자료구조 실습06

Data Structures Lab06

Lab06

◎ 목표: 이질리스트의 이해와 적용

◎ 내용:

- ☞ 다양한 타입의 자료를 저장할 수 있는 이질리스트 이해
- ☞ 상속(inheritance), 가상함수(virtual function), 오버로딩(overloading), 오버라이딩(overriding) 이해

◎ 방법

- ☞ 주어진 코드를 분석하여 상속 및 오버로딩을 이해

◎ 과제

- ☞ 주어진 코드를 분석해보고 이질 리스트를 사용함으로써 얻어지는 장점을 설명하라
- ☞ 현재 음원 관리 프로그램에서 이질 리스트를 어떻게 사용할 수 있을 지 클래스 다이어그램을 통해 설명

Overloading / Overriding

◎ **Overloading**

- ☞ 같은 이름의 함수를 다른 매개 변수로 다르게 정의하는 것
- ☞ ex) 연산자 오버로딩

◎ **Overriding**

- ☞ 상위 클래스가 가지고 있는 함수를 하위 클래스에서 재정의해 사용하는 것

◎ 위의 방법들을 통해 프로그램은 다형성 및 재사용성을 확보!

동적 바인딩 / 정적 바인딩

```cpp
#include <iostream>

using namespace std;

class parent {
public :
    virtual void v_print() {
        cout << "parent" << "\n";
    }
    void print() {
        cout << "parent" << "\n";
    }
};

class child : public parent {
public :
    void v_print() {
        cout << "child" << "\n";
    }
    void print() {
        cout << "child" << "\n";
    }
};

int main() {
    parent* p;
    child c;
    p = &c;

    p->v_print();
    p->print();

    return 0;
}
```

◎ 정적 바인딩

☞ 컴파일하는 시점에서 호출 주소가 결정

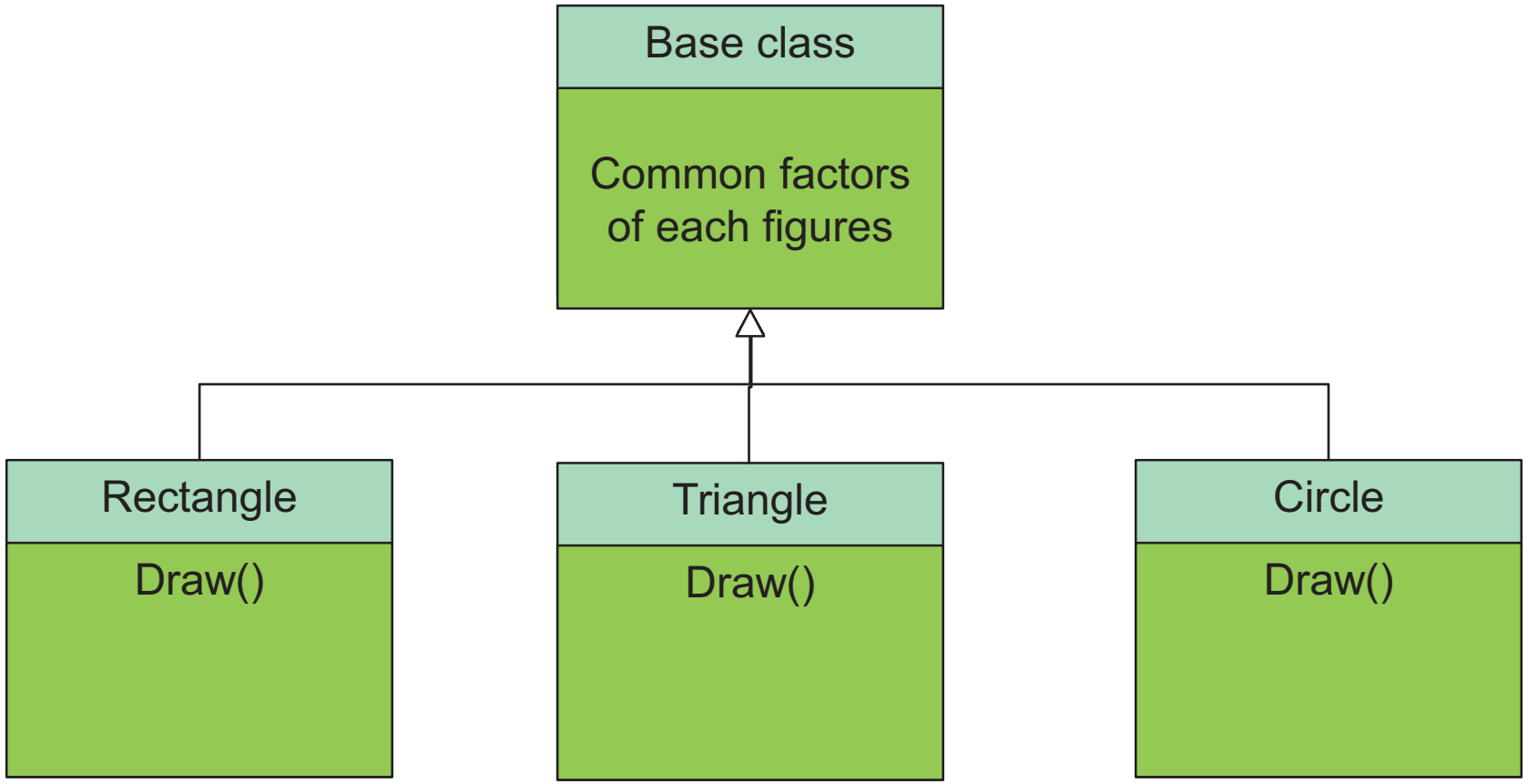
◎ 동적 바인딩

☞ 런타임 중 함수 호출이 되었을 때 주소를 알려주는 것.

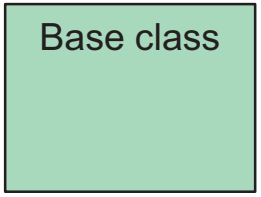
☞ 가상 함수가 그 대표적인 예이다.

☞ 동적 바인딩은 한 번의 함수 호출을 위해 두 번의 접근 필요(가상함수 테이블) 따라서 성능상 안좋을 수 있지만 다형성이라는 장점을 얻을 수 있음.

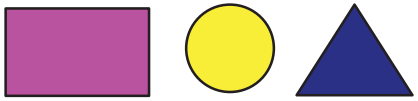
Outline


Base class
Common factors of each figures
Rectangle
Draw()
Triangle
Draw()
Circle
Draw()

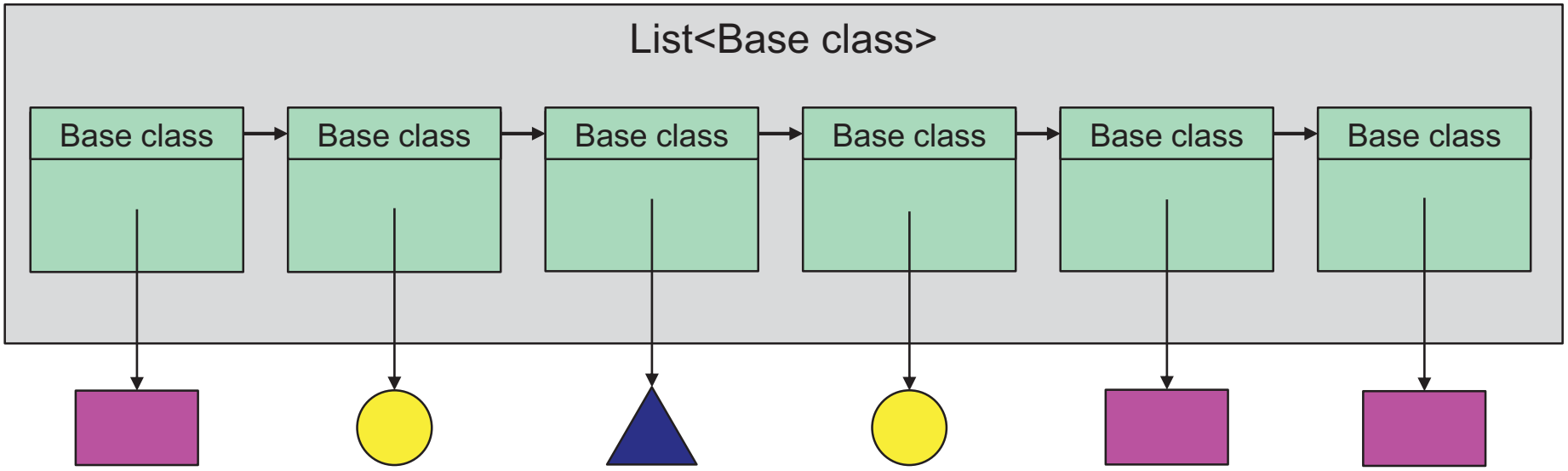
Outline



: Base class 노드



: 각 도형들



MyBase

```
class MyBase
{
public:
    MyBase();

   virtual void Draw()=0;          // 순수 가상함수
   virtual void Display()=0;       // 순수 가상함수

protected:
   int attribute;                  // 도형의 속성
   double area;                    // 도형의 넓이
};
```

MyRectangle

```
class MyRectangle : public MyBase
{
public:
    MyRectangle();          // this->attribute = 0;

   void Draw();             // rectangle을 생성 + 도형 넓이 계산
   void Display();          // 화면에 출력

private:
   Point vertex[2];
};
```

MyTriangle

```
class MyTriangle : public MyBase
{
public:
    MyTriangle ();          // this->attribute = 1;

   void Draw();             // triangle 을 생성 + 도형 넓이 계산
   void Display();          // 화면에 출력

private:
   Point vertex[2];
};
```

MyCircle

```
class MyCircle : public MyBase
{
public:
    MyCircle();                 // this->attribute = 2;

   void Draw();                 // circle을 생성 + 도형 넓이 계산
   void Display();              // 화면에 출력

private:
   Point vertex[2];
};
```

Reference

◎ 자식 클래스(Derived class) 객체는 부모 클래스(Base class) 객체로 포인팅 가능

☞ 즉, 이질리스트는 base class의 object pointer를 이용하여 다양한 자료를 입력

➢ 따라서 모두 포인터 변수로 선언되어야 함

☞ 모든 자료형은 base class를 상속받아 구현되어야 함

◎ 예시

```
MyBase * newBase;

if ( 생성하고자 하는 것이 원 )
        newBase = new MyCircle;
else if ( 생성하고자 하는 것이 사각형 )
        newBase = new MyRectangle;
…
```

Reference

◎ 오버라이드(override) 멤버 호출

☞ 부모 객체(Base object)를 통해 자식 객체(Derived object)의 오버라이드(override)된 멤버 호출 방법

(newBase= new MyCircle; 로 동적할당 된 경우)

```
(MyCircle*)newBase->Draw();
// MyCircle의 Draw() 구현이 호출된다.
```