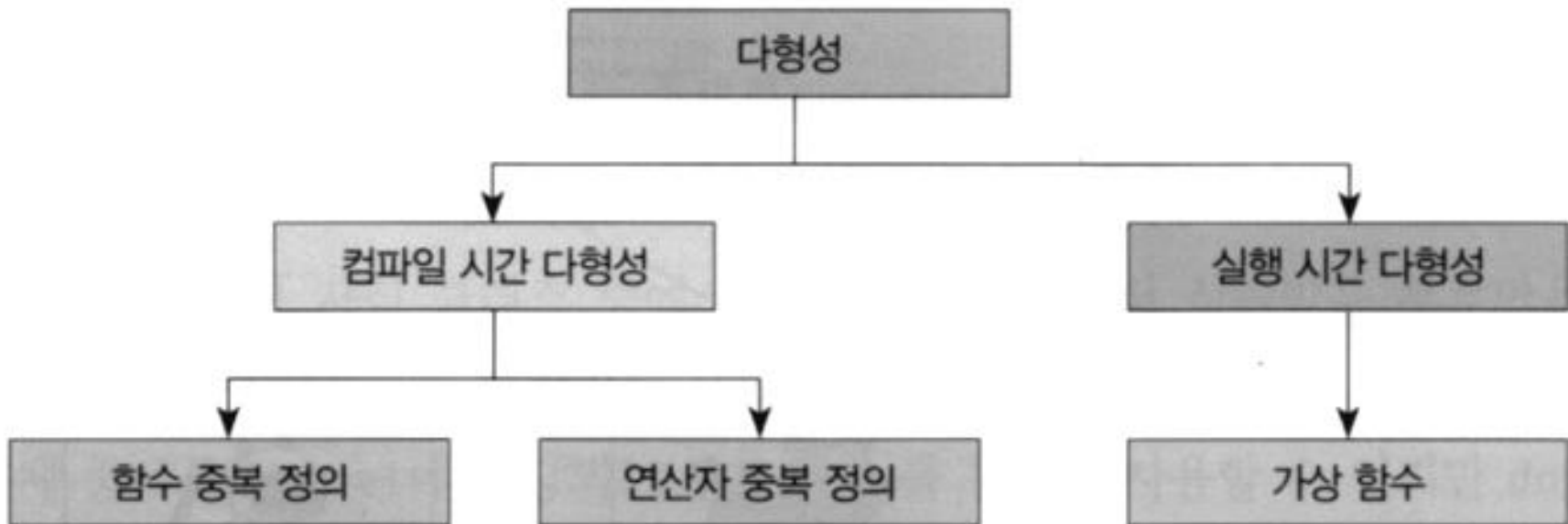


**다형성**

# 다형성

## ■ 다형성

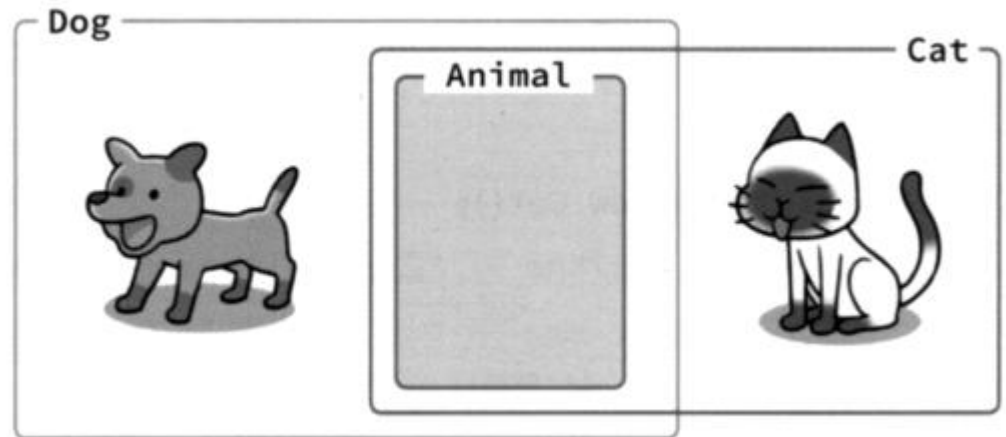
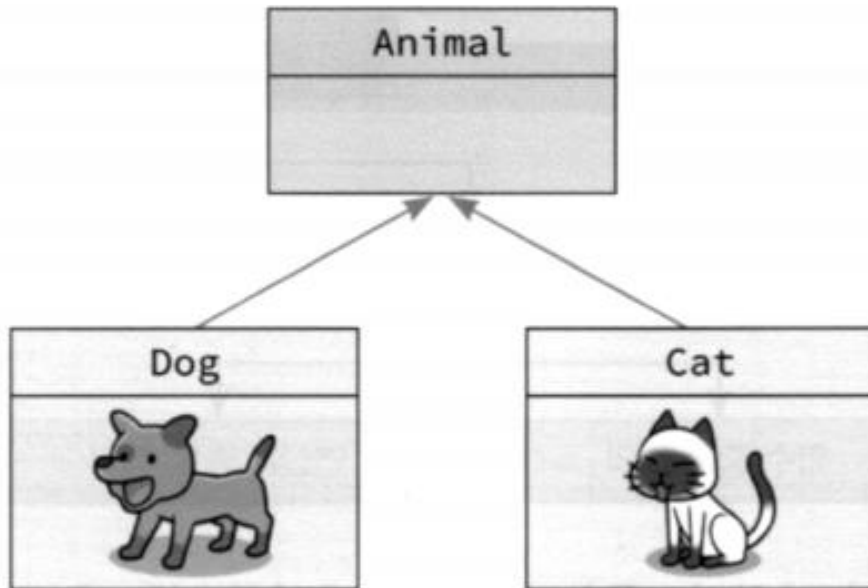
- 형태는 동일하나 다른 기능을 수행하는 것



# 다형성

## ■ 상향 형변환

○ `Animal *p = new Dog();`



## ■ chapter10/ex01\_upcasting.cpp] 상향 형변환

```
#include <iostream>
#include <string>
using namespace std;

class Animal {
public:
    void speak() {
        cout << "Animal speak()" << endl;
    }
};

class Dog : public Animal {
public:
    int age;
    void speak() {
        cout << "멍멍" << endl;
    }
};
```

# 다형성

## ■ chapter10/ex01\_upcasting.cpp] 상향 형변환

```
class Cat : public Animal {
public:
    void speak() {
        cout << "야옹" << endl;
    }
};

int main(int argc, char const *argv[])
{
    Animal *pa1 = new Dog();
    pa1->speak();
    // pa1->age = 20;    // 에러

    Animal *pa2 = new Cat();
    pa2->speak();
    return 0;
}
```

```
Animal speak()
Animal speak()
```

# 다형성

---

## ■ chapter10/ex01\_upcasting.cpp] 가상 함수 virtual

```
class Animal {  
public:  
    virtual void speak() {  
        cout << "Animal speak()" << endl;  
    }  
};
```

---

멍멍

야옹

---

# 순수 가상 함수

---

## ■ 순수 가상 함수

- virtual 함수인데 함수 정의를 가지지 않는 것
- virtual 반환형 함수명(매개변수 목록)=0;
- 순수 가상 함수를 가지는 클래스는 인스턴스화 불가
- 추상 클래스
  - 순수 가상 함수를 가지는 클래스
- 자식 클래스는 부모 클래스의 순수 가상 함수를 반드시 정의해야 함
- 정의하지 않으면 자식 클래스도 추상 클래스가 됨 --> 인스턴스화 불가

# 다형성

## ■ chapter10/ex02\_virtual.cpp] 순수 가상 함수

```
#include <iostream>
#include <string>
using namespace std;

class Shape {
    int x, y;

public:
    Shape(int xloc, int yloc): x(xloc), y(yloc) {    }
    virtual void draw() = 0; // 순수 가상 함수
};

class Rectangle: public Shape {
    int width, height;

public:
    Rectangle(int x,int y, int w, int h): Shape(x, y), width(w), height(h) { }

    void draw() {
        cout << "Rectangle Draw" << endl;
    }
};
```



# 다형성

---

## ■ chapter10/ex02\_virtual.cpp] 순수 가상 함수

```
int main(int argc, char const *argv[])
{
    Shape *ps = new Rectangle(0, 0, 100, 100);

    ps->draw();

    return 0;
}
```

---

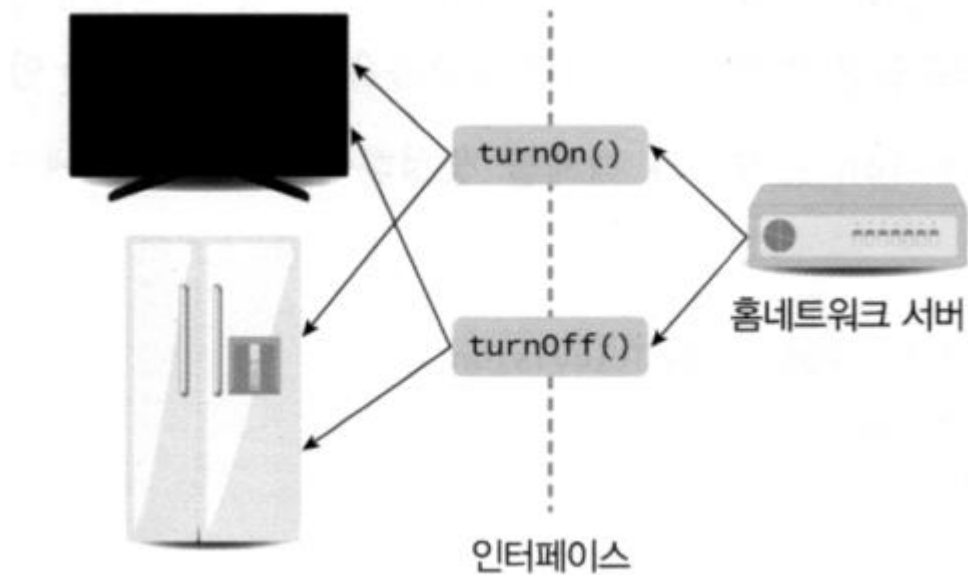
Rectangle Draw

---

# 인터페이스

## ■ 인터페이스

- 객체 타입의 모양만을 정의
  - 순수 가상 함수로만 구성
- 인터페이스 타입 포인터로 자식 클래스의 인스턴스들을 포인트 가능
  - 구체적인 자식 클래스의 타입을 알 필요 없음
  - 인터페이스에서 규정한 멤버 함수 호출 가능



# 인터페이스

---

## ■ 인터페이스

```
class RemoteControl {  
    // 순수 가상 함수 정의  
    virtual void turnON() = 0;           // 가전 제품을 켜다.  
    virtual void turnOFF() = 0;         // 가전 제품을 끈다.  
}
```

# 인터페이스

---

## ■ 인터페이스 구현

- 부모 클래스의 순수 함수를 정의하는것

```
class Television : public RemoteControl {  
    void turnON()  
    {  
        // 실제로 TV의 전원을 켜기 위한 코드가 들어 간다.  
        ...  
    }  
    void turnOFF()  
    {  
        // 실제로 TV의 전원을 끄기 위한 코드가 들어 간다.  
        ...  
    }  
}
```

# 다형성

## ■ 인터페이스 운영

```
RemoteControl *pt1 = new Television();  
pt1->turnOn();  
pt1->turnOff();  
  
RemoteControl *pt2 = new Refrigerator();  
pt2->turnOn();  
pt2->turnOff();
```

- --> 함수의 매개변수 타입으로 인터페이스에 대한 포인터 운영
  - 인터페이스 구현 객체이면 어떤 객체든 함수의 인자로 전달 가능