Bridge 패턴

타입

Structural Pattern

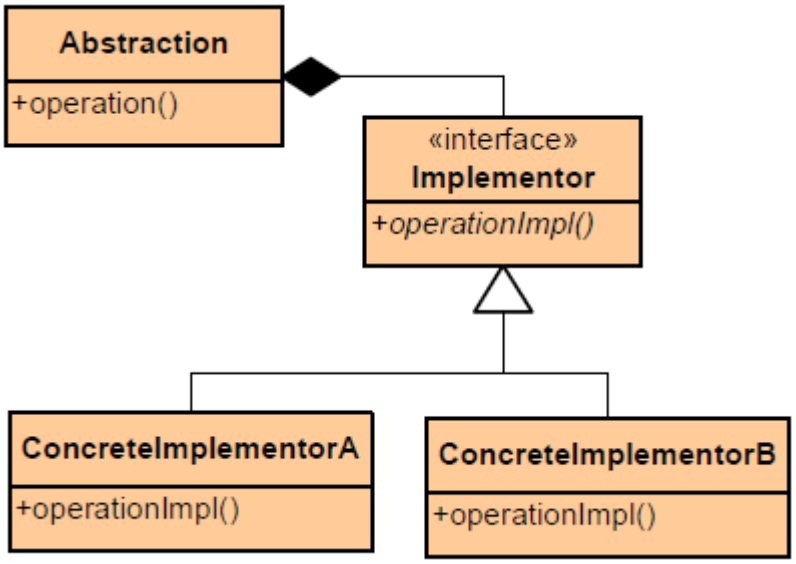
문제

클래스의 내용이 자주 바뀐다.

해결

인터페이스와 구현 클래스를 분리한다.

클래스 다이어그램



예제

bridge.cpp

```
#include <iostream>

using namespace std;

/* Implementor*/
```

```
class DrawingAPI {
  public:
   virtual void drawCircle(double x, double y, double radius) = 0;
   virtual ~DrawingAPI() {}
};

/* Concrete ImplementorA*/
class DrawingAPI1 : public DrawingAPI {
  public:
   void drawCircle(double x, double y, double radius) {
      cout << "API1.circle at " << x << ':' << y << ' ' << radius <<
endl;
   }
};

/* Concrete ImplementorB*/
class DrawingAPI2 : public DrawingAPI {
public:
   void drawCircle(double x, double y, double radius) {
      cout << "API2.circle at " << x << ':' << y << ' ' <<  radius <<
endl;
   }
};

/* Abstraction*/
class Shape {
  public:
   virtual ~Shape() {}
   virtual void draw() = 0;
   virtual void resizeByPercentage(double pct) = 0;
};

/* Refined Abstraction*/
class CircleShape : public Shape {
  public:
   CircleShape(double x, double y,double radius, DrawingAPI
*drawingAPI) :
           m_x(x), m_y(y), m_radius(radius), m_drawingAPI(drawingAPI)
   {}
   void draw() {
      m_drawingAPI->drawCircle(m_x, m_y, m_radius);
   }
   void resizeByPercentage(double pct) {
      m_radius *= pct;
   }
  private:
   double m_x, m_y, m_radius;
   DrawingAPI *m_drawingAPI;
};

int main(void) {
```

```
    CircleShape circle1(1,2,3,new DrawingAPI1());
    CircleShape circle2(5,7,11,new DrawingAPI2());
    circle1.resizeByPercentage(2.5);
    circle2.resizeByPercentage(2.5);
    circle1.draw();
    circle2.draw();
    return 0;
}
```

이 예제에서는 Shape (또는 이를 상속받는 클래스) 이 인터페이스에 해당된다.

참고

http://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns#Bridge

From:
http://obg.co.kr/doku/ - **OBG WiKi**

Permanent link:
http://obg.co.kr/doku/doku.php?id=programming:design_pattern:bridge

Last update: **2020/11/29 14:09**