객체프로그래밍 정리노트

202204103 ict융합공학부 이승훈
202204023 ict융합공학부 김보민

[상속]
-기능이 중복 된 클래스들이 있다면 상속 관계로 클래스를 간결화 할 수 있다

[예시]
class Student
-말하기, 먹기, 걷기, 잠자기, **공부하기**

class StudentWorker
-말하기, 먹기, 걷기, 잠자기, **공부하기, 일하기**

class Researcher
-말하기, 먹기, 걷기, 잠자기, **연구하기**

class Professer
-말하기, 먹기, 걷기, 잠자기, **연구하기, 가르치기**

이런 경우, 공통 기능(말하기, 먹기, 걷기, 잠자기)을 class Person으로 작성

class student에 공부하기 추가, class StudentWorker가 상속 받고 일하기 추가

class Researcher에 연구하기 추가, class Professor가 상속 받고 가르치기 추가

[상속 선언]

```
class Student:public Person{
 .......
}
```
-Student 클래스는 Person 클래스의 멤버를 물려받는다

```
class StudentWorker:public Student{
..............
}
```
-StudentWorker 클래스는 Student의 멤버를 물려받는다

[예제 8-1]

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

class Point {
        int x, y;
public:
        void set(int x, int y) {
                this->x = x; this->y = y;
        }
        void showPoint() {
                cout << "(" << x << "," << y << ")" << endl;
        }
};
class ColorPoint :public Point {
        string color;
public:
        void setColor(string color) { this->color = color; }
        void showColorPoint();

};
void ColorPoint::showColorPoint() {
        cout << color << ":";
        showPoint();
}
int main() {
        Point p;
        ColorPoint cp;
        cp.set(3, 4);
        cp.setColor("Red");
        cp.showColorPoint();
}
```

-colorPoint 클래스에 Point클래스를 상속 받고 colorPoint 자신만의 속성과 기능을 추가해준다

[업 캐스팅/다운 캐스팅]

업 캐스팅 -> 부모 클래스의 포인터가 자식 클래스를 가리키는 것
다운 캐스팅 -> 자식 클래스의 포인터가 부모 클래스를 가리키는 것

[차이점]
업 캐스팅은 기본 클래스에 정의된 멤버만 호출 가능하다, 따라서 파생 클래스의 고유의 기능을 사용 불가능
ㄴ 해결 방법 -> 다운 캐스팅을 해서 기본 클래스 포인터가 파생클래스의 객체를 가리키는 형태에서 반대로 파생 클래스의 포인터가 파생 클래스의 객체 형태로 원래 타입으로 변환시켜줌
다운 캐스팅은 동일한 타입의 포인터가 동일한 타입을 가리키는 것
다운 캐스팅은 업 캐스팅과 달리 명시적으로 타입변환을 지정

[예제 8-2]

```
#include <iostream>
#include <string>
using namespace std;
class Point {
protected:
	int x, y;
public:
	void set(int x, int y);
	void showPoint();
};
void Point::set(int x, int y) {
	this->x = x;
	this->y = y;
}
void Point::showPoint() {
	cout << "(" << x << "," << y << ")" << endl;
}
class ColorPoint : public Point {
	string color;
public:
	void setColor(string color);
	void showColorPoint();
	bool equals(ColorPoint p);
};
void ColorPoint::setColor(string color) {
	this->color = color;
}
void ColorPoint::showColorPoint() {
	cout << color << ":";
```

```
        showPoint();
}
bool ColorPoint::equals(ColorPoint p) {
        if (x == p.x && y == p.y && color == p.color) // ①
                return true;
        else
                return false;
}
int main() {
        Point p;
        p.set(2, 3); // ②
        p.x = 5; // ③
        p.y = 5; // ④
        p.showPoint();
        ColorPoint cp;
        cp.x = 10; // ⑤
        cp.y = 10; // ⑥
        cp.set(3, 4);
        cp.setColor("Red");
        cp.showColorPoint();
        ColorPoint cp2;
        cp2.set(3, 4);
        cp2.setColor("Red");
        cout << ((cp.equals(cp2)) ? "true" : "false"); // ⑦
}

// 어떤게 오류날지 먼저 찾아보자
-> 접근 지정자를 신경써서 보는 것이 중요한데 2345번이 오류날 것 같아.
// 맞아 나도 그렇게 생각해 x,y는 protected로 선언되어 있어서 자신의 클래스와 상속 받은 클래스에서만 호출이 가능해. 그래서 오류가 나는 것 같아
```

[예제 8-3]
-출력 값 예상하기
//보민 , ->승훈

//파생된 클래스의 매개변수를 보면 size=32이고, ip=192.0.0.1로 나올 것 같아
->응 나도 그렇게 생각해 근데 저 boolalpha의 역할을 모르겠어서 videoln의 출력 값을 예상 못 하겠어
//나도 잘 모르겠어. 검색해볼게. boolalpha는 불린 값을 true. false로 출력되게 하는 조작자 역할을 한다고 나오네.
->그러면 출력값은
size=32
videoln=true
IP=192.0.0.1이겠네

[상속 지정]
-상속 선언 시 public, private, protected의 3가지 중 하나 지정

public-기본 클래스의 protected, public 멤버 속성을 그대로 계승

private-기본 클래스의 protected, public 멤버 속성을 private로 계승

protected-기본 클래스의 protected, public 멤버 속성을 protected로 계승

[예제 8-5]
-컴파일 오류 찾기

```
#include <iostream>
using namespace std;
class Base {
        int a;
protected:
        void setA(int a) { this->a = a; }
public:
        void showA() { cout << a; }
};
class Derived : protected Base {
        int b;
protected:
        void setB(int b) { this->b = b; }
```

```
public:
        void showB() { cout << b; }
};
int main() {
        Derived x;
        x.a = 5; // ①
        x.setA(10); // ②
        x.showA(); // ③
        x.b = 10; // ④
        x.setB(10); // ⑤
        x.showB(); // ⑥
}
```

```
//보민 , ->승훈
```

-> 나는 3번이랑 6번 빼고는 다 컴파일 오류가 발생할 것 같아
// 나는 6번 빼고 다 오류가 발생할 거라고 예상해. 3번의 showA() 클래스는 Base 클래스의 public 멤버이지만 Derived 클래스는 Base 클래스를 protected로 상속 받았기 때문에 Derived 클래스 외부에서 showA() 메소드를 호출할 수 없어.
-> 아 그럼 내가 2번도 잘못 이해하고 있던 거구나. 그럼 2번이 오류 나는 이유 또한 Base 클래스의 protected 멤버인데 protected 멤버는 해당 클래스와 하위 클래스 내에서만 접근 가능한데 main에서 호출했기에 오류가 난 거 맞지 ?
// 맞아 그렇게 보면 돼.

[다중 상속]
[예제 8-7]
Adder, Subtractor를 다중 상속받는 calculator를 작성하려면
class Calculator:pubic Adder,public Subtractor{
public:...};로 작성

```
#include <iostream>
using namespace std;
class Adder {
protected:
        int add(int a, int b) { return a + b; }
};
class Subtractor {
protected:
        int minus(int a, int b) { return a - b; }
};
```

```
class Calculator : public Adder, public Subtractor {
public:
	int calc(char op, int a, int b);
};
int Calculator::calc(char op, int a, int b) {
	int res = 0;
	switch (op) {
	case '+': res = add(a, b); break;
	case '-': res = minus(a, b); break;
	}
	return res;
}

int main() {
	Calculator handCalculator;
	cout << "2 + 4 = "
		<< handCalculator.calc('+', 2, 4) << endl;
	cout << "100 - 8 = "
		<< handCalculator.calc('-', 100, 8) << endl;
}
```