클래스로부터 생성된 모든 객체들은 클래스 변수를 공유

☞클래스 변수를 이용하여 객체들 사이의 통신에 사용하거나 객체들의 공통 속성을 나타낼 수 있다.

☞객체변수와는 달리 클래스 변수는 클래스 이름을 통하여 접근

멤버 변수 - 클래스 변수

클래스 Test



클래스 Test로 부터 생성된 모든 객체들은 Test.number로 클래스 변수에 접근 할 수 있다

객체생성

객체 고유 데이터

객체 a

객체 고유 데이터

객체 b

객체 고유 데이터

객체 c

객체 고유 데이터

객체 d

```
public class Static {
	public int instance_var=100;
	public static int static_var=100;
	public void instance_method(){
		System.out.println("instance_method() invoke...");
	}
	public static void static_method(){
		System.out.println("static_method() invoke...");
	}
}

/*  static 제한자
	1. 멤버변수,메소드 앞에 부칠 수 있다.
	2. static 제한자가 부터 있는 변수나 메소드는 객체생성없이 사용가능
	3. 클래스(정적,공용) 변수(메소드)
    형식   - 멤버변수:  public static int i;
	 - 멤버 메쏘드: public static int add(){}		*/
```

```
public class StaticMain {
        public static void main(String[] args) {
                //static 멤버접근
                Static.static_var=40000;
                System.out.println("Static.static_var:"+Static.static_var);
                Static.static_method();
                //instance 멤버접근
                Static instance=new Static();
                instance.instance_var=500;
                System.out.println("instance.instance_var:"+instance.instance_var);
                instance.instance_method();
                Static instance1=new Static();
                instance1.instance_var=700;
                System.out.println("instance.instance_var:"+instance1.instance_var);
                instance1.instance_method();
                //instance를통한 static 멤버접근
                instance.static_var=8900;
                System.out.println("instance.static_var:"+instance.static_var);
                instance.static_method();
        }
}
```

멤버 변수 - 클래스 변수

```
class Box {
  int width;  int height;   int depth;   long idNum;
  static long boxID = 0;    ← 클래스변수 선언
  public Box() {
    idNum = boxID++;
  }
}
class StaticDemo {
  public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
    Box mybox3 = new Box();
    Box mybox4 = new Box();
    System.out.println("mybox1의 id 번호 : " + mybox1.idNum);
    System.out.println("mybox2의 id 번호 : " + mybox2.idNum);
    System.out.println("mybox3의 id 번호 : " + mybox3.idNum);
    System.out.println("mybox4의 id 번호 : " + mybox4.idNum);
    System.out.println("전체 박스의 개수는 "+ Box.boxID + "입니다.");
  }
}
```

```
class Static
        static int num = 0;      // 클래스 변수 선언
        int a = 10;              // 객체속성 변수
        int b = 20;              // 객체속성 변수
}
class StaticRun
   public static void main(String[] args)  {
        Static s1 = new Static();
        Static s2 = new Static();
        s1.num = 10;
        s1.a = 20;
        s1.b = 30;

        System.out.println("s1의 값 num="+s1.num+" a="+s1.a+" b="+s1.b);
        System.out.println("s2의 값 num="+s2.num+" a="+s2.a+" b="+s2.b);
        System.out.println("클래스 변수 num = " + Static.num );
    }
 }
```

```
public class Car {
        public String color;    public String model;        public static int count;
        private Car() {       }
        public Car(String color, String model) {
                this.color = color;
                this.model = model;
                Car.count=Car.count+1;
        }
}
public class CarFactory {
        public static void main(String[] args) {
                Car.count=100;
                Car car1=new Car("yellow","audi");
                Car car2=new Car("red","lexus");
                Car car3=new Car("blue","k7");
                Car car4=new Car("white","k5");
                //Car car5=new Car();
                System.out.println("Car.count:"+Car.count);
                System.out.println("car4.count:"+car4.count);
        }
}
```

멤버 변수 - 종단(final) 변수

☞ 예약어 **final**을 사용하여 종단변수 지정

☞ 변할 수 없는 상수 값을 나타낸다

☞ 종단변수는 관례상 대문자로 표기한다.

```
final int MAX = 100;
final int MIN = 1;
```

```
class Circle {
   public static void main(String[] args) {
      final float PI = 3.1415f;  // 원주율
      int r = 10;                // 원의 반지름
      double area = PI * r * r;
      double round = 2 * PI * r;

      System.out.println("원의 넓이 = " + area);
      System.out.println("원의 둘레 = " + round);
   }
}
```

생성자(constructor)란?

- 생성자란?
  - 인스턴스가 생성될 때마다 호출되는 '**인스턴스 초기화 메서드**'
  - 인스턴스 변수의 초기화 또는 인스턴스 생성시 수행할 작업에 사용
  - 몇가지 조건을 제외하고는 메서드와 같다.
  - 모든 클래스에는 반드시 하나 이상의 생성자가 있어야 한다.

* 인스턴스 초기화 – 인스턴스 변수에 적절한 값을 저장하는 것.

```
Card c = new Card();
```

1. 연산자 new에 의해서 메모리(heap)에 Card클래스의 인스턴스가 생성된다.
2. 생성자 Card()가 호출되어 수행된다.
3. 연산자 new의 결과로, 생성된 Card인스턴스의 주소가 반환되어 참조변수 c에 저장된다.

생성자의 조건

▸ 생성자의 조건

- 생성자의 이름은 클래스의 이름과 같아야 한다.
- 생성자는 리턴값이 없다. (하지만 void를 쓰지 않는다.)

```
클래스이름(타입 변수명, 타입 변수명, ... ) {
    // 인스턴스 생성시 수행될 코드
      // 주로 인스턴스 변수의 초기화 코드를 적는다.
}
```

```
class Card {
      ...
    Card() {  // 매개변수가 없는 생성자.
        // 인스턴스 초기화 작업
    }
    Card(String kind, int number) { // 매개변수가 있는
        생성자
        // 인스턴스 초기화 작업
    }
}
```

기본 생성자(default constructor)

- 기본 생성자란?

  - 매개변수가 없는 생성자
  - 클래스에 생성자가 하나도 없으면 컴파일러가 기본 생성자를 추가한다.
    (생성자가 하나라도 있으면 컴파일러는 기본 생성자를 추가하지 않는다.)

```
클래스이름() { }
Card() { } // 컴파일러에 의해 추가된 Card클래스의 기본 생성자. 내용이
   없다.
```

"모든 클래스에는 반드시
하나 이상의 생성자가 있어야 한다."

생성자

매개변수가 있는 생성자

```
class Car {
      String color;          // 색상
      String gearType;       // 변속기 종류 - auto(자동), manual(수동)
      int door;              // 문의 개수

      Car() {}  // 생성자
      Car(String c, String g, int d) { // 생성자
            color = c;
            gearType = g;
            door = d;
      }
}
```

코드의 간소화

```
Car c = new Car();
c.color = "white";
c.gearType = "auto";
c.door = 4;
```

→

```
Car c = new Car("white","auto",4);
```

```
public class Constructor {
	int i;		int j;
	public Constructor(){
		//디폴트생성자
		System.out.println("Constructor()");
		this.i=100;
		this.j=100;
	}
	public Constructor(int i){  // 매개변수가 있는 생성자
		System.out.println("Constructor(int i)");
		this.i=i;
		this.j=100;
	}
	public Constructor(int i,int j){
		System.out.println("Constructor(int i,int j)");
		this.i=i;
		this.j=j;
	}
}
```

```
public class ConstructorMain {
	public static void main(String[] args) {
		//case1
		Constructor constructor1= new Constructor();
		System.out.println("**************************");
		System.out.println("i="+constructor1.i);
		System.out.println("j="+constructor1.j);
		//case2
		System.out.println("**************************");
		Constructor constructor2=new Constructor(900);
		System.out.println("i="+constructor2.i);
		System.out.println("j="+constructor2.j);
		//case 3
		System.out.println("**************************");
		Constructor constructor3=new Constructor(899, 898);
		System.out.println("i="+constructor3.i);
		System.out.println("j="+constructor3.j);

		DefaultConstructor dc=new DefaultConstructor();
	}
}
```

```
class Number   {
        int num;
        public Number(int n)        {
                num=n;
                System.out.println("인자 전달: "+n);
        }
        public int getNumber()      {
                return num;
        }
}
class Constructor2   {
        public static void main(String[] args)          {
                Number num1=new Number(10);
                System.out.println("메소드 반환 값: "+num1.getNumber());

                Number num2=new Number(20);
                System.out.println("메소드 반환 값: "+num2.getNumber());
        }
}
```

생성자 함수--예제

```
class Box {
  private int width;      // 변수를 private로 선언하여 외부에서 접근을 막는다
  private int height;     // 정보의 은폐 제공
  private int depth;
  private int vol;
  public Box(int a, int b, int c) {  // 클래스의 이름과 동일하게 생성자함수 선언
    width = a;   // 초기화 작업 수행
    height = b;
    depth = c;
  }
  public int volume() {
    vol =  width * height * depth;
    return vol;
  }
}
class BoxTestDemo {
  public static void main(String args[]) {
    int vol;
    Box mybox1 = new Box(10, 20, 30);
    vol = mybox1.volume();
    System.out.println("mybox1 객체의 부피 : " + vol);
  }
}
```

```
class Info {
   private String name;    // 이름
   private int age;          // 나이
   private char sex;        // 성별
   public Info(String n, int a ,char s) { // 생성자
      name = n;         age = a;      sex = s;
   }
    public void display() { // 정보 출력
       System.out.println("이름 = " + name);
       System.out.println("나이 = " + age);
       System.out.println("성별 = " + sex);
    }
}
class InfoExe{
    public static void main(String[] args) {
       Info info1 = new Info("홍길동", 20, 'm'); // info1 초기화
       Info info2 = new Info("홍길순", 19, 'f'); // info2 초기화
       info1.display();   // info1 화면 출력
       info2.display();   // info2 화면 출력
    }
}
```

생성자

[연습예제] 생성자 오버로딩

- ✓ 매개변수 1개인 경우 : 정사각형의 넓이 = w * w
- ✓ 매개변수 2개인 경우 : 직사각형의 넓이 = w * h
- ✓ 매개변수 3개인 경우 : 직육면체의 부피 = w * h * d

62

생성자

[연습문제] 생성자 오버로딩

```
class Calc{
int a, b, c;
  Calc(int w){          // 생성자 오버로딩
        square(w);
  }
    Calc(int w, int h){
        rectangle(w, h);
  }
  Calc(int w, int h, int d){
          hexa(w, h, d);
           }
   void square(int w){
        this.a = w * w;
    }
    void rectangle(int w, int h){
          this.b = w * h;
    }
  void hexa(int w, int h, int d){
        this.c = w * h * d;
    }
}
```

```
      public class MethodOverload {
public static void main(String[] args) {
          Calc o1 = new Calc(5);
    Calc o2 = new Calc(5, 10);
  Calc o3 = new Calc(5, 10, 2);
  System.out.println("정사각형 넓이 : " + o1.a);
  System.out.println("직사각형 넓이 : " + o2.b);
  System.out.println("직육면체 부피 : " + o3.c);
 }
 }
```

```
public class CarConstructor {
	//속성(멤버변수)
	private String no ;    private int inTime;   private int outTime;         private int fee;
	//행위(메쏘드)
	public CarConstructor() {  System.out.println("CarConstructor()");  }
	public CarConstructor(String no,int inTime,int outTime,int fee){
		System.out.println("CarConstructor(String no,int inTime,int outTime,int fee)");
		this.no=no; this.inTime=inTime;  this.outTime=outTime; this.fee=fee;
	}
	//1.요금계산
	public void calculateFee(){ this.fee = (this.outTime-this.inTime)*1000;          }
	//2.정보출력
	public void print(){
		System.out.println("********<<"+this.no+">>******");
		System.out.println("차량번호:"+this.no);
		System.out.println("입차시간:"+this.inTime);
		System.out.println("출차시간:"+this.outTime);
		System.out.println("주차요금:"+this.fee);
		System.out.println("**********************");
	}
	//3.DATA
	//set
	public void setNo(String no){      this.no=no;                }
	public void setInTime(int inTime) {  this.inTime = inTime;           }
	public void setOutTime(int outTime) {        this.outTime = outTime;             }
	public void setFee(int fee) {        this.fee = fee;             }
```

```
        public int getFee(){ return this.fee;        }
        public String getNo() {  return no;          }
        public int getInTime() { return inTime;  }
        public int getOutTime() {  return outTime;          }
}
public class CarConstructorMain {
   public static void main(String[] args) {
        //case1
        CarConstructor car1=new CarConstructor();
        car1.setNo("1111");
        car1.setInTime(12);
        //case2
        CarConstructor car2 = new CarConstructor("2222", 13, 0, 0);
        car1.setOutTime(14);
        car1.calculateFee();
        car1.print();
        car2.setOutTime(14);
        car2.calculateFee();
        car2.print();
   }
}
```

생성자

[연습문제] 매개변수 있는 생성자를 이용하여 멤버변수 초기화

```
public class Tv {
    String color;           // 색깔
   boolean power;    // 전원상태(on/off)
    int channel;            // 채널

  Tv(String c, boolean p, int ch){    // 매개변수를 갖는 생성자
    color = c;
    power = p;
    channel = ch;
  }
  void power() { power = !power; }    // 전원on/off
  void channelUp(){channel++;}        // 채널 높이기
  void channelDown() {channel--;}     // 채널 낮추기

 public static void main(String[] args) {
      Tv t = new Tv("Green", false, 10); // 생성자로 객체 생성
           System.out.println("TV 색상 : " + t.color);
          System.out.println("전원 상태 : " + t.power);
         System.out.println("현재 채널 : " + t.channel);
  }
}
```

TV 색상 : Green

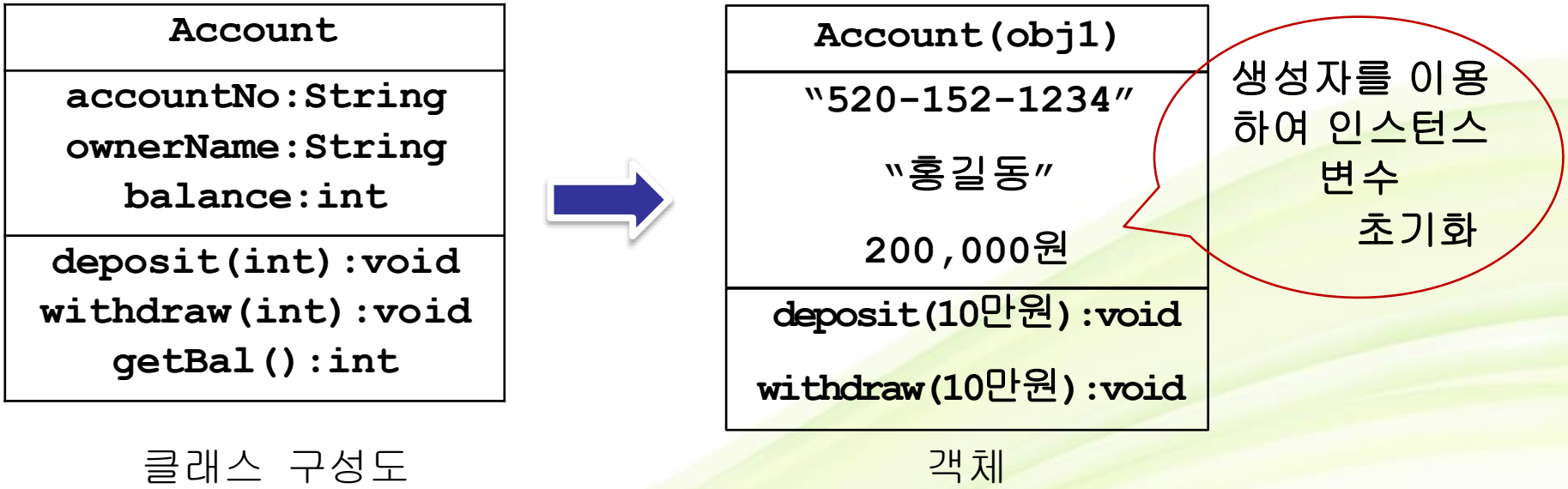
전원 상태 : false

현재 채널 : 10

생성자

[연습문제] 은행 계좌(Account) 클래스 설계

구분	내용
속성	계좌번호, 예금주, 잔액
기능	예금하다. 인출하다.



클래스 구성도

객체

생성자

```
public class Account {
   String accountNo;
   String owerName;
   int balance;
 Account(String a, String o, int b){
   accountNo = a;
   owerName = o;
   balance = b;
 }
 void deposit(int b){
      balance += b;
 }
 int withdraw(int w){
  if(balance < w)
    return 0;
  else
     return balance -= w;
 }
 int getBal() {
     return  balance;
}
```

```
public static void main(String[] args) {
   Account obj1 = new Account("520", "홍길동", 200000);
   Account obj2 = new Account("425", "이순신", 500000);
        System.out.print(obj1.owerName);
   obj1.deposit(100000);              // 예금하기
   System.out.println(": "+ obj1.getBal());
       System.out.print(obj2.owerName);
   int bal = obj2.withdraw(300000);  // 인출하기
   if(bal == 0)
     System.out.println(" 잔액이 부족합니다");
   else
     System.out.println(": "+obj2.getBal());
  }   // main()
}   // Account
```

홍길동: 300000

이순신: 400000

Account

```
public class Account {
        String accountNo, name;        int balance;
        Account(String a, String n, int b) {
                accountNo = a; name= n; balance = b;
        }
        void deposit(int money) {
                balance += money;
                System.out.println(name +"님 입금액 "+money);
        }
        void withdraw(int money) {
                if (balance >= money) {
                        balance -=money;
                        System.out.println(name +"님 출금액 "+money);
                } else System.out.println("돈도 없는 놈이 꺼져");
        }
        void print() {
                System.out.println("예금주 : "+name);
                System.out.println("현재 잔액 : "+balance);
                System.out.println("===============");
        }
}
```

AccountEx

```
public class AccountEx {
	public static void main(String[] args) {
		int money = 0;
		Account at1 = new Account("신한1234", "홍길동", 1000);
		for (int i=0; i< 5;i++) {
			money = (int)(Math.random()*1000)+100;
			at1.deposit(money);
			money = (int)(Math.random()*1500)+100;
			at1.withdraw(money);
			at1.print();
		}
		// 통장 개설후에 입금 및 출금
		Account at2 = new Account("국민3456", "정유라", 2000);
		for (int i=0; i< 5;i++) {
			money = (int)(Math.random()*1000)+100;
			at2.deposit(money);
			money = (int)(Math.random()*2500)+100;
			at2.withdraw(money);   at2.print();
		}
	}
}
```

생성자

[실습문제] 생성자를 이용하여 인터넷가입자 정보를 멤버변수에 초기화한 후 다음과 같이 콘솔에 출력하시오.

인터넷가입정보 클래스

SubscriberInfo

name: String

이름

id : String

아이디

password : String

패스워드

phonNo : String

전호번호

address : String

주소

SubscriberInfo( )

생성자( )

changePasswd( ) :void

패스워드 변경( )

<<creates>>

Class13

+main() : void

처리 내용

1. 생성자를 이용하여 멤버변수 초기화 (임의 값)
2. 멤버 변수 초기화 값 출력(패스워드 제외)
3. 패스워드 변경/출력

콘솔 출력결과

```
 name     id      전화번호     주소
홍길동 kimjs   850-2525   서울시
기존 패스워드 :1234
신규 패스워드 :kjs1234
```

생성자

```
class SubscriberInfo{
 String name, id, phonNo, address;
 private String passWord;
 //SubscriberInfo(){}
 SubscriberInfo(String n, String i, String
    ph, String pa, String a){
  name = n;
  id=i;
  phonNo = ph;
  passWord=pa;
  address = a;
 }
 void changePasswd(String pa){
     this.passWord = pa;
 }
  String passwd_getter(){
      return passWord;
 }
}
```

```
public class SubscriberInfoMain {
   public static void main(String[] args) {
       SubscriberInfo obj;
       obj = new SubscriberInfo("홍길동","kimjs",
             "850- 2525","1234", "서울시");
      System.out.println(obj.name+" "+obj.id +" "+obj.phonNo+"
             +obj.address);
       System.out.println("기존 패스워드    :“
            +obj.passwd_getter()); obj.changePasswd("kjs1234");
       System.out.println("신규 패스워드    :“
           +obj.passwd_getter());
   }
}
```

홍길동 kimjs 850-2525 서울시

기존 패스워드 :1234

신규 패스워드 :kjs1234

변수의 초기화 – 예시(examples)

선언예	설 명
int i=10; int j=10;	int형 변수 i를 선언하고 10으로 초기화 한다. int형 변수 j를 선언하고 10으로 초기화 한다.
int i=10, j=10;	같은 타입의 변수는 콤마(,)를 사용해서 함께 선언하거나 초기화 할 수 있다.
int i=10, long j=0;	타입이 다른 변수는 함께 선언하거나 초기화할 수 없다.
int i=10; int j=i;	변수 i에 저장된 값으로 변수 j를 초기화 한다. 변수 j는 i의 값인 10으로 초기화 된다.
int j=i; int i=10;	변수 i가 선언되기 전에 i를 사용할 수 없다.

```
class Test
{
    int j = i;
    int i = 10; // 에러!!!
}
```

```
class Test
{
    int i = 10;
    int j = i;  // OK
}
```

멤버변수의 초기화

- 멤버변수의 초기화 방법

1. 명시적 초기화(explicit initialization)

```
class Car {
    int door = 4;               // 기본형(primitive type) 변수의 초기화
    Engine e = new Engine();    // 참조형(reference type) 변수의 초기화

    //...
}
```

```
Car c = new Car();
c.color = "white";
c.gearType = "auto";
c.door = 4;
```

→

```
Car c = new Car("white","auto",4);
```

2. 생성자(constructor)

```
Car(String color, String gearType, int door){
    this.color = color;
    this.gearType = gearType;
    this.door = door;
}
```

3. 초기화 블럭(initialization block)

- 인스턴스 초기화 블럭 : { }
- 클래스 초기화 블럭 : static { }

초기화 블럭(initialization block)

- 클래스 초기화 블럭 – 클래스변수의 복잡한 초기화에 사용되며 클래스가 로딩될 때 실행된다.
- 인스턴스 초기화 블럭 – 생성자에서 공통적으로 수행되는 작업에 사용되며 인스턴스가 생성될 때 마다 (생성자보다 먼저) 실행된다.

```
class InitBlock {
    static { /* 클래스 초기화블럭 입니다. */ }

    { /* 인스턴스 초기화블럭 입니다. */ }

    // ...
}
```

```
1  class StaticBlockTest {
2      static int[] arr = new int[10]; // 명시적 초기화
3
4      static { // 배열 arr을 1~10사이의 값으로 채운다.
5          for(int i=0;i<arr.length;i++) {
6              arr[i] = (int)(Math.random()*10) + 1;
7          }
8      }
9      //...
10 }
```

멤버변수의 초기화 시기와 순서

- 클래스변수 초기화 시점 : 클래스가 처음 로딩될 때 단 한번
- 인스턴스변수 초기화 시점 : 인스턴스가 생성될 때 마다

```
1  class InitTest {
2      static int cv = 1;  // 명시적 초기화
3      int iv = 1;         // 명시적 초기화
4
5      static {    cv = 2; }   // 클래스 초기화 블럭
6      {   iv = 2; }           // 인스턴스 초기화 블럭
7
8      InitTest() { // 생성자
9          iv = 3;
10     }
11 }
```

```
InitTest it = new InitTest();
```

클래스 초기화			인스턴스 초기화			
기본값	명시적 초기화	클래스 초기화블럭	기본값	명시적 초기화	인스턴스 초기화블럭	생성자
cv 0	cv 1	cv 2	cv 2 iv 0	cv 2 iv 1	cv 2 iv 2	cv 2 iv 3
1	2	3	4	5	6	7

```
class BlockTest {
        static {
                System.out.println("static { }");
        }
        {
                System.out.println("{ }");
        }
        public BlockTest() {
                System.out.println("생성자");
        }
        public static void main(String args[]) {
                System.out.println("BlockTest bt = new BlockTest(); ");
                BlockTest bt = new BlockTest();

                System.out.println("BlockTest bt2 = new BlockTest(); ");
                BlockTest bt2 = new BlockTest();
        }
}
```

```
class StaticBlockTest {
	static int[] arr = new int[10];

	static {
		for(int i=0;i<arr.length;i++) {
			// 1과 10사이의 임의의 값을 배열 arr에 저장한다.
			arr[i] = (int)(Math.random()*10) + 1;
		}
	}

	public static void main(String args[]) {
		for(int i=0; i<arr.length;i++)
			System.out.println("arr["+i+"] :" + arr[i]);
	}
}
```