

[lab13 보고서]

자율전공학과

2019111677

김지연

1. 소스코드

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

class GameCharacter{
public:
    virtual void draw() {}
};

class Hobbit : public GameCharacter{
    void draw() {
        cout << "호빗을 그립니다." << endl;
    }
};

class Titan : public GameCharacter{
    void draw() {
        cout << "타이탄을 그립니다." << endl;
    }
};

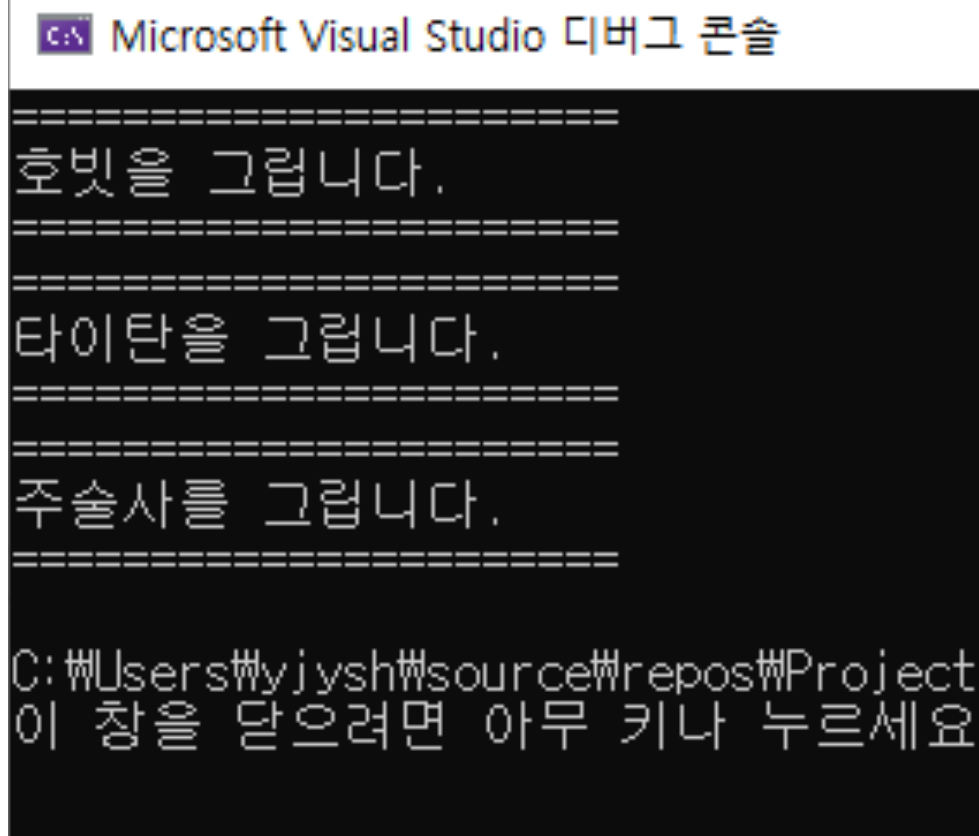
class Magician : public GameCharacter{
    void draw() {
        cout << "주술사를 그립니다." << endl;
    }
};

int main(void){
    GameCharacter* arrayOfGameCharacter[3];

    arrayOfGameCharacter[0] = new Hobbit();
    arrayOfGameCharacter[1] = new Titan();
    arrayOfGameCharacter[2] = new Magician();

    for (int i = 0; i < 3; i++) {
        cout << "=====" << endl;
        arrayOfGameCharacter[i]->draw();
        cout << "=====" << endl;
    }
    return 0;
}
```

2. 실행결과화면



```
C:\N Microsoft Visual Studio 디버그 콘솔

=====
호빗을 그립니다.
=====

=====
타이탄을 그립니다.
=====

=====
주술사를 그립니다.
=====

C:\Users\wyjysh\source\repos\Project
이 창을 닫으려면 아무 키나 누르세요
```

3. 문제 정의 및 분석

부모클래스 GameCharacter

자식클래스 Hobbit, Titan, Magician

부모클래스인 GameCharacter를 정의한다.

Draw()멤버함수를 가상함수로 정의한다. (객체 안에서 가상 함수로 정의하게 되면 우선순위가 높은 오버라이딩된 함수가 호출되기 때문에 부모객체의 draw함수가 아니라 자식클래스의 draw함수를 호출할 수 있게 된다.)

자식클래스의 Hobbit, Titan, Magician 을 GameCharacter클래스에서 상속받아 만든다.

클래스 GameCharacter에 대한 포인터 배열을 선언한다.

포인터를 사용하여 draw()함수를 호출한다. (업캐스팅-부모클래스가 자식클래스를 가리킴, 메세징) 동적바인딩에 의하여 서로 다른 draw()가 호출된다.

1. 소스코드

```
#include <iostream>
#include <string>
#define PI 3.141592
using namespace std;
class Shape {
protected:
    int width;
    int height;
public:
    // 여기에 도형 클래스의 생성자와 멤버함수 정의
    Shape(int w, int h) :width(w), height(h) {}
    virtual void printShape() {}
};

class TwoDimShape : public Shape {
private:
    int area;
public:
    // 여기에 2차원도형 클래스의 생성자와 멤버함수들 정의
    TwoDimShape(int w, int h) :Shape(w, h) {}
    virtual double getArea() { return area; }
    void printShape() {}
};

class ThreeDimShape : public Shape {
private:
    int volume;
public:
    // 여기에 3차원도형 클래스의 생성자와 멤버함수들 정의
    ThreeDimShape(int w, int h) :Shape(w, h) {}
    virtual double getVolume() { return volume; }
    void printShape() {}
};

class Rectangle : public TwoDimShape {
    // 사각형 클래스에 추가 멤버변수 없음!
public:
    // 여기에 사각형 클래스의 생성자와 멤버함수 정의
    Rectangle(int w, int h) :TwoDimShape(w, h) {}
    double getArea() {
        return width * height;
    }
    void printShape() {
        cout << "Rectangle 면적: " << getArea() << endl;
    }
};

class Ellipse : public TwoDimShape {
    // 타원 클래스에 추가 멤버변수 없음!
public:
    // 여기에 타원 클래스의 생성자와 멤버함수 정의
    double getArea() {
        return width * height * PI / 4;
    }
    Ellipse(int w, int h) :TwoDimShape(w, h) {}
    void printShape() {
```

```

        cout << "Ellipse 면적: " << getArea() << endl;
    }
};

class Triangle : public TwoDimShape {
    // 삼각형 클래스에 추가 멤버변수 없음!
public:
    // 여기에 삼각형 클래스의 생성자와 멤버함수 정의
    double getArea() {
        return width * height / 2;
    }
    Triangle(int w, int h) :TwoDimShape(w, h) {}
    void printShape() {
        cout << "Triangle 면적: " << getArea() << endl;
    }
};

class Sphere : public ThreeDimShape {
private:
    int radius;
public:
    // 여기에 구 클래스의 생성자와 멤버함수 정의
    double getVolume() {
        return ((double)3 / 4) * radius * radius * radius * PI;
    }
    Sphere(int w, int h, int r) :ThreeDimShape(w, h), radius(r) {}
    void setRadius(int r) { radius = r; }
    int getRadius() { return radius; }
    void printShape() {
        cout << "Sphere 체적: " << getVolume() << endl;
    }
};

class Cube : public ThreeDimShape {
private:
    int base;
public:
    // 여기에 육면체 클래스의 생성자와 멤버함수 정의
    double getVolume() {
        return width * height * base;
    }
    Cube(int w, int h, int b) :ThreeDimShape(w, h), base(b) {}
    void setBase(int b) { base = b; }
    int getBase() { return base; }
    void printShape() {
        cout << "Cube 체적: " << getVolume() << endl;
    }
};

class Cylinder : public ThreeDimShape {
private:
    int radius;
public:
    // 여기에 원통 클래스의 생성자와 멤버함수 정의
    double getVolume() {
        return radius * radius * PI * height;
    }
};

```

```

    }
    Cylinder(int w, int h, int r) :ThreeDimShape(w, h), radius(r) {}
    void setRadius(int r) { radius = r; }
    int getRadius() { return radius; }
    void printShape() {
        cout << "Cylinder 체적: " << getVolume() << endl;
    }
};

void printShape() {

}

int main(void) {
    // Ellipse(1,2), Rectangle(3,4), Triangle(5,6)
    TwoDimShape* t[3];
    t[0] = new Ellipse(1, 2);
    t[1] = new Rectangle(3, 4);
    t[2] = new Triangle(5, 6);

    // Sphere(7,8,9), Cube(10,11,12), Cylinder(13,14,15)
    ThreeDimShape* h[3];
    h[0] = new Sphere(7, 8, 9);
    h[1] = new Cube(10, 11, 12);
    h[2] = new Cylinder(13, 14, 15);
    // 위와 같은 6개의 객체를 생성해서 up-casting해야 아래의 반복문을 사용할 수 있음
    for (int i = 0; i < 3; i++) {
        t[i]->printShape();
        h[i]->printShape();
    }
    return 0;
}

```

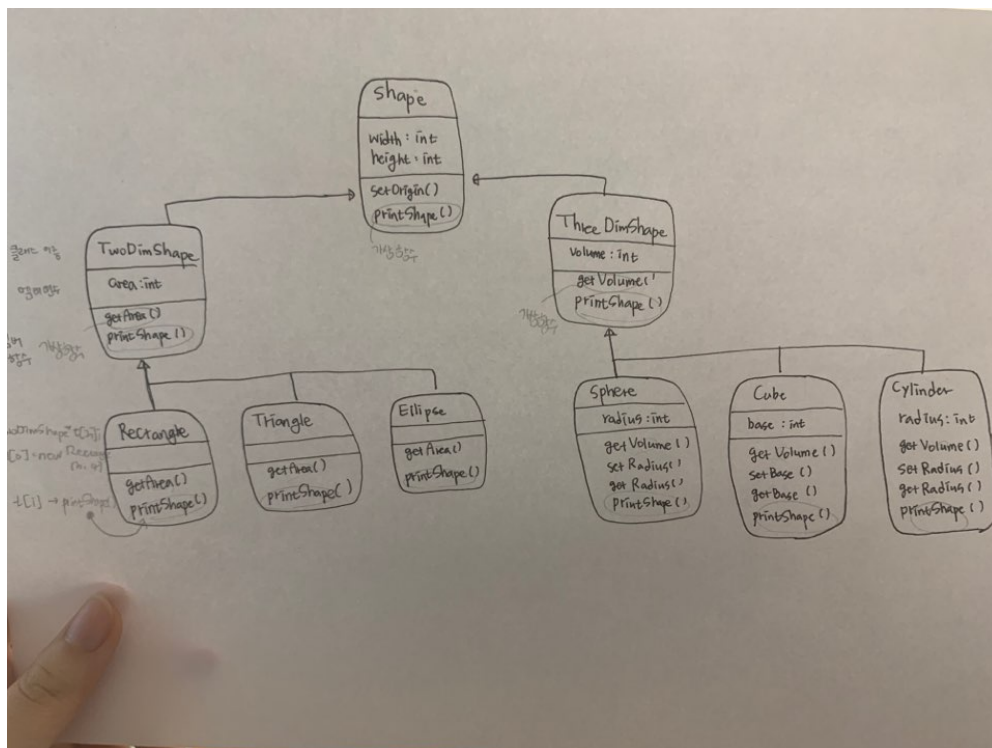
2. 실행결과화면

```
C:\Microsoft Visual Studio 디버그 콘솔

Ellipse 면적: 1.5708
Sphere 체적: 1717.67
Rectangle 면적: 12
Cube 체적: 1320
Triangle 면적: 15
Chlinder 체적: 9896.01

C:\Users\wyjysh\source\repos\Project20\
이 창을 닫으려면 아무 키나 누르세요.
```

3. 문제 정의 및 분석



Shape 클래스에 폭과 높이에 대한 생성자와 printShape()함수를 만들어 준다.
printShape()함수- 면적과 체적 값을 호출하는 함수, 가상함수로 만들어 주기

TwoDimShape 클래스와 ThreeDimShape클래스는 Shape클래스를 상속받는다.

TwoDimShape클래스에 폭과 높이에 대한 생성자를 부모클래스로부터 상속받고, getArea()함수를 만들고, printShape()함수 재정의 한다.

getArea()함수- 면적을 계산하는 함수, 가상함수로 만들어 주기

ThreeDimShape 클래스에 폭과 높이에 대한 것은 부모클래스로부터 상속받고, getVolume()함수 만들고, printShape()함수 재정의 한다.

getVolume()함수- 체적을 계산하는 함수, 가상함수로 만들어 주기

TwoDimShape의 자식클래스인 Ractangle, triangle, ellipse.

Ractangle, triangle, ellipse 클래스는 생성자를 TwoDimShape에서 상속받고 getArea()함수와 printShape()함수를 재정의한다.

getArea()- 각각의 도형 면적 구하는 공식으로 재정의

printShape()- 각각의 도형마다 출력할 문장과 면적 호출하는 함수로 재정의

ThreeDimShape의 자식클래스 Sphere, cube, cylinder.

각각의 자식클래스는 생성자를 ThreeDimShape에서 상속받고, getVolume()함수와 printShape()함수를 재정의한다.

추가적으로 필요한 멤버변수도 정의해준다.

getVolume()- 각각의 도형 면적 구하는 공식으로 재정의

printShape()- 각각의 도형마다 출력할 문장과 체적 호출하는 함수로 재정의

메인함수

부모클래스인 TwoDimShape와 ThreeDimShape에 대한 포인터 배열을 선언한다.

포인터를 사용하여 printShape()함수를 호출한다. (업캐스팅- 부모타입의 포인터가 자식 가리킴)

반복문에서 함수 호출

동적 바인딩에 의하여 서로 다른 printShape()가 호출된다.

*가상함수로 만들어 주는 이유- 객체 안에서 가상 함수로 정의하게 되면 우선순위가 높은 오버라이딩된 함수가 호출되기 때문에 부모객체의 함수가 아니라 자식클래스의 함수를 호출할 수 있게 되기 때문이다. (동적 바인딩)

*업캐스팅- 부모타입의 포인터가 자식객체를 가리킴, 부모의 이름으로 자식을 호출하기 위하여 사용한다.