Object-Oriented Programming Lab #12

Department: 응용물리학과 Student ID: 2017103038 Name: 권인회

A. Exercises (Write the questions down on your answer sheet)

(pp. 540-541), Exercises 1

(write output analysis for all exercises)

1. Consider the following C++ code:

```
#include <iostream>
#include <vector>

class Widget {
public:
        virtual int f() { return 1; }
};
class Gadget : public Widget {
public:
        virtual int f() { return 2; }
};
class Gizmo : public Widget {
public:
        virtual int f() { return 3; }
};
void do_it(Widget* w) {
        std::cout << w->f() << " ";
}
int main() {
        std::vector<Widget*> widgets;
        Widget wid;
        Gadget gad;
        Gizmo giz;
        widgets.push_back(&wid);
        widgets.push_back(&gad);
        widgets.push_back(&giz);
        for (size_t i = 0; i < widgets.size(); i++)
                do_it(widgets[i]);
}
```

(a) What does the program print?

: 1 2 3

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 131032개)이(

가) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

main 함수를 보면 widgets를 Widget*형의 벡터로 선언하였다. 각 class, Widget의 wid, Gadget의 gad, Gizmo의 giz를 선언해주고 이 class 객체들의 주소값을 벡터에 push_back하여 차례대로 넣어준다. 그리고 for문을 이용해 벡터 안의 객체의 메소드 f()를 출력해준다. 이 class의 경우 base class인 Widget의 메소드 함수 f()에 virtual 이 지정되어있기 때문에 벡터가 Widget 객체의 주소를 저장하는 타입임에도 불구하고 Widget* 타입으로 된 &gad, &giz도 각각의 파생 클래스에 있는 함수를 사용할 수 있게 된다.

(b) Would the program still compile and run if the f method within the Widget class were a pure virtual function?

: 컴파일이 되지 않는다.

추상 class 형식 Widget의 개체를 사용할 수 없는 상태이다. 상속받는 class에서 해당 함수의 내부구현을 만들어서 사용해야하는데, 지금은 main 함수에서 가장 부모 클래스인 Widget을 직접 선언해주기 때문에 당장 이 객체가 사용할 인스턴스가 없는 것이다. 따라서 컴파일이 불가능하다.

(c) How would the program run differently if the virtual keyword were removed from all the code?

: 1 1 1

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 133624개)이(

가) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

(b)에서 설명한 것처럼 virtual이 지정되지 않으면 벡터의 타입이 Widget*이기 때문에 모든 객체가 Widget에 있는 함수 method를 기준으로 실행된다.

(d) Would the program behave the same if the virtual keyword were removed from all the classes except for Widget?

: 1 2 3

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 136732개)이(

가) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

처음처럼 잘 실행된다. 이로써 base class의 메소드 함수만 virtual을 지정해줘도 프로그램이 의도대로 돌아가는 것을 볼 수 있다. 이 함수가 실행시간에 적절한지 파악해서 base에 있는 메소드를 사용할 지 파생 클래스에 있는 메소드를 사용할 지 결정한다.

B. Additional exercises (Write the questions down on your answer sheet)

B-1. Write the declarations and definitions for the geometric classes define in Figure B-1. Use the following hints: (include test codes)

a. Define Shape, TwoDimensional and ThreeDimensional as abstract classes.

b. Define PI as static class member in the Shape class.

c. Define area calculation and print functions as pure virtual functions in the Shape class.

d. Use public inheritance.

e. Define all data or member functions common to two-dimensional shapes in the TwoDimensional class. Do the same for all common data or member functions for the three-dimension shape.

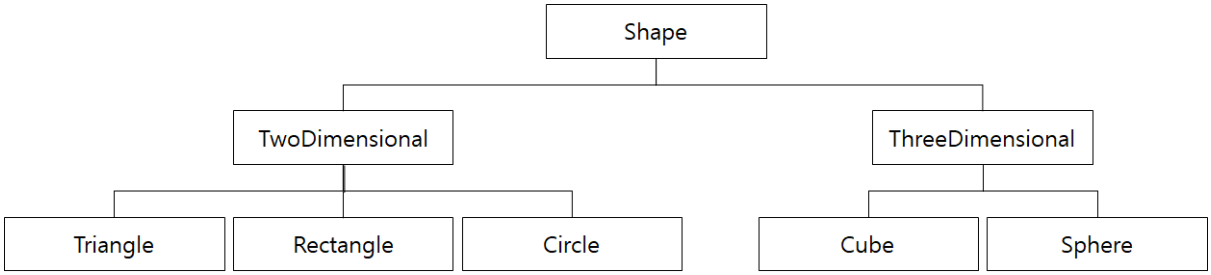

Figure B-1.

< 코드 >

```
#include <iostream>
#include <cmath>

class Shape {
public:
        // static const double PI;
        virtual double area() const = 0;
        virtual void print() {
                std::cout << area() << std::endl;
        }
};

class TwoDimensional : public Shape {
public:
        // 차원을 알려주는 함수
        void dimension() {
                std::cout << "This is TwoDimensional." << std::endl;
        }
};

class ThreeDimensional : public Shape {
```

```
public:
        void dimension() {
                std::cout << "This is ThreeDimensional." << std::endl;
        }
};

class Triangle : public TwoDimensional {
protected:
        double side1;
        double side2;
        double side3;
public:
        Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}
        double area() const override;
};

class Rectangle : public TwoDimensional {
protected:
        double length;
        double width;
public:
        Rectangle(double len, double wid) : length(len), width(wid) {}
        double area() const override;
};

class Circle : public TwoDimensional {
protected:
        double radius;
public:
        Circle(double rad) : radius(rad) {}
        double area() const override;

};

class Cube : public ThreeDimensional {
protected:
        double length;
        double width;
        double height;
public:
        Cube(double len, double wid, double hei) : length(len), width(wid), height(hei) {}
        double area() const override;
};

class Sphere : public ThreeDimensional {
protected:
        double radius;
public:
        Sphere(double rad) : radius(rad) {}
        double area() const override;
};

static double PI = 3.14159;

double Rectangle::area() const {
```

```
        return length * width;
}

double Triangle::area() const {
        double s = (side1 + side2 + side3) / 2;
        return sqrt(s * (s - side1) * (s - side2) * (s - side3));
}

double Circle::area() const {
        return PI * radius * radius;
}

double Cube::area() const {
        return 2 * length * width + 2 * length * height + 2 * width * height;
}

double Sphere::area() const {
        return 4 * PI * radius * radius;
}

int main() {
        Rectangle r1 = { 10, 10 };
        r1.print();
        Triangle t1 = { 4, 4, 4 };
        t1.print();
        Circle c1 = { 4 };
        c1.print();
        Cube c2 = { 1, 1, 1 };
        c2.print();
        Sphere s1 = { 0.5 };
        s1.print();
}
```

< 실행결과 >

100

6.9282

50.2654

6

3.14159

This is TwoDimensional.

This is ThreeDimensional.

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 94796개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

우선 문제에 주어진 조건이 잘 이해가 되지 않아 힌트를 참고하기보단 문제를 풀어 기능구현을 하도록 하였다. b에서 Shape class내에 PI 변수를 static으로 초기화하는 것은 IDE에서 불가능하다고 하며, c에서 어떤 것을 print하는 함수를 만들라는지 명시되어있지 않아 우선 area를 print하는 것으로 하였습니다. e에서 삼각형, 사각형, 원에서 공통된 요소가 무엇이 있는지 파악이 안되어 공통으로 사용할 수 있는 함수를 만들어 두었습니다. 최대한 조건을 고려하여 기능구현에 맞췄으며, 조건에서 요구한 추상 class적으로 함수를 제일 base class에 선언해두고 그것을 상속받거나 이후에 정의해주었다. area함수를 통해 각 도형마다의 넓이를 계산해주고 return해준다. 그리고 print함수는 그것을 출력해준다.

B-2. Write the declarations and definitions for the classes that work as the following codes and comments.

```
// All data members of Base and Derived classes must be declared
// as private access types
Base *p1 = new Derived(10, 20);               // (x, y)
Base *p2 = new Base(5);                       // (x)
p1->print();                    // prints 10, 20
p1->Base::print();              // prints 10
p2->print();                    // prints 5
Derived *p3 = dynamic_cast<Derived *>(p1);
if (p3 != nullptr) p3->print();                        // prints 10, 20

const Base b1 = *p2;
b1.print();                               // prints 5

Derived d1(1, 3), d2(2, 4);
Derived d3 = (d1 < d2) ? d1 : d2;      // operator <: (d1.x+d1.y) < (d2.x+d2.y)
d3.print();                               // prints 1, 3
```

< 코드 >

```
#include <iostream>

class Base {
public:
        int x;
        Base(int n) : x(n) {}
        virtual void print () const {
                std::cout << x << std::endl;
        }
};
```

```
class Derived : public Base {
        int y;
public:
        Derived(int n1, int n2) : Base(n1), y(n2) {}
        void print() const {
                std::cout << x << ", " << y << std::endl;
        }
        friend bool operator<(const Derived& d1, const Derived& d2);
};
bool operator<(const Derived& d1, const Derived& d2) {
        return (d1.x + d1.y) < (d2.x + d2.y);
}

int main() {
        Base* p1 = new Derived(10, 20);           // (x, y)
        Base* p2 = new Base(5);                   // (x)
        p1->print();                      // prints 10, 20
        p1->Base::print();                // prints 10
        p2->print();                      // prints 5
        Derived* p3 = dynamic_cast<Derived*>(p1);
        if (p3 != nullptr) p3->print();                   // prints 10, 20

        const Base b1 = *p2;
        b1.print();                               // prints 5

        Derived d1(1, 3), d2(2, 4);
        Derived d3 = (d1 < d2) ? d1 : d2; // operator <: (d1.x+d1.y) < (d2.x+d2.y)
        d3.print();                               // prints 1, 3

}
```

< 실행결과 >

10, 20

10

5

10, 20

5

1, 3

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 106556개)이(

가) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

아무 조건이 없어 가장 간단하게 기능 구현 하는 것에 목적을 두고 하였다. 사용하고 있는 class인 Base와 Derived를 정의해준다. class 이름으로부터 Base class를 상속받는 Derived class임을 알 수 있고, print함수를 상속받아 사용하고 있다. 그리고 마지막 코드에서 비교연산자 <를 사용하므로 연산자 오버로딩도 해줘야한다. 그렇게 코드를 구축해서 실행시키면 주석에서 원하는 결과가 나온다.

B-3. Write the declarations and definitions for the classes that work as the following codes and comments.

```
Base b1(2), b2(10);

b1.print();             // 2
b2.print();             // 10
for (int i = 0; i < 5; i++) {
        b1.setN(i, (i+1) * 20);
        b2.setN(i, (i+1) * 10);
}
b1.printData();  // 20 40
b2.printData();  // 10 20 30 40 50 0 0 0 0 0

Derived d(5);
d.print();              // 5
d.printData();          // 0 0 0 0 0
for (int i = 0; i < 10; i++) {
        d.setN(i, (i + 1) * 3);
}
d.printData(); // 3 6 9 12 15
d.insert(99);  // "Base" class does not have "insert" method.
d.printData(); // 3 6 9 12 15 99
```

< 코드 >

```
#include <iostream>
#include <vector>

class Base {
public:
        std::vector<int> v;
        Base(int n) : v(n) {}
        virtual void print () const {
                std::cout << v.size() << std::endl;
        }
        virtual void printData() const {
```

```
                for (int i = 0; i < v.size(); i++) {
                        std::cout << v[i] << ' ';
                }
                std::cout << '₩n';
        }
        virtual void setN(int x, int y) {
                if (x < v.size())
                        v[x] = y;
        }
};

class Derived : public Base {
public:
        Derived(int n) : Base(n) {}
        void insert(int n) {
                v.push_back(n);
        }
};

int main() {
        Base b1(2), b2(10);

        b1.print();             // 2
        b2.print();             // 10
        for (int i = 0; i < 5; i++) {
                b1.setN(i, (i + 1) * 20);
                b2.setN(i, (i + 1) * 10);
        }
        b1.printData();  // 20 40
        b2.printData();  // 10 20 30 40 50 0 0 0 0 0

        Derived d(5);
        d.print();              // 5
        d.printData();          // 0 0 0 0 0
        for (int i = 0; i < 10; i++) {
                d.setN(i, (i + 1) * 3);
        }
        d.printData();   // 3 6 9 12 15
        d.insert(99);    // "Base" class does not have "insert" method.
        d.printData();   // 3 6 9 12 15 99

}
```

< 실행결과 >

2

10

20 40

10 20 30 40 50 0 0 0 0 0

5

0 0 0 0 0

3 6 9 12 15

3 6 9 12 15 99

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 113660개)이(

가) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

사용되는 class Base와 Derived를 정의해준다. 둘 다 파라미터로 int형 정수 하나를 받는다. 이 정수는 class 내에 있는 벡터의 사이즈를 결정해준다. 그래서 print 메소드는 class에 있는 벡터의 사이즈를 출력하도록 하였다. setN 메소드는 파라미터로 정수 두개를 받아서 첫번째 정수의 벡터 요소에 두번째 정수의 값을 할당한다. 이 경우, 벡터의 크기를 벗어날 위험이 있으므로 if문을 이용해준다. printData 메소드는 벡터의 요소들을 다 출력해주도록 한다. insert 메소드는 Derived class에서만 사용할 수 있는 것으로, 파라미터로 받은 정수를 벡터에 push_back 해준다. 이로써, 주석에 쓰여있는 기능을 모두 수행한다.