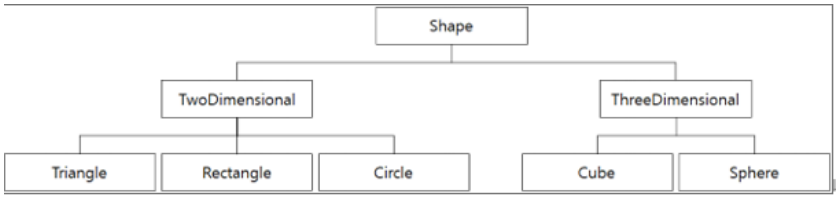
Object-Oriented Programming Lab #7

Department: 화학공학과 **Student ID: 2019101074** **Name: 안용상**

1. 아래와 같은 특징을 가지는 그림 1과 같은 클래스를 선언하고 정의하라. (테스트할 코드도 작성해서 결과를 포함할 것)

- Shape, TwoDimensional, ThreeDimensional 클래스는 추상 클래스이다.
- Shape 클래스는 면적을 계산하고 출력하는 순수 가상 함수를 가진다.
- TwoDimensional 클래스는 둘레를 계산하고 출력하는 순수 가상 함수를 가진다.
- ThreeDimensional 클래스는 부피를 계산하고 출력하는 순수 가상 함수를 가진다.
- Shape 클래스의 PI는 static 멤버이다. (원주율로 소수점 5자리까지 지정)
- 삼각형, 사각형, 원, 육면체, 구를 그림 1과 같이 각각 Triangle, Rectangle, Circle, Cube, Sphere 클래스로 정의하고, 각 도형을 표현할 수 있는 변수를 정의하고, 필요한 함수를 정의하라.

그림 1.



소스코드

```
#include <iostream>
#include <cmath> // 제곱근이나 제곱을 연산하기위해 include

class Shape {
public:
  Shape() {}; // Shape 클래스의 기본 생성자
  virtual ~Shape() {}; // Shape 클래스의 가상 소멸자
  static double PI; // 정적 멤버 변수 PI 선언
  virtual void area() const = 0; // 면적을 계산하고 출력하는 순수 가상 함수
};

class TwoDimensional : public Shape {
  double width; // 너비
  double height; // 높이
public:
  TwoDimensional(double x, double y) : width(x), height(y) {}; // 생성자로 너비와 높이 초기화
  virtual ~TwoDimensional() {} // TwoDimensional 클래스의 가상 소멸자
  virtual void area() const = 0; // 면적을 계산하고 출력하는 순수 가상 함수
  virtual void round() const = 0; // 둘레를 계산하고 출력하는 순수 가상 함수
  double get_width() const { return width; }; // 너비 getter
  double get_height() const { return height; }; // 높이 getter
};

class Triangle : public TwoDimensional {
  double cross_x; // 교차점 x 좌표
public:
  Triangle(double x, double y, double cross_x) :
    TwoDimensional(x, y), cross_x(cross_x) {}; // 생성자로 너비, 높이, 교차점 x 좌표 초기화
  virtual ~Triangle() {}; // Triangle 클래스의 가상 소멸자
  void area() const override { //추상클래스였던 area를 Triangle클래스에서 override해서 정의
    std::cout << "this is Triangle \n this area is "
      << get_width() * get_height() / 2
      << std::endl
      << std::endl; // 삼각형의 면적 계산 및 출력
  };
  double get_cross_x() const {//삼각형의 둘레를 어떻게 얻을 수 있을까 고민하다가,
    //높이 선과 밑변선이 서로 교차하는 지점을 이용하는 아이디어를 내봤습니다.
    return cross_x; // 교차점 x 좌표 getter
```

```
  }
  void round() const override {//추상클래스였던 round를 Triangle클래스에서 override해서 정의
    std::cout << "this is Triangle \n this round is "
      << get_width() + std::sqrt(std::pow(get_cross_x(), 2) + std::pow(get_height(), 2))
      + std::sqrt(std::pow(get_width() - get_cross_x(), 2) + std::pow(get_height(), 2))
      << std::endl << std::endl; // 삼각형의 둘레 계산 및 출력
  };
};

class Rectangle : public TwoDimensional {
public:
  Rectangle(double x, double y) : TwoDimensional(x, y) {}; // 생성자로 너비와 높이 초기화
  ~Rectangle() {}; // Rectangle 클래스의 소멸자
  void area() const override {//추상클래스였던 area를 Rectangle클래스에서 override해서 정의
    std::cout << "this is Rectangle \n this area is "
      << get_width() * get_height()
      << std::endl
      << std::endl; // 직사각형의 면적 계산 및 출력
  };
  void round() const override {//추상클래스였던 round를 Rectangle클래스에서 override해서 정의
    std::cout << "this is Rectangle \n this round is "
      << get_width() * 2 + get_height() * 2
      << std::endl
      << std::endl; // 직사각형의 둘레 계산 및 출력
  };
};

class Circle : public TwoDimensional {
public:
  Circle(double x) : TwoDimensional(x, x) {}; // 생성자로 반지름 초기화
  ~Circle() {}; // Circle 클래스의 소멸자
  void area() const override {//추상클래스였던 area를 Circle클래스에서 override해서 정의
    std::cout << "this is circle \n this area is "
      << get_width() * get_width() * Shape::PI
      << std::endl
      << std::endl; // 원의 면적 계산 및 출력
  };
  void round() const override {//추상클래스였던 round를 Circle클래스에서 override해서 정의
    std::cout << "this is circle \n this round is "
      << PI * 2 * get_width()
      << std::endl
      << std::endl; // 원의 둘레 계산 및 출력
  }
};

class ThreeDimensional : public Shape {
  double width; // 너비
  double height; // 높이
  double z_index; // Z 축 값
public:
  ThreeDimensional(double w, double h, double z) : width(w), height(h), z_index(z) {};
  // 생성자로 너비, 높이, Z 축 값 초기화
  virtual ~ThreeDimensional() {}; // ThreeDimensional 클래스의 가상 소멸자
  virtual void area() const = 0; // 면적을 계산하고 출력하는 순수 가상 함수
  virtual void volume() const = 0; // 부피를 계산하고 출력하는 순수 가상 함수
  double get_width() const {
    return width; // 너비 getter
  };
  double get_height() const {
    return height; // 높이 getter
  }
  double get_zindex() const {
    return z_index; // Z 축 값 getter
  };
};

class Cube : public ThreeDimensional {
public:
  Cube(int x, int y, int z) : ThreeDimensional(x, y, z) {} // 생성자로 너비, 높이, Z 축 값 초기화
  ~Cube() {}; // Cube 클래스의 소멸자
  void area() const override {
    std::cout
      << "this is a cube\n this area is "
      << get_width() * get_height() * 2 + get_width() * get_zindex() * 2 + get_height() * get_zindex() * 2
      << std::endl
      << std::endl; // 정육면체의 면적 계산 및 출력
  };
  void volume() const override {
    std::cout
      << "this is a cube\n this volume is "
      << get_width() * get_height() * get_zindex()
      << std::endl
      << std::endl; // 정육면체의 부피 계산 및 출력
  };
```

```
};

class Sphere : public ThreeDimensional {
public:
  Sphere(int x) : ThreeDimensional(x, x, x) {}; // 생성자로 반지름 값 초기화
  ~Sphere() {}; // Sphere 클래스의 소멸자
  void area() const override {
    std::cout
      << "this is a sphere\n this area is "
      << get_width() * get_width() * 4 * PI
      << std::endl
      << std::endl; // 구의 면적 계산 및 출력
  };
  void volume() const override {
    std::cout
      << "this is a sphere\n this volume is "
      << std::pow(get_width(), 3) * 4 / 3 * PI
      << std::endl
      << std::endl; // 구의 부피 계산 및 출력
  };
};

double Shape::PI = 3.14159; // 정적 멤버 변수 PI 초기화

int main() {
  Circle cir(4); // 반지름이 4인 원 객체 cir를 생성
  cir.area(); // cir 객체의 면적을 계산하고 출력
  cir.round(); // cir 객체의 둘레를 계산하고 출력
  Triangle tri(3, 4, 1); // 밑변의 길이가 3, 높이가 4인 삼각형 객체 tri를 생성
  tri.area(); // tri 객체의 면적을 계산하고 출력
  tri.round(); // tri 객체의 둘레를 계산하고 출력

  Rectangle rec(6, 4); // 너비가 6이고 높이가 4인 직사각형 객체 rec를 생성
  rec.area(); // rec 객체의 면적을 계산하고 출력
  rec.round(); // rec 객체의 둘레를 계산하고 출력

  Cube cube(12, 3, 5); // 너비가 12, 높이가 3, Z 축 값이 5인 정육면체 객체 cube를 생성
  cube.area(); // cube 객체의 면적을 계산하고 출력
  cube.volume(); // cube 객체의 부피를 계산하고 출력

  Sphere sph(6); // 반지름이 6인 구 객체 sph를 생성
  sph.area(); // sph 객체의 면적을 계산하고 출력
  sph.volume(); // sph 객체의 부피를 계산하고 출력

  return 0; // main 함수의 반환값을 0으로 설정하여 프로그램을 종료
}
```

결과코드

```
this is circle
 this area is 50.2654

this is circle
 this round is 25.1327

this is Triangle
 this area is 6

this is Triangle
 this round is 11.5952

this is Rectangle
 this area is 24

this is Rectangle
 this round is 20

this is a cube
 this area is 222

this is a cube
 this volume is 180

this is a sphere
 this area is 452.389

this is a sphere
 this volume is 904.778


C:\Users\admin\Desktop\객체지향동영상\객체지향12강클래스실습02\12강과제1번\x64\Debug\12강과제1번.exe(프로세스 15552개)이(가) 종료되었습니다(코드: 0개).
```

```
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

2. 아래의 주석과 같이 동작하는 클래스를 선언하고 정의하라.

```
// 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
```

소스코드

```
#include <iostream>
class Base {
  int x;  // private data member x
public:
  Base(int x) : x(x) {};  // Base 클래스의 생성자
  virtual void print() const {  // 가상 함수 print() 정의
    std::cout << x << std::endl;  // x 출력
  }
  int get_x() const {  // x 값을 반환하는 멤버 함수
    return x;
  }
};
class Derived : public Base {
  int y;  // private data member y
public:
  Derived(int x, int y) : Base(x), y(y) {};  // Derived 클래스의 생성자, Base 클래스의 생성자 호출
  void print() const override {  // print() 함수를 오버라이드하여 재정의
    std::cout << get_x() << ", " << y << std::endl;  // x와 y 출력
  }
  Derived& operator=(const Derived& der) = default;  // 대입 연산자를 디폴트로 설정
  bool operator<(const Derived& der) {  // < 연산자 정의
    if ((this->get_x() + this->y) > (der.get_x() + der.y)) return 0;  // x + y가 der.x + der.y보다 크면 false 반환
    else return 1;  // 그렇지 않으면 true 반환
  }
};
int main() {
  Base* p1 = new Derived(10, 20);  // Base 포인터 p1은 Derived 객체를 가리킴
  Base* p2 = new Base(5);  // Base 포인터 p2는 Base 객체를 가리킴
  p1->print();  // Derived 클래스의 print() 함수 호출, 10, 20 출력
  p1->Base::print();  // Base 클래스의 print() 함수 호출, 10 출력
  p2->print();  // Base 클래스의 print() 함수 호출, 5 출력
  Derived* p3 = dynamic_cast<Derived*>(p1);  // p1을 Derived 포인터로 다운캐스팅
  if (p3 != nullptr) p3->print();  // p3가 nullptr이 아니라면 Derived 클래스의 print() 함수 호출, 10, 20 출력

  const Base b1 = *p2;  // b1은 p2의 복사본, const로 선언되어 값을 수정할 수 없음
  b1.print();  // Base 클래스의 print() 함수 호출, 5 출력

  Derived d1(1, 3), d2(2, 4);  // Derived 객체 d1과 d2 생성
  Derived d3 = (d1 < d2) ? d1 : d2;  // < 연산자를 이용하여 d1과 d2를 비교하여 작은 값을 d3에 할당
  d3.print();  // Derived 클래스의 print() 함수 호출, 1, 3 출력
}
```

출력

```
10, 20
10
5
10, 20
5
1, 3
```

```
C:\Users\admin\Desktop\객체지향동영상\객체지향12강클래스실습02\12강과제2번\x64\Debug\12강과제2번.exe(프로세스 17896개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

3. 아래의 주석과 같이 동작하는 클래스를 선언하고 정의하라.

```
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
```

소스코드

```
/*주어진 코드는 `Base` 클래스와 `Derived` 클래스를 포함한다
 `Base` 클래스는 정수형 멤버 변수 `x`와 정수형 벡터 `nVec`를 가지며,
 `Derived` 클래스는 `Base` 클래스를 상속받는다.

`Base` 클래스의 생성자 `Base(int x)`는 `x`를 인자로 받아 `x` 값을 설정하고,
벡터 `nVec`를 크기 `x`로 초기화한다. `print()` 함수는 `x` 값을 출력하며,
`setN()` 함수는 인덱스 `i`에 있는 `nVec`의 요소를 `result` 값으로 설정한다.
`printData()` 함수는 `nVec`의 모든 요소를 공백으로 구분하여 출력한다.
`get_nVec()` 함수는 `nVec` 벡터를 반환한다.

`Derived` 클래스는 `Base` 클래스를 상속받아 생성되며,
`insert()` 함수는 `nVec` 벡터에 값을 추가하는 기능을 한다.

`main()` 함수에서는 `Base` 클래스와 `Derived` 클래스의 객체를 생성하고,
해당 객체들의 멤버 함수를 호출하여 예상 출력을 확인한다.
`b1`과 `b2` 객체의 `print()` 함수를 통해 `x` 값을 출력하고,
`setN()` 함수를 사용하여 `nVec`의 요소를 설정한다.
그 후 `printData()` 함수를 호출하여 `nVec`의 요소를 출력한다.

`d` 객체는 `Derived` 클래스로 생성되었으며,
`print()` 함수로 `x` 값을 출력하고,
`printData()` 함수로 `nVec`의 요소를 출력한다.
`setN()` 함수를 사용하여 `nVec`의 요소를 설정하고,
`insert()` 함수를 사용하여 `nVec`에 값을 추가한다.

마지막으로 `printData()` 함수를 호출하여 `nVec`의 최종 결과를 출력한다.

*/

#include <iostream>
#include <vector>

class Base {
    int x;                          // 정수형 멤버 변수 x 선언
    std::vector<int> nVec;          // 정수형 벡터 nVec 선언

public:
    Base(int x) : x(x) {            // Base 클래스의 생성자 정의
        for (int i = 0; i < x; i++) {
            nVec.push_back(0);      // 벡터 nVec에 0 추가
        }
    }

    void print() const {            // print 함수 정의
        std::cout << x << std::endl;    // x 출력
    }
```

```
    // 인덱스 i에 있는 요소의 값을 result로 설정하는 함수
    virtual void setN(int i, int result) {
        if (i < x)                   // i가 x보다 작을 경우에만 실행
            nVec[i] = result;        // 벡터 nVec의 i번째 요소를 result로 설정
    }

    void printData() const {        // printData 함수 정의
        for (int elem : nVec) {
            std::cout << elem << ' ';   // nVec의 요소들을 공백으로 구분하여 출력
        }
        std::cout << std::endl;
    }

    std::vector<int>& get_nVec() {  // nVec 벡터를 반환하는 함수
        return nVec; // 이 함수는 const함수로 작성이 불가능했다. reference로 실체값을
//반환해야하기 때문에, const로 접근하기에는 const의 의미가 흐려지는 상황이 생길 수 있어서
//그렇게 막아놓은것같다.
    }
};

class Derived : public Base {       // Base 클래스를 상속받는 Derived 클래스 정의
public:
    Derived(int x) : Base(x) {}     // Derived 클래스의 생성자 정의

    void insert(int i) {            // insert 함수 정의
        get_nVec().push_back(i);    // nVec 벡터에 i 추가
    }
};

int main() {
    Base b1(2), b2(10);             // Base 클래스의 객체 b1, b2 생성

    b1.print();                     // Output: 2 - b1의 x 값 출력
    b2.print();                     // Output: 10 - b2의 x 값 출력

    for (int i = 0; i < 5; i++) {
        b1.setN(i, (i + 1) * 20);   // b1의 nVec 요소를 (i+1)*20으로 설정
        b2.setN(i, (i + 1) * 10);   // b2의 nVec 요소를 (i+1)*10으로 설정
    }

    b1.printData();                 // Output: 20 40 - b1의 nVec 요소 출력
    b2.printData();                 // Output: 10 20 30 40 50 0 0 0 0 0 - b2의 nVec 요소 출력

    Derived d(5);                   // Derived 클래스의 객체 d 생성

    d.print();                      // Output: 5 - d의 x 값 출력
    d.printData();                  // Output: 0 0 0 0 0 - d의 nVec 요소 출력

    for (int i = 0; i < 10; i++) {
        d.setN(i, (i + 1) * 3);     // d의 nVec 요소를 (i+1)*3으로 설정
    }

    d.printData();                  // Output: 3 6 9 12 15
                                    //d의 벡터를 출력.
  d.insert(99); // Derive클래스에 정의된 insert함수를 통해 벡터에 99를 push_back한다
  d.printData();  // Output: 3 6 9 12 15 99 - d의 nVec 요소에 99를 추가한 후 출력

}
```

출력

```
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\admin\Desktop\객체지향동영상\객체지향12강클래스실습02\12강과제3번\x64\Debug\12강과제3번.exe(프로세스 20828개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```