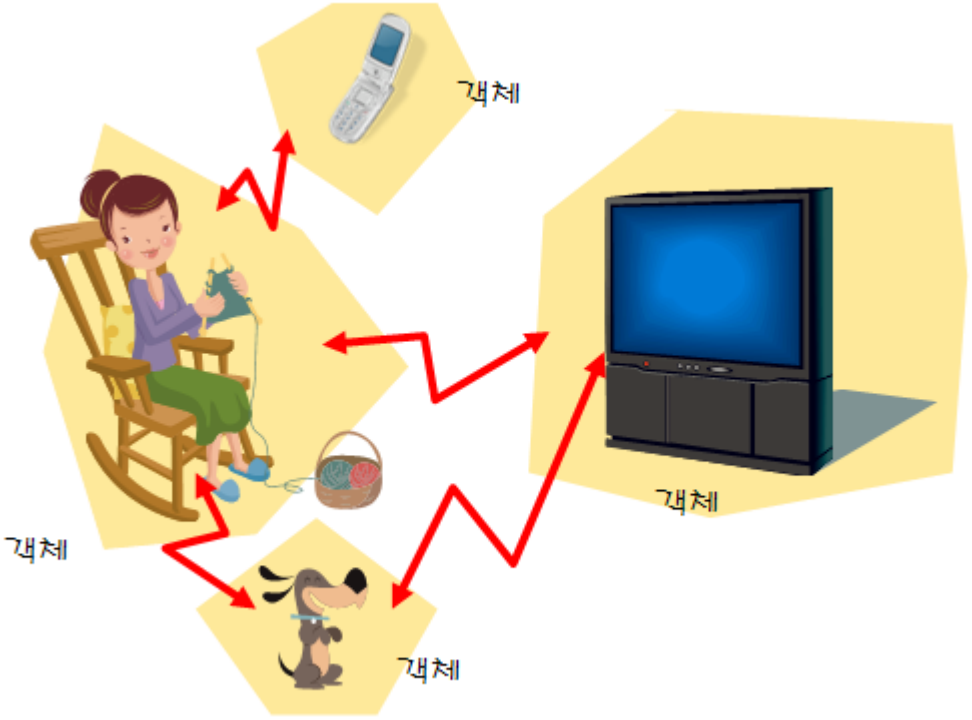
C++ Espresso

제9-2장 가상함수



이번 장에서 학습할 내용

- 가상 함수
- 순수 가상 함수

다형성은
객체들이 동일한
메시지에 대하여
서로 다르게
동작하는 것
입니다.

가상 함수

- 단순히 자식 클래스 객체를 부모 클래스 객체로 취급하는 것이 어디에 쓸모가 있을까?
- 다음과 같은 상속 계층도를 가정하여 보자.

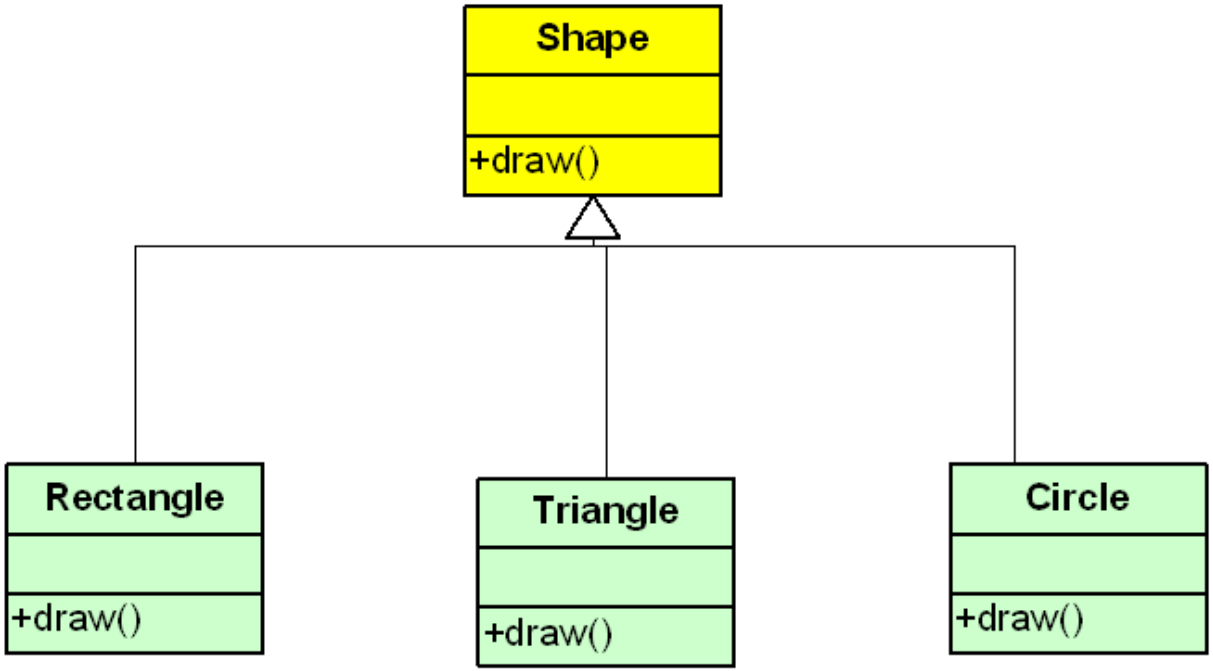


그림 14.6 도형의 UML

가상 함수

```
class Shape {
protected:
    int x, y;
public:
    void setOrigin(int x, int y) {
        this -> x = x;
        this -> y = y;
    }
     void draw() {
        cout << "Shape Draw" << endl;
    }
};
class Rectangle : public Shape {
    int width, height;
public:
    void setWidth(int w) { width = w; }
    void setHeight(int h) { height = h; }
    void draw() {
        cout << "Rectangle Draw" << endl;
    }
};
```

```
class Circle :public Shape {
    int radius;
public:
    void setRadius(int r) { radius = r; }
    void draw() {
        cout << "Circle Draw" << endl;
    }
};

void main()
{
    Shape *ps = new Rectangle;
    ps->draw();
    delete ps;

    Shape *ps1 = new Circle;
    ps1->draw();
    delete ps1;
}
```

Shape 포인터이기 때문에 Shape의 draw()가 호출

```
Shape Draw
Shape Draw
```

가상 함수

- 만약 Shape 포인터를 통하여 멤버 함수를 호출하더라도 도형의 종류에 따라서 서로 다른 draw()가 호출된다면 상당히 유용할 것이다.

- 즉 사각형인 경우에는 사각형을 그리는 draw()가 호출되고 원의 경우에는 원을 그리는 draw()가 호출된다면 좋을 것이다.

  -> draw()를 **가상 함수**로 작성하면 가능

가상 함수

```
class Shape {
protected:
    int x, y;
public:
    void setOrigin(int x, int y) {
        this -> x = x;
        this -> y = y;
    }
    virtual void draw() {
        cout << "Shape Draw" << endl;
    }
};
class Rectangle : public Shape {
    int width, height;
public:
    void setWidth(int w) { width = w; }
    void setHeight(int h) { height = h; }
    void draw() {
        cout << "Rectangle Draw" << endl;
    }
};
```

가상함수 정의!

재정의

```
class Circle :public Shape {
    int radius;
public:
    void setRadius(int r) { radius = r; }
    void draw() {
        cout << "Circle Draw" << endl;
    }
};

void main()
{
    Shape *ps = new Rectangle;
    ps->draw();
    delete ps;

    Shape *ps1 = new Circle;
    ps1->draw();
    delete ps1;
}
```

재정의

```
Rectangle Draw
Circle Draw
```

실습예제

1. 이전에 작성했던 Shape 프로그램에 Triangle 클래스를 추가해보자.

   - Shape 클래스를 public으로 상속받는다.
   - 멤버 변수 : private 형식의 정수형 변수 base, height
   - 멤버 함수 : public 형식으로, 부모클래스의 draw()를 재정의하여 "Triangle Draw"를 출력한다.

```
class Triangle :public Shape {
private:
    int base, height;
public:
    void draw() {
        cout << "Triangle Draw" << endl;
    }
};
```

실습예제

2. 이전에 작성했던 프로그램의 main함수를 변경해보자

```
void main()
{
    Shape *arrayOfShapes[3];

    arrayOfShapes[0] = new Rectangle();
    arrayOfShapes[1] = new Triangle();
    arrayOfShapes[2] = new Circle();

    for (int i = 0; i < 3; i++)
        arrayOfShapes[i]->draw();
}
```

이것이 가능할까?

Shape ps = new Rectangle;

```
Shape ps = new Rectangle;
```

```
void *__cdecl operator new(size_t _Size)

+ 3개 오버로드

"Rectangle *"에서 "Shape"(으)로 변환하기 위한 적절한 생성자가 없습니다.
```

-> 업캐스팅을 하기 위해서는 반드시 객체포인터를 사용한다!

```
Shape *ps = new Rectangle;
```

예제

실습

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

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

class Dog : public Animal
{
public:
  Dog() { cout <<"Dog 생성자" << endl; }
  ~Dog() { cout <<"Dog 소멸자" << endl; }
  void speak() { cout <<"멍멍" << endl; }
};
```

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

int main()
{
  Animal *a1 = new Dog();
  a1->speak();

  Animal *a2 = new Cat();
  a2->speak();
  return 0;
}
```

```
Animal 생성자
Dog 생성자
멍멍
Animal 소멸자
Animal 생성자
Cat 생성자
야옹
Animal 소멸자
```

순수 가상 함수

- 순수 가상 함수(pure virtual function): 함수 헤더만 존재하고 함수의 몸체는 없는 함수

```
virtual    반환형      함수이름(매개변수  리스트) = 0;
```

- (예) virtual void draw() = 0;

- 추상 클래스(abstract class): 순수 가상 함수를 하나라도 가지고 있는 클래스

순수 가상 함수의 예

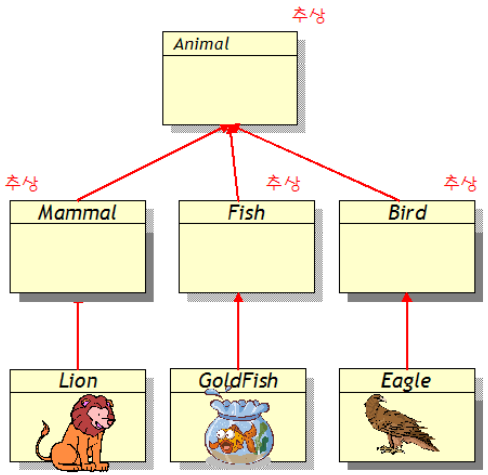
```
class Shape {
protected:
        int x, y;

public:
        ...
        virtual void draw() = 0;
};

class Rectangle : public Shape {
private:
        int width, height;

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

```
int main()
{

  Shape *ps = new Rectangle(); // OK!
  ps->draw();              // Rectangle의 draw()가 호출된다.
  delete ps;

  return 0;
}
```

```
Rectangle Draw
```

추상 클래스

- 추상 클래스(abstract class): 순수 가상 함수를 가지고 있는 클래스
- 추상 클래스는 추상적인 개념을 표현하는데 적당하다.
- 객체를 생성하지 못하며 순수 가상 함수는 클래스 상속에 의해 파생 클래스에서 함수 오버라이딩되어 사용되는 것을 목적으로 한다.



예제

```
class Animal {
        virtual void move() = 0;
        virtual void eat() = 0;
        virtual void speak() = 0;
};

class Lion : public Animal {
        void move(){
                cout << "사자의 move() << endl;
        }
        void eat(){
                cout << "사자의 eat() << endl;
        }
        void speak(){
                cout << "사자의 speak() << endl;
        }
};
```

Q & A