프로그래밍 언어론

Assignment #6

전기컴퓨터공학부 정보컴퓨터공학전공

201524582 정희석

1. Why do we need null virtual classes in C++? We could just refrain from allocating objects of this class and simply use subclasses of the superclass.

-> 순수가상클래스를 C++에서 사용해야 하는 이유? 이 클래스의 객체를 할당하는데 있어서 제한하고 간단히 슈퍼클래스의 서브클래스로 사용하는 것으로 대체할 수 있지 않은가.
–구체적인 C++ 예를 통해서 설명하시오.
(다른 참고문헌에서 예를 발췌한 경우에는 해당 참고문헌의 정보를 논문 기입 양식대로 작성하시오.)
A.
순수가상클래스(null virtual class)는 하나 이상의 순수 가상 함수를 가지는 클래스로 추상 클래스(abstract class)와 같다고 여겨지며 이는 실체가 없는 추상적인 클래스이다. 즉 객체를 가질 수 없는 개념적인 클래스이다. 즉 선언(Declaration)만 존재하고 구현(definition)이 없는 클래스이다. 인터페이스를 제공하고 실제 구현 및 동작은 유도(자식) 클래스에게 맡기는 클래스이다.
그리고 Virtual 은 Late Binding, Dynamic Binding 으로 사전에 (compile 단계에서)정의되지 않고 사용될 때 동작이 정의가 된다. 즉 Overriding(같은 이름의 함수, 같은 인자, 다른 동작)이 된다는 것이다. 하지만 일반적인 상속관계에 있는 클래스에서는 부모(슈퍼) 클래스에서 정의된 함수는 자식(서브) 클래스에서 재정의 할 수 없다. 이는 이미 부모(슈퍼)클래스에서 정의가 되어서 구현이 되어 있기 때문에 충돌이 발생하기 때문이다.
즉 슈퍼클래스와 서브클래스로 상속관계, 어떠한 제한 조건을 통해 무조건 구현을 해야 하는 함수를 넣을 수 있지만, 같은 이름에 같은 인자를 가지지만 각 서브 클래스마다 다른 동작을 하도록 함수를 Overriding 은 할 수 없기 때문에 null virtual function 을 가지는 null virtual class 를 사용해야 한다. (virtual class 를 사용하면 코드 재사용에 용이)

<코드> => 예시: 도형 Class => Super class: Shape, Sub classes: Rectangle, Triangle, Circle, …

```
#include <iostream>
#include <cmath>
Ver.1)
class Shape{
	public:
		virtual double getArea() = 0; //Null virtual function
};
class Rectangle: public Shape {
	public:
		Rectangle(int len, int height){
			this.len = len; this.height = height;
		}
		double getArea(){//@Overriding
			return len * height;
		}
	Private:
		Int len; int height;
};
class Triangle: public Shape {
	public:
		Triangle(int len, int height){
			this.len = len; this.height = height;
		}
		double getArea(){//@Overriding
			return (len * height)/2 ;
		}
	Private:
		Int len; int height;
};
class Circle: public Shape {
	public:
		Circle(int radius){
			this.radius = radius;
		}
		double getArea(){//@Overriding
			return radius*radius/M_PI;
		}
	private:
		int radius; };
```

Ver.2 ->엄밀히 말하면 완전 null virtual class는 아님 abstract class에 가까움)

```
class Shape{
	public:
		Shape();
		Shape(int len, int height){
			this.len = len;
			this.height = height;
		}
		virtual double getArea() = 0; //Null virtual function
	protected:
		int len; int height;
};
class Rectangle: public Shape {
	public:
		Rectangle(int len, int height):Shape(len,height){}
		double getArea(){//@Overriding
			return len * height;
		}
};
class Triangle: public Shape {
	public:
		Triangle(int len, int height):Shape(len,height){}
		double getArea(){//@Overriding
			return (len * height)/2 ;
		}
};
class Circle: public Shape {
	public:
		Circle(int radius){
			this.radius = radius;
		}
		double getArea(){//@Overriding
			return radius*radius/M_PI;
		}
	private:
		int radius;
};
```