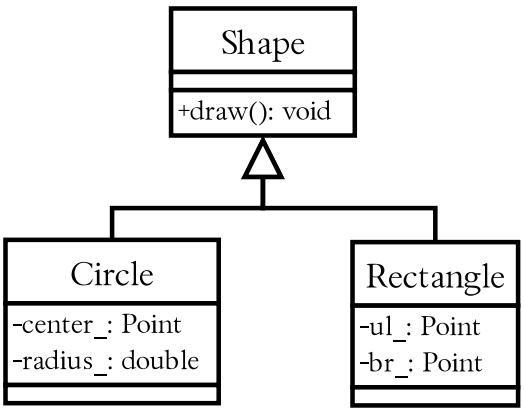
# Наследяване. Полиморфизъм.
# (*Упражнение*)

Любомир Чорбаджиев
Технологическо училище "Електронни системи"
Технически университет, София
lchorbadjiev@elsys-bg.org
*Revision* : 1.2

9 януари 2005 г.

**Задача 1:** Да се разработят `C++` класове, съответстващи на следната йерархия:




Класът `Shape` е абстрактен клас, а класовете `Rectangle` и `Circle` са конкретни реализации, предефиниращи виртулания метод `draw()`.
□

**Задача 2:** Да се разработи клас `Drawing`, който да държи указатели към обекти от типа `Shape`. Класът `Drawing` да притежава метод `draw()`, който съответно извиква методите `draw()` на всички обекти `Shape`, включени в `Drawing`.
□

**Задача 3:** Да се разработи клас `BoxedCircle`, който представлява окръжност, около която е описан квадрат.
□

```
1  #include <iostream>
2  using namespace std;
3
4  class Point {
5    double x_, y_;
6  public:
7    Point(double x=0, double y=0)
8      : x_(x), y_(y)
9    {}
10
11   double get_x() const { return x_;}
12   double get_y() const { return y_;}
```

```
13
14    void dump(void) {
15      cout << "(" << x_ << ",␣" << y_ << ")";
16    }
17  };
18
19  class Shape {
20  public:
21    virtual void draw() = 0;
22    virtual ~Shape() {}
23  };
24
25  class Rectangle : public Shape {
26    Point ul_;
27    Point br_;
28  public:
29    Rectangle(const Point& ul, const Point& br)
30      : ul_(ul), br_(br)
31    {}
32
33    virtual void draw() {
34      cout << "Rectangle(";
35      ul_.dump();
36      cout << ",␣";
37      br_.dump();
38      cout << ");" << endl;
39    }
40  };
41
42  class Circle : public Shape {
43    Point center_;
44    double radius_;
45  public:
46    Circle(const Point& center, double radius)
47      : center_(center), radius_(radius)
48    {}
49    virtual void draw() {
50      cout << "Circle(";
51      center_.dump();
52      cout << ",␣" << radius_ << ");" << endl;
53    }
```

```
54 };
55
56
57 class Drawing {
58   static const int size_=10;
59   Shape* shapes_[size_];
60   int last_index_;
61 public:
62   Drawing()
63     : last_index_(0)
64   {}
65
66   ~Drawing() {
67     for(int i=0;i<last_index_;i++) {
68       delete shapes_[i];
69       shapes_[i]=0;
70     }
71   }
72
73   void add_shape(Shape* shape) {
74     if(last_index_>=size_) {
75       cout << "ERROR: too many shapes..." << endl;
76       return;
77     }
78     shapes_[last_index_]=shape;
79     last_index_++;
80   }
81
82   void draw() {
83     cout << "DRAWING:" << endl;
84     cout << "----------------------------" << endl;
85     for(int i=0;i<last_index_;i++) {
86       shapes_[i]->draw();
87     }
88     cout << "----------------------------" << endl;
89   }
90 };
91
92 int main() {
93
94   Drawing drawing;
```

```
95
96    drawing.add_shape(new Rectangle(Point(0.0,0.0),
97                                    Point(1.0,1.0)));
98    drawing.add_shape(new Circle(Point(5.0,5.0),2.0));
99
100   drawing.draw();
101
102   return 0;
103 }
```

```
lubo@dobby:~/school/cpp/notes> a.out
DRAWING:
----------------------------
Rectangle((0, 0), (1, 1));
Circle((5, 5), 2);
----------------------------
```

```
1  class BoxedCircle: public Shape {
2    Shape* shapes_[2];
3  public:
4    BoxedCircle(const Point& center, double radius) {
5      shapes_[0]=new Circle(center,radius);
6      shapes_[1]=new Rectangle(Point(center.get_x()-radius,
7                                     center.get_y()-radius),
8                               Point(center.get_x()+radius,
9                                     center.get_y()+radius));
10   }
11   ~BoxedCircle() {
12     delete shapes_[0];
13     delete shapes_[1];
14   }
15   virtual void draw() {
16     cout << "BOXED␣CIRCLE:..............." << endl;
17     shapes_[0]->draw();
18     shapes_[1]->draw();
19     cout << "............................" << endl;
20   }
21 };
```