8. 클래스와 객체

객체지향 기술(Object Oriented Technology)

- **객체지향 기술은 소프트웨어 부품화, 소프트웨어 컴포넌트의 재사용을 주요 목표로 한다.**

- **객체지향 언어 개념과 대비되는 언어로서 절차지향(procedural-oriented) 언어가 있다.**

- **객체란?**
  - 효율적으로 정보를 관리하기 위해서 사람들이 의미를 부여하고 분류하는 논리적인 단위
  - 실세계에 존재하는 모든것이 될 수 있으며, 명사로서의 성질을 갖춘 모든 것은 객체로 만들어질 수 있다.

객체지향 기술(Object Oriented Technology)

❖ 객체지향의 장점

- 문제를 쉽고 자연스럽게 프로그램화(모델링) 할 수 있다
- 쉬운 프로그램의 개발로 인한 생산성 향상 시킬 수 있다
- 프로그램 모듈을 재사용 할 수 있다
- 프로그램의 확장 및 유지 보수가 용이하다

클래스(Class)와 객체(Object)

❖ 객체의 구성요소

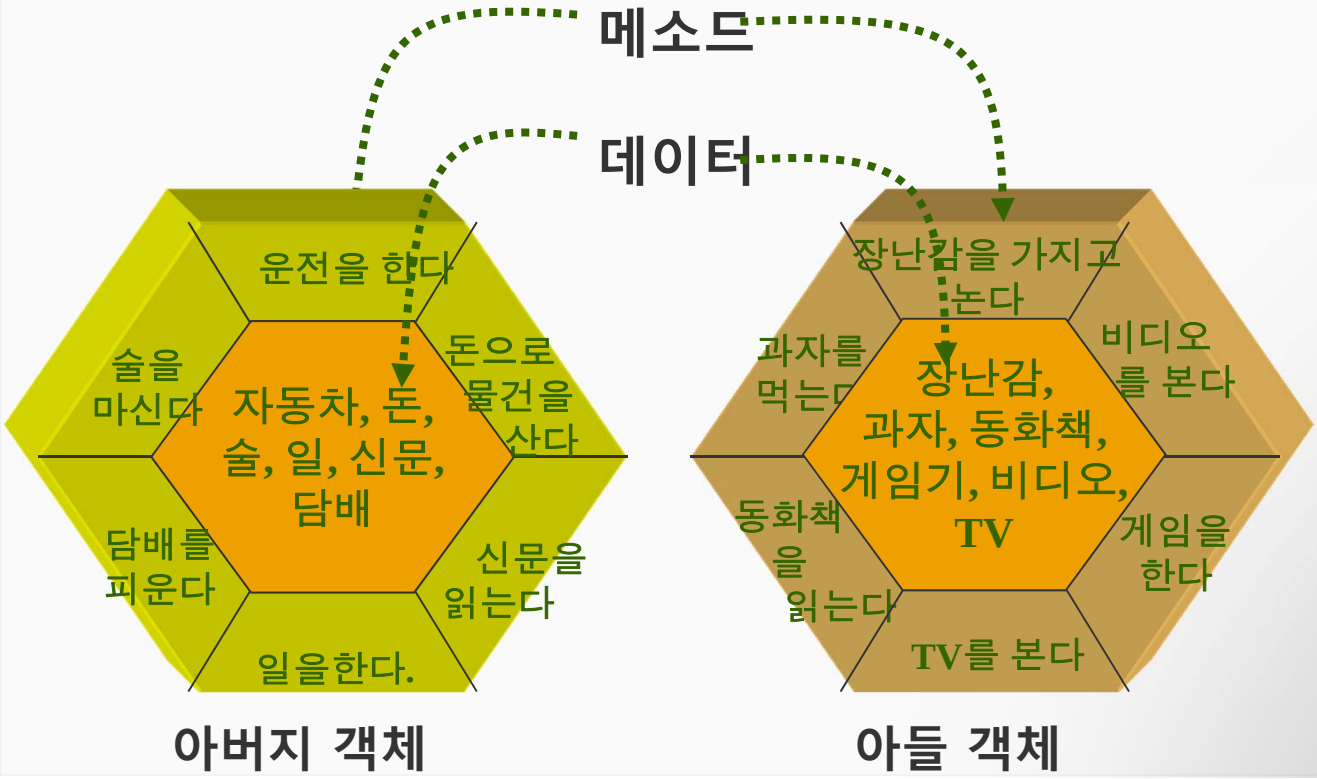
- 속성
  - attributes, information, data, field
  - 객체의 특성을 표현하는 정적인 성질
- 행위
  - message, action, behavior
  - 객체들간에 서로 영향을 주고받을 수 있게 하는 동적인 행동

❖ 클래스

- 하나의 클래스로부터 여러 개의 객체를 생성하기 위해사용되는 형판(template)

객체(Object)

❖ 객체는 프로그래머에 의해 모델링 된다

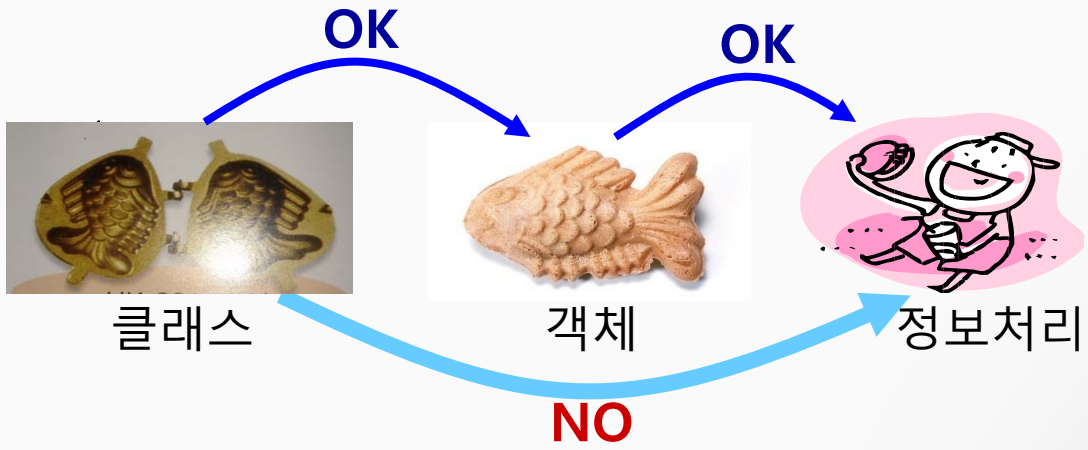


클래스(Class)

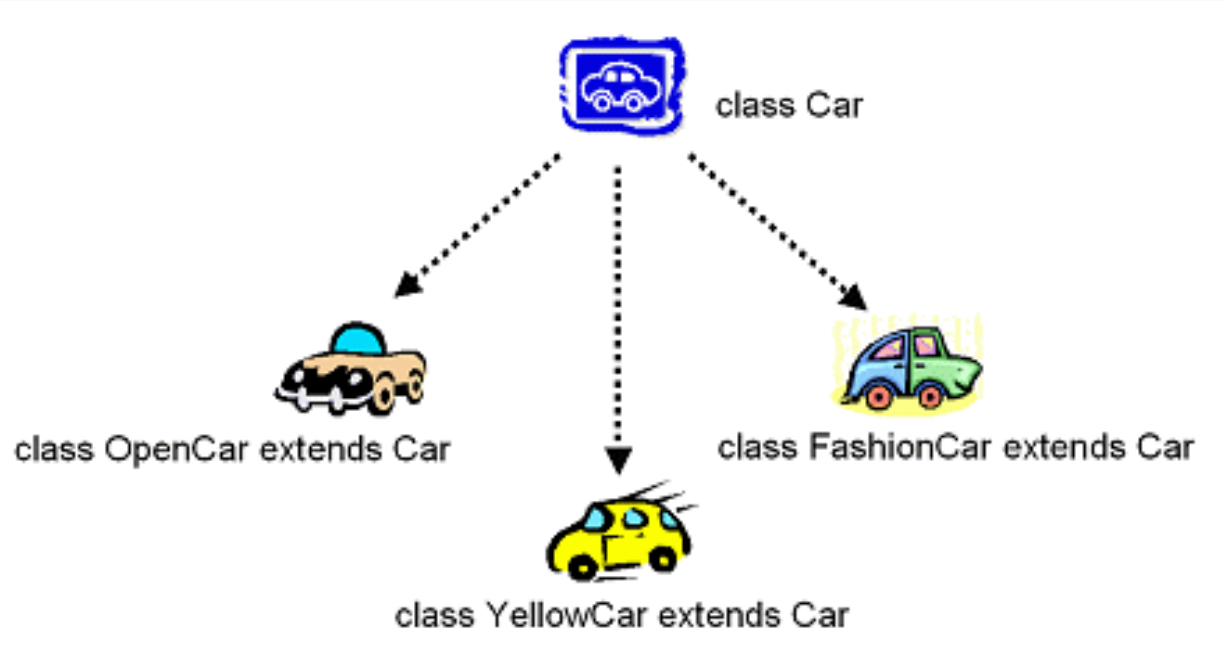
- 객체는 항상 클래스로부터 생성된다. 즉 클래스는 객체를 생성하는 형판(template)
- 클래스는 두개의 구성요소(member)인 자료구조(필드)와 연산(메소드)을 가진다
- 클래스로부터 생성된 객체를 instance라 한다.

  객체 = instance
- 정보처리의 주체는 클래스가 아니라 객체이다
- 객체지향 프로그래밍의 시작은 클래스의 생성이다

클래스(Class)와 객체(Object)

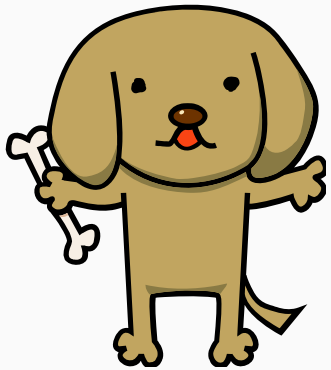


클래스(Class)와 객체(Object)



❖ **위 그림에서 OpenCar, YelloCar, FashionCar는 모두 클래스 Car의 객체입니다.**

클래스(Class)와 객체(Object)



누렁이
1살

낑낑

발발이
2살

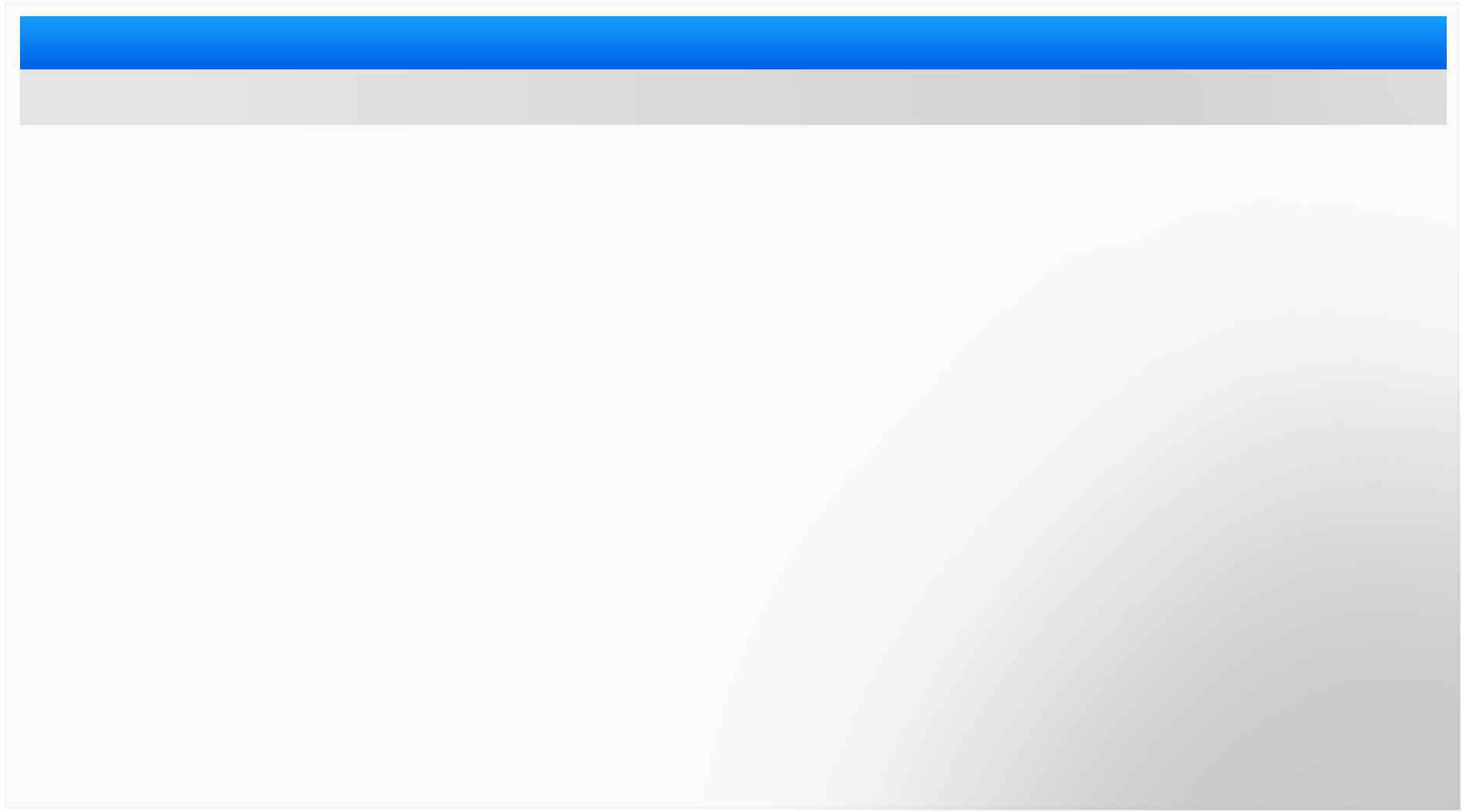
월월

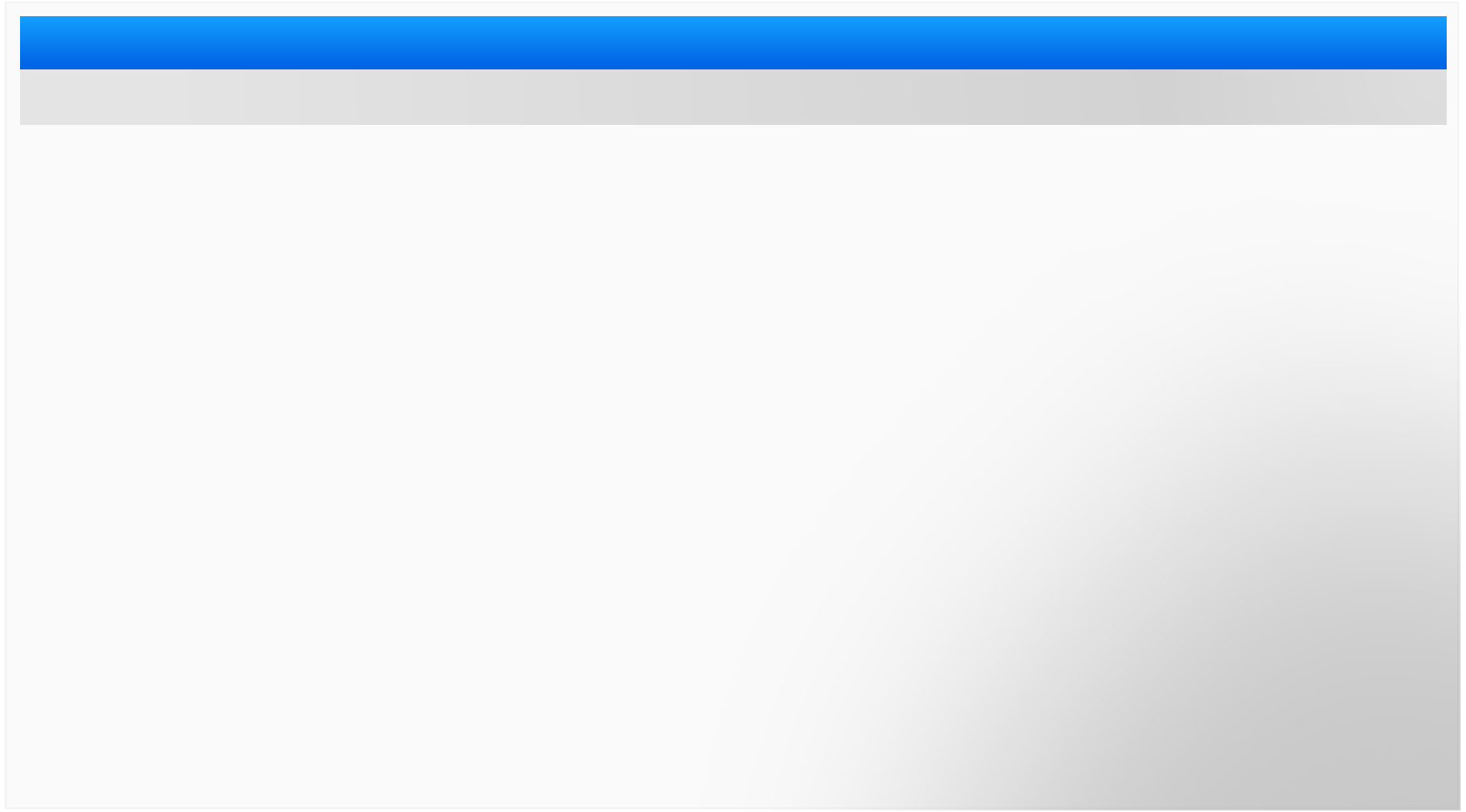
해피
3살

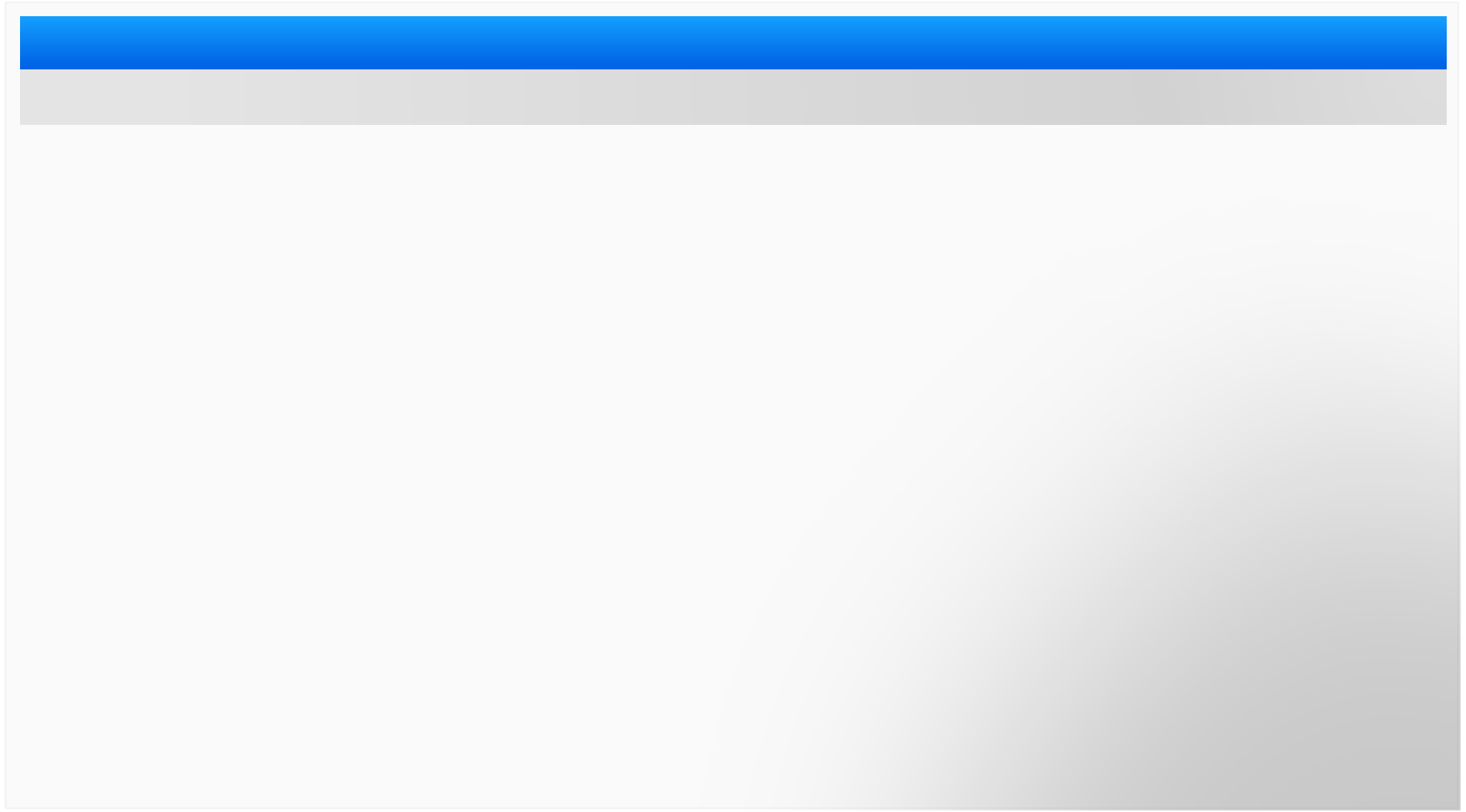
멍멍

클래스(Class)와 객체(Object)

```
String[3] name = {“누렁이”, “발발이”,
“해피”};

int[] age = {1, 2, 3};
...
void bark(int dogNumber)
{
  if(age[dogNumber < 2)
            System.out.println(“낑낑”);
  else
            System.out.println(“멍멍”);
}
```





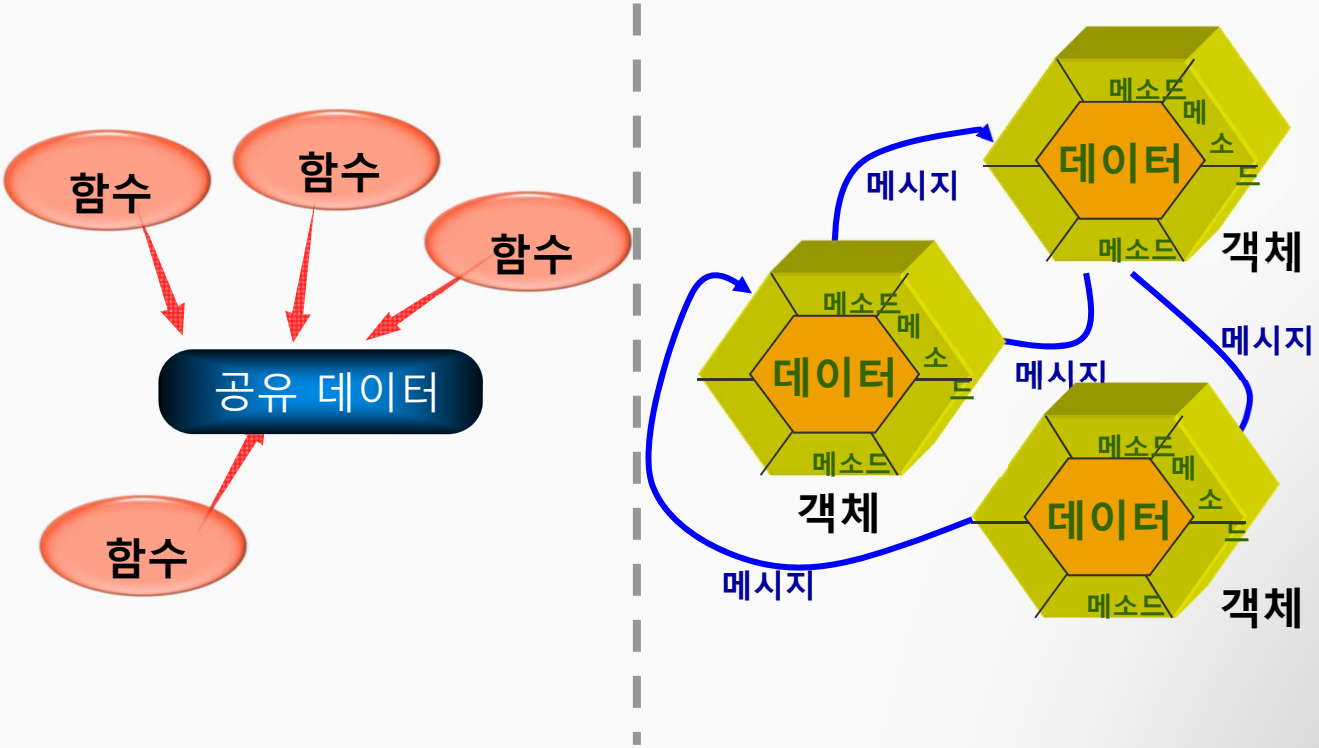


객체지향프로그래밍 (Object-Oriented Programming)

- ***Procedural Programming*** **vs.** ***OOP***
  - *Procedural programming*
  - *절차지향 언어에서는 프로그램이 대부분 데이터를 변화시키는 알고리즘 중심으로 구성된다. 따라서, 프로그램을 작성하기 위해서는 모든 데이터 구조를 이해하고 있어야 한다.*
  - *Object-oriented programming*
  - *객체지향 언어는 프로그램이 객체들의 집합이고, 객체는 자신이 가지는 고유의 데이터와 그 데이터를 처리할 수 있는 메소드를 가지고 있는 하나의 단위이다.*

객체지향프로그래밍 (Object-Oriented Programming)

❖ ***Procedural Programming*** **vs.** ***OOP***



객체지향프로그래밍 (Object-Oriented Programming)

❖ OOP

- Object가 프로그램의 기본 단위
- 하나의 프로그램은 여러 종류의 object들의 집합으로 표현

❖ Object in OOP

- 정의 : an object is a software bundle of member variables and related methods
  - Member variable (멤버변수): 데이터 또는 상태(state)를 나타냄
  - Method (함수): 행위(behavior)를 나타냄

객체의 특성

❖ 멤버 변수 (member variable)

- 객체가 가지는 데이터 또는 상태를 표현
- 각각의 객체를 구별 할 수 있음
  - My Dog : *name* = '누렁이' and age = '1살'
  - My Dog is *hungry* now

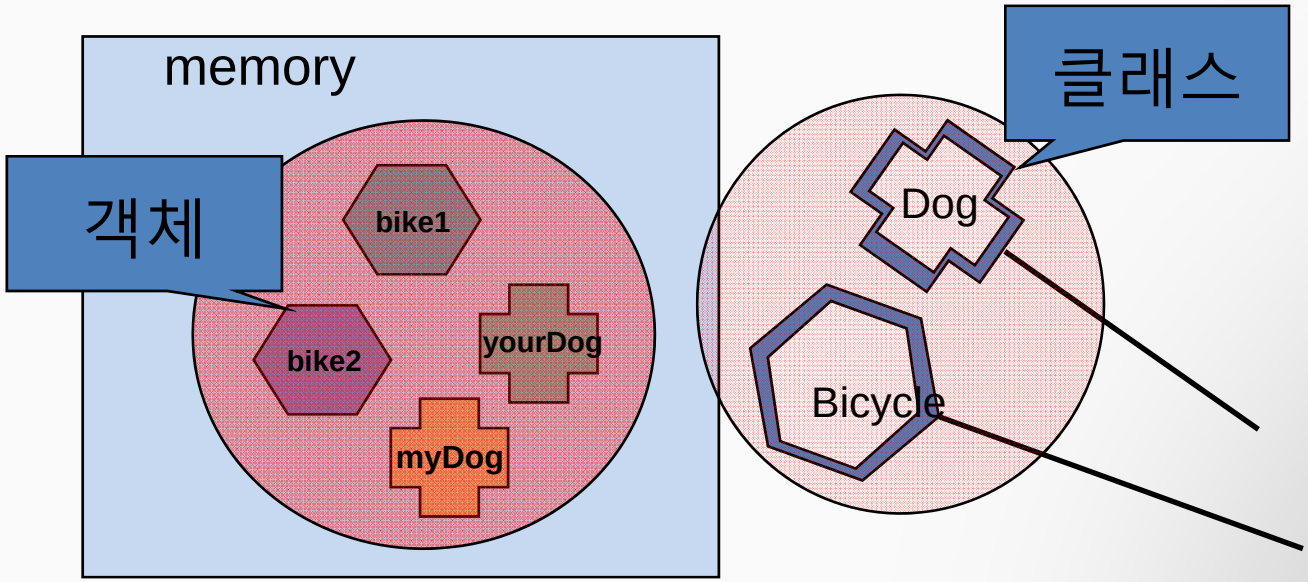
❖ 함수 (method)

- 객체가 할 수 있는 행위
  - **"짖다"**
- 객체의 멤버 변수의 값을 바꿀 수 있음
  - 이름, 색깔, 같은 멤버 변수 값 변화

자바에서 객체 만들기

❖ 객체 생성과 그 사용

- 먼저 객체의 정의(definition), 즉 **class** 만듦
- 만든 클래스를 이용해서 **객체 생성**
- 그 후, 만든 **객체를 사용**



Class 정의 하기

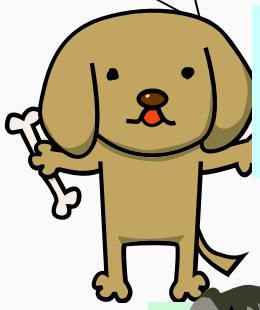
Class Definition

Class Name: dog

Variable:
- name
- age

Methods:
- bark

객체(Object)



누렁이
1살

낑낑

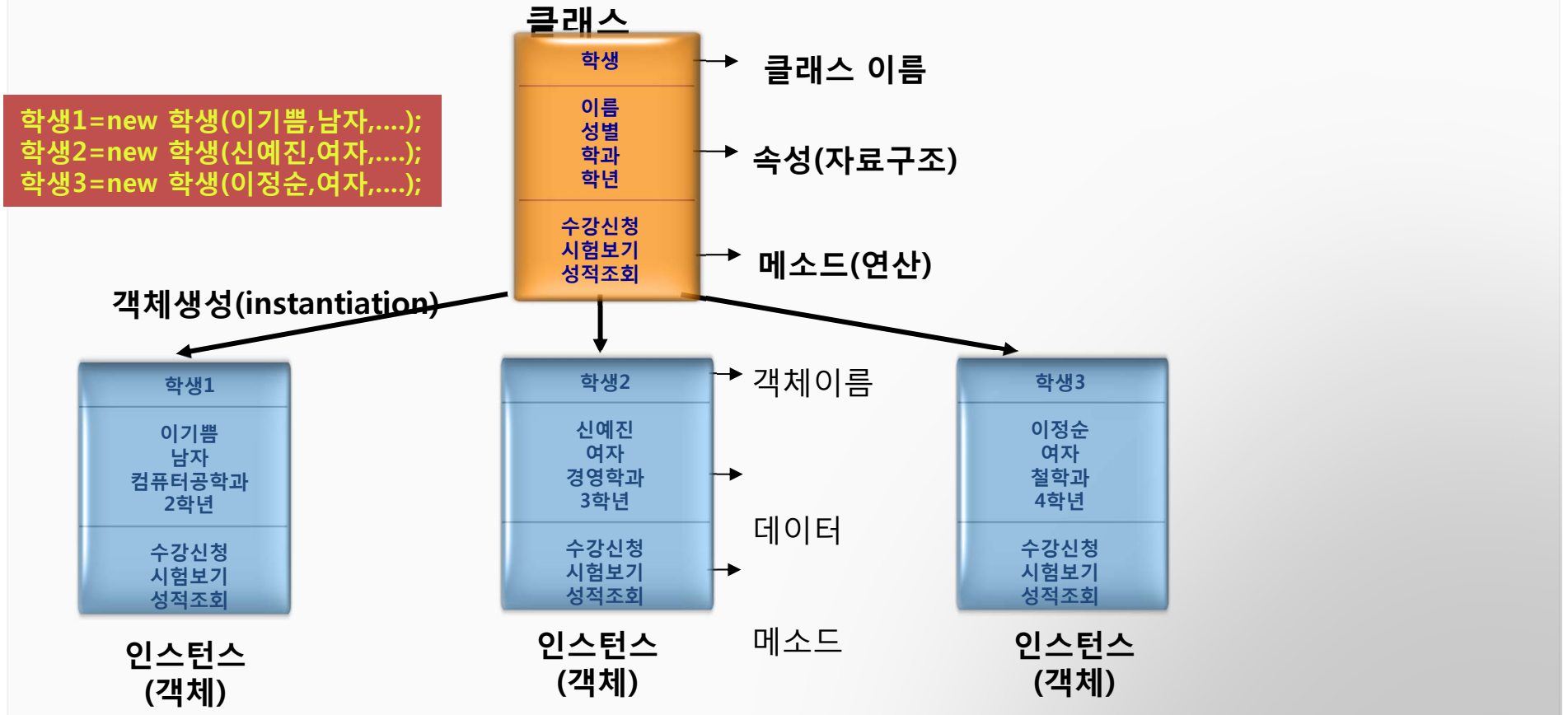
해피
3살

멍멍

발발이
2살

낑낑

Class 정의 및 객체 생성



Java로 Class 정의 하기

Class Definition

Class Name: dog

Variable:
- name
- age

Methods:
- bark

```
public class Dog {

        String name;
        int age;

        void bark()
        {
                if (age < 2)
                  System.out.println("낑낑");
                else
                  System.out.println("멍멍");
        }
}
```

객체 생성 하기

- **객체 생성 : use *new* keyword**

객체 이름, object를 가리키는 reference

```
Dog myDog = new Dog();
```

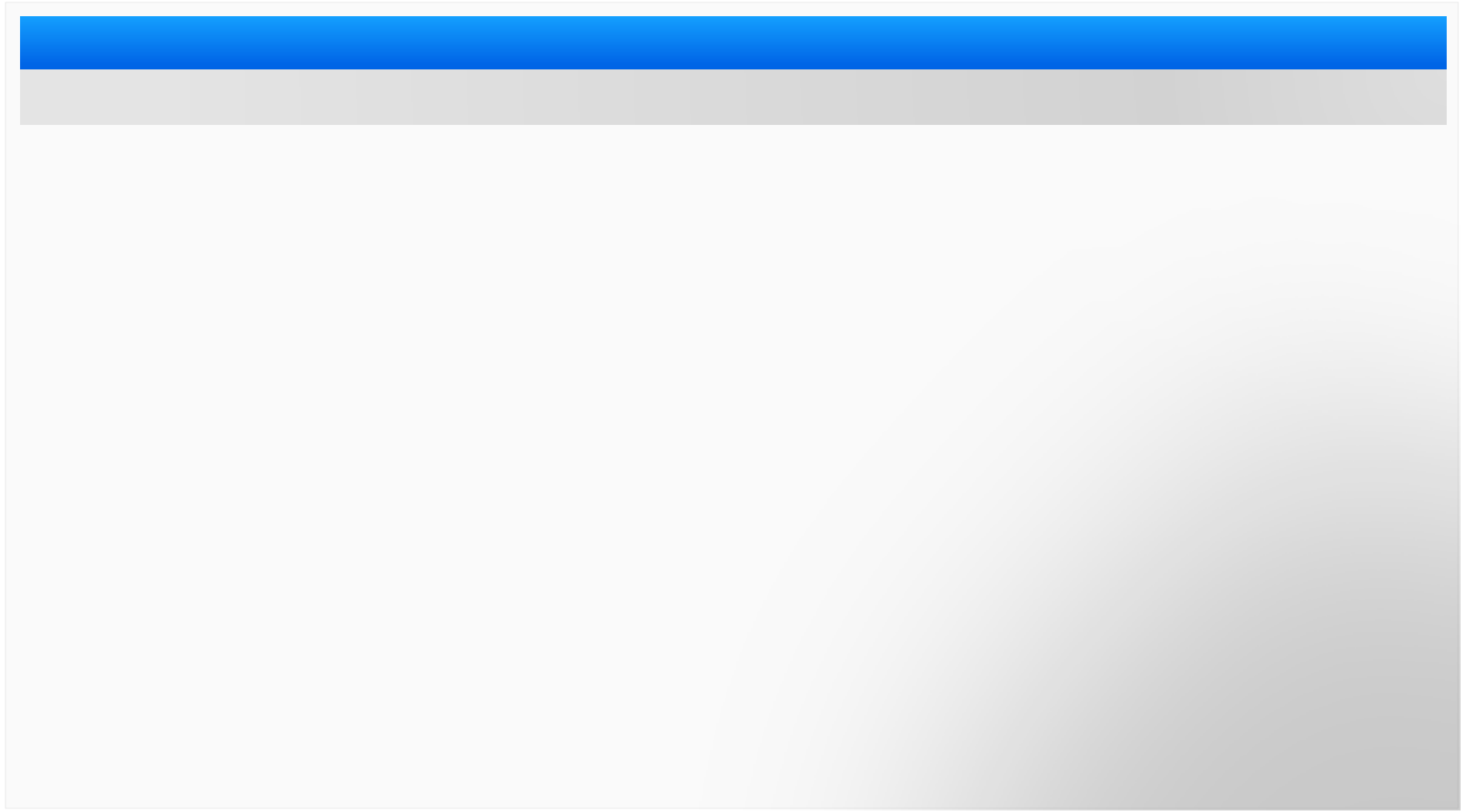
객체의 종류, class 종류

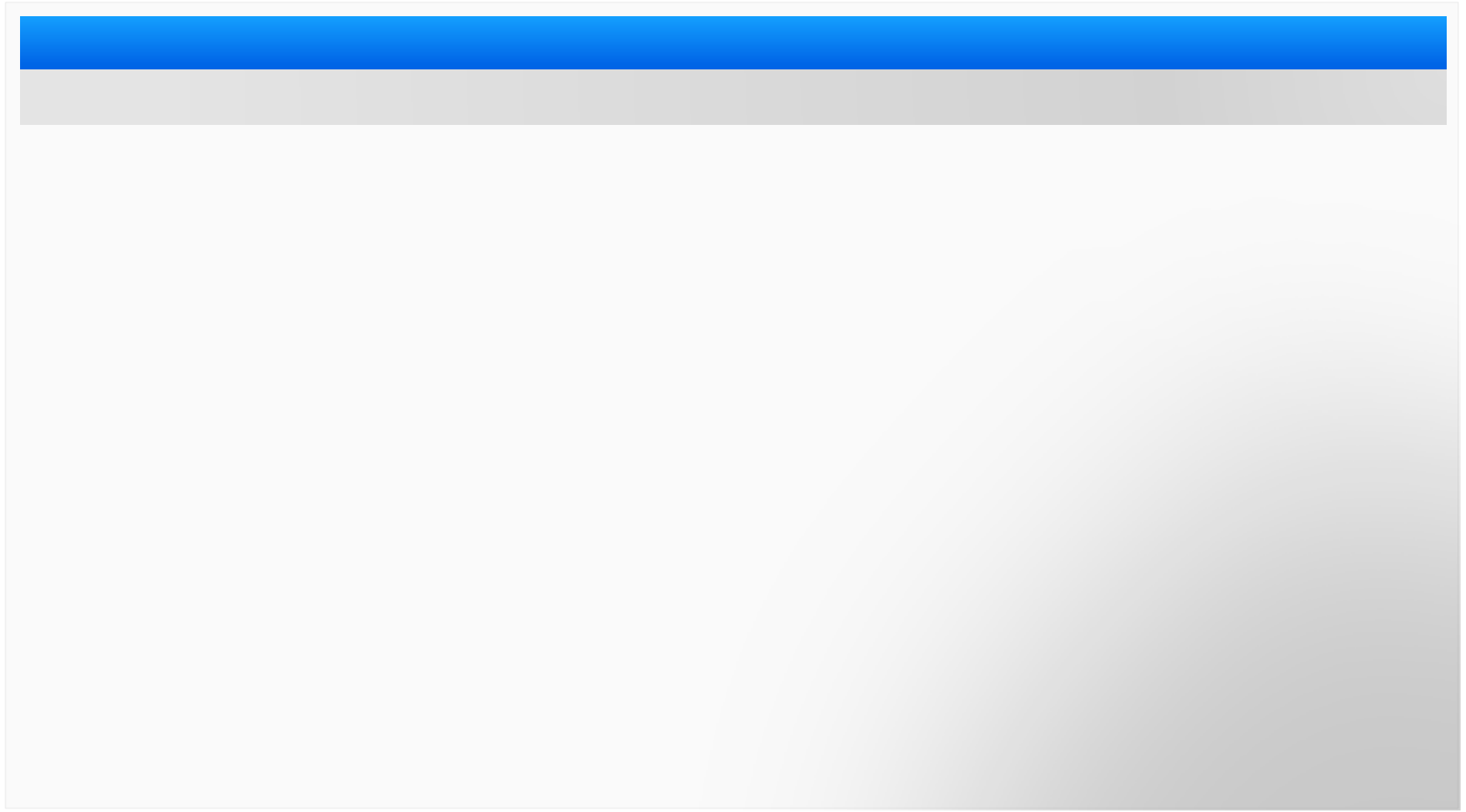
생성자 호출

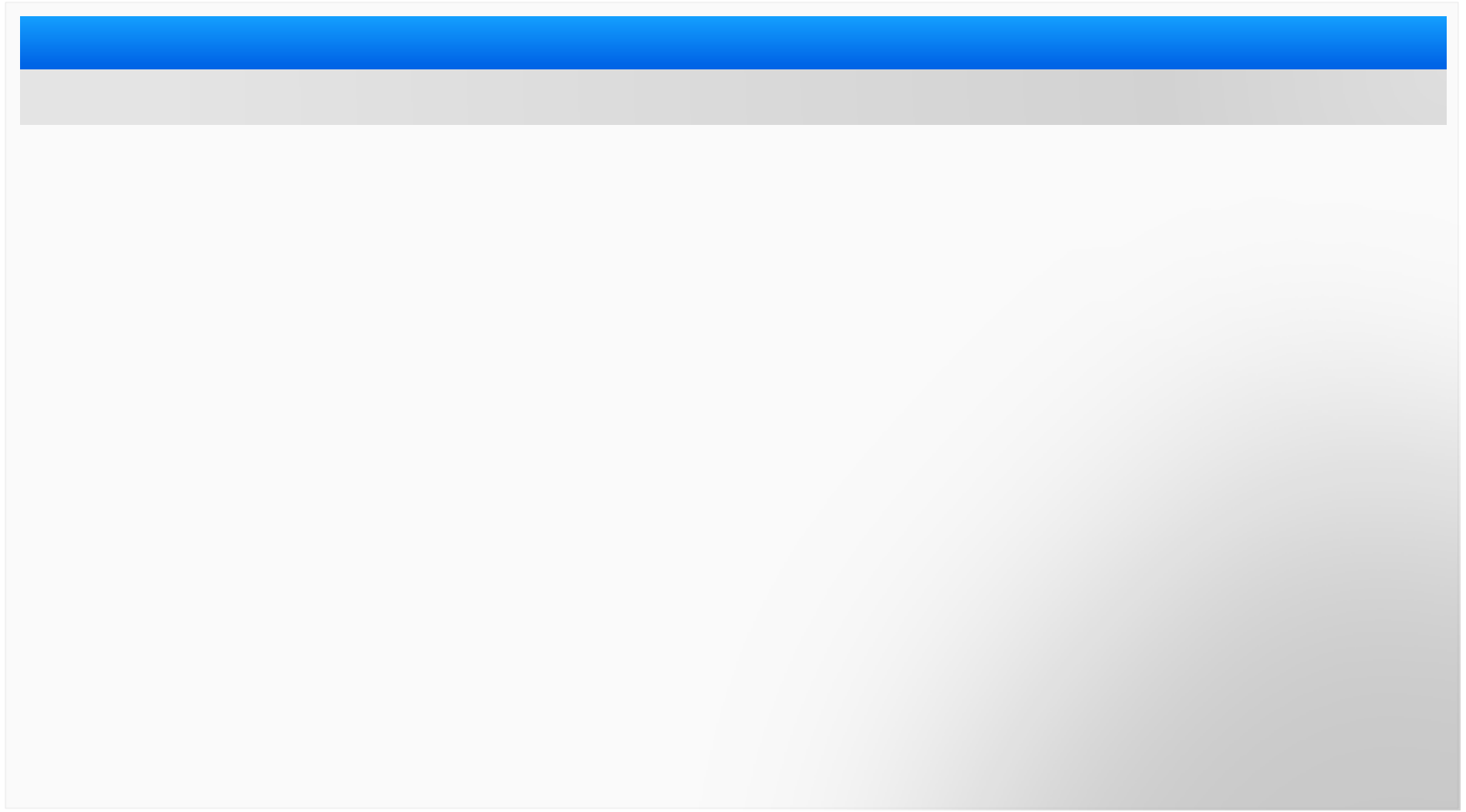
클래스 및 객체 생성 및 활용 예시

```
public class Dog {

        String name;
         int age;

        void bark()
        {
                if (age < 2)
                  System.out.println("낑낑"
                else
                  System.out.println("멍멍"
        }

}
```

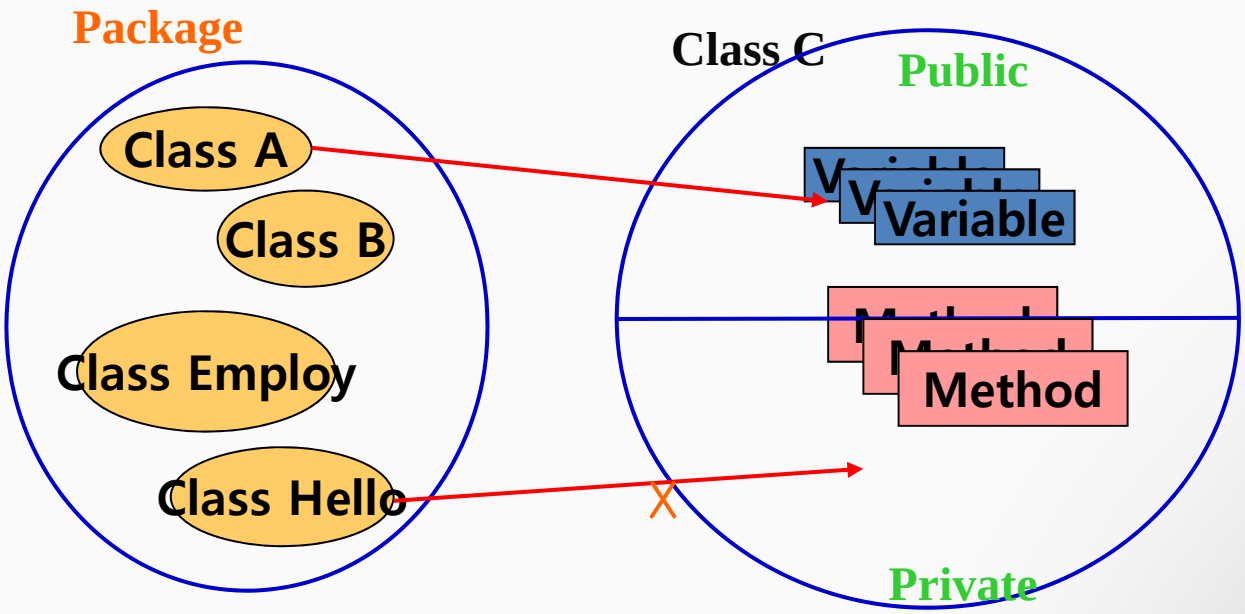
```
public class TestDog {
        public static void main(String[]
        args)
        {
                Dog d1 = new Dog();
                d1.name = "누렁이";
                d1.age  = 1;
                d1.bark();

                Dog d2 = new Dog();
                d2.name="발발이";
                ….
        }
}
```







접근자


Package
Class A
Class B
Class Employ
Class Hello
Class C
Public
Variable
Method
Private

접근자 사용 예시

```
public class Dog {
        private String name;
       private int age;

        void setName(String tempName)
        {
                name = tempName;
        }
        void setAge(int age)
        {       this.age = age;
        }
        void bark()
        {
                if (age < 2)
                  System.out.println(“낑낑”
                else
                  System.out.println(“멍멍”
        }
}
```

```
public class TestDog {
        public static void main(String[] args)
        {
                Dog d1 = new Dog();
                d1.setName( “누렁이”);
                //d1.name = “누렁이”

                d1.setAge(1); // d1.age=1
                d1.bark();

                Dog d2 = new Dog();
                ….
        }
}
```

생성자(constructor)를 활용한 객체 생성

```
public class Dog {
        private String name;
      private int age;

        public Dog(String name, int a
        {
                this.name = name;
                this.age = age;
        }
        void setName(String tempNa
      {         name = tempName;
        }
      void setAge(int age)
        {       this.age = age;
        }
        void bark()
        {
                System.out.println("멍
        }
}
```

```
public class TestDog {
        public static void main(String[] args)
        {
                Dog d1 = new Dog("누렁이", 1);
               d1.bark();

              Dog d2 = new Dog();
                ....
        }
}
```

Ex 1) 자동차 클래스

Class Definition

Class Name: Automobile

Variable:
- amount of fuel
- speed
- license plate

Methods:
- increaseSpeed
- stop

객체(Object)

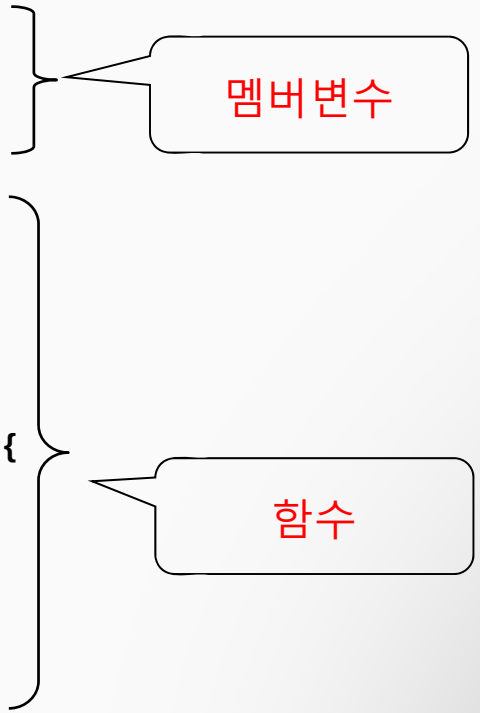
amount of fuel : 10
speed: 155
license plate: 135XJK

amount of fuel : 8
speed: 125
license plate: 121XLG

amount of fuel : 20
speed: 145
license plate: 221SPG

Ex 1) 자동차 클래스

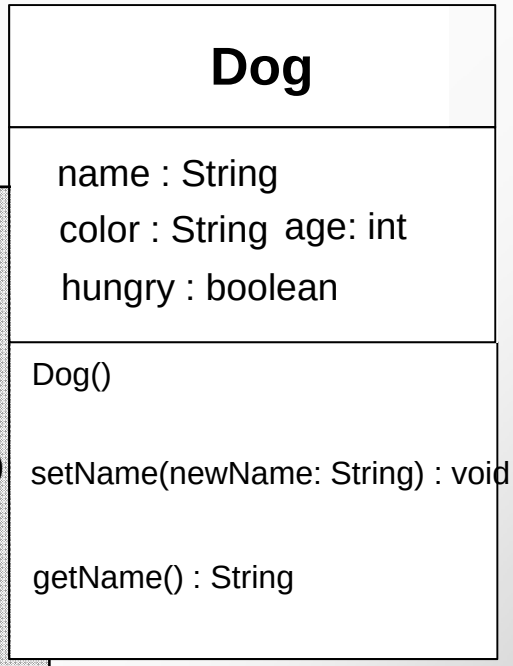
```
public class Automobile {

        private int amountOfFuel;
        private double speed;
        private String licensePlate;

        public Automobile() {
                amountOfFuel = 0;
                speed = 0.0;
                licensePlate = “”;
        }
        public void increaseSpeed(int increment ) {
                speed = speed + increment;
        }
        public void stop( int decrement ) {
                speed = speed – decrement;
        }
}
```



Ex 2) 개 (Dog) 클래스

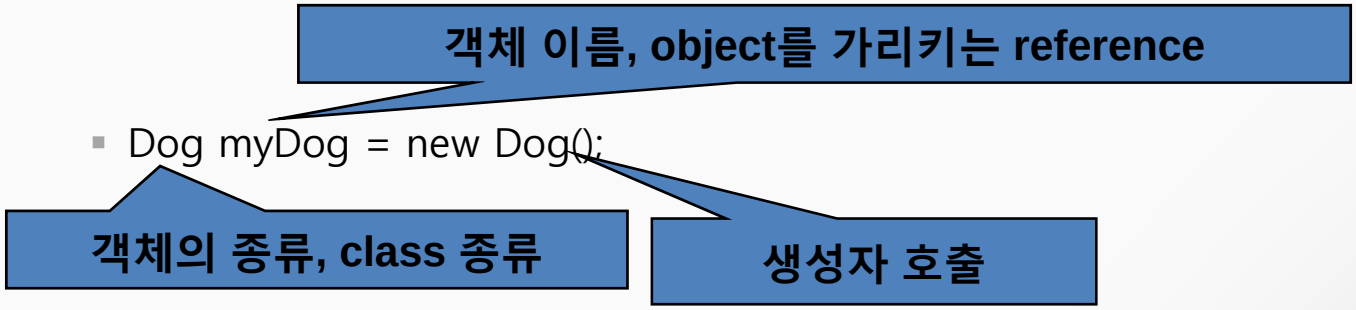
❖ **클래스 정의**

```
public class Dog {
   private String name;
   private int age;
   public String color;
   boolean hungry;
   public void Dog() {
        name = "";
        age = 0;
        color = "";
        hungry = false;
   }
   public void setName(String newName) {
        name = newName;
   }
    public String getName(){
        return name;
    }
}
```



객체 생성 정리

❖ **객체 생성 : use *new* keyword**

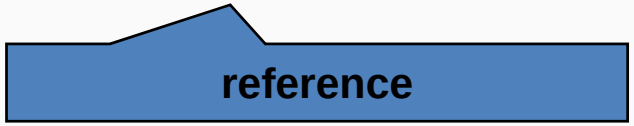


❖ **객체를 하나 생성을 하면..**

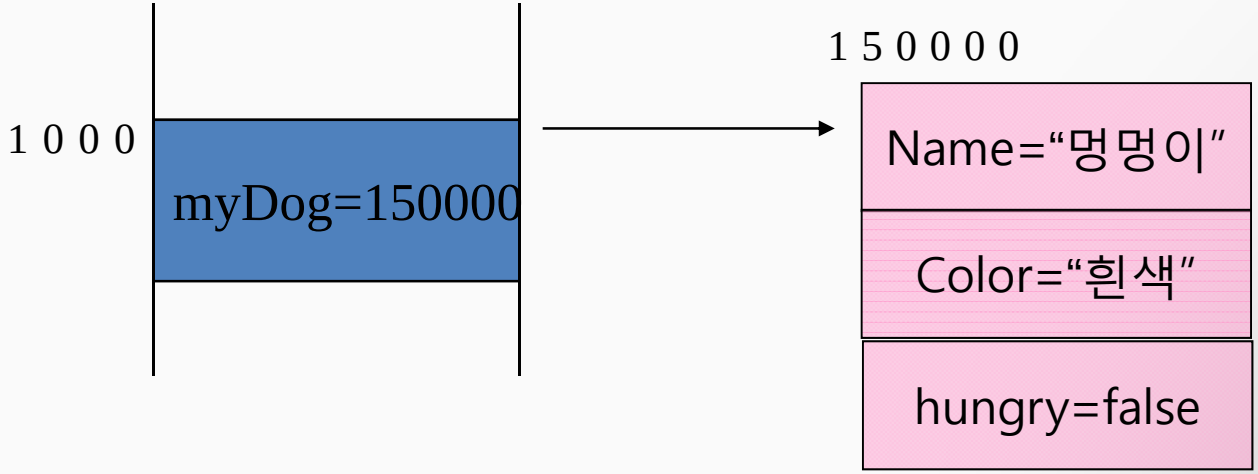
- 메모리에 그 객체를 위한 공간 할당
  - 멤버 변수와 같은 것을 위한 공간 할당
- 멤버 변수의 디폴트(default) 값으로 초기화
- 생성자 안의 사용자가 정의한 명령문 수행

객체 생성 정리

❖ **Dog myDog = new Dog();**




```
public void Dog() {
        name = "멍멍이";
        color = "흰색";
        hungry = false;
}
```



생성된 객체의 사용

❖ **클래스를 사용하는 것이 아니고 객체를 사용한다!**

❖ **객체 사용**

- 객체는 그 객체를 가리키는 변수 (reference)를 사용
- .(점)을 이용해서 객체 멤버에 접근
  - *name_of_object.name_of_member*
    - System.out.println(myDog.name);
    - myDog.name = "복실이";
    - System.out.println(myDog.getName());
    - myDog.setName("복실이");

간단한 객체 생성 사용 예

```
public class Dog2 {
  private String name;
  public String color;
  boolean hungry;
  public void Dog(String n, String c, boolean h) {
      name = n; color = c; hungry = h;
  }
  public void setName(String newName){  name = newName;  }
  public String getName(){ return name; }
  public void setColor(String newColor){ color = newColor; }
  public String getColor(){ return color; }

  public static void main(String[] args) {
      Dog dog_1 = new Dog("멍멍이", "검은색", false);
      System.out.println(dog_1.getName());
      dog_1.setName("깜둥이");
      System.out.println(dog_1.getName());

      Dog dog-2 = new Dog("복실이", "흰색", false);
      System.out.println(dog_2.getName());
      dog_2.setName(dog_1.getName());
      System.out.println(dog_2.getName());
  }
}
```

OUTPUT

```
멍멍이
깜둥이
복실이
깜둥이
```

용어 정리

- 객체(object)는 특정 클래스(class)의 하나의 인스턴스(*instance*) 이다.
- 클래스(class)의 멤버(member)로는 데이터 및 상태를 가지는 멤버 변수(member variable)와 행위 또는 일을 하는 함수 (method)가 있다
- 멤버 변수는 필드(*field) 또는 인스턴스 변수(instance variable)라고도 불린다.*
- *객체를 가르키는 변수를 그 객체의 참조변수 (reference variable, reference)라고 한다.*

“Learn by studying examples”
“Learn by hand programming”