```
1  # ifndef __shapes__
2  # define __shapes__
3
4  // Abstract class Shape.
5  // Methods are purely virtual because a generic shape is an
6  // abstract concept and its method cannot have an
7  // implementation.
8
9
10
11 ////////////////////////////////////////////////////////////
12 //                    BASE CLASS                          //
13 ////////////////////////////////////////////////////////////
14
15 // This base class that has no specification at all but provides
16 // with an idea to build on to derive specific shapes.
17
18 class Shape {
19 public:
20     // = 0 makes the methods pure virtual
21     // They are necessary to exploit polymorphism
22     virtual double area()      = 0;
23     virtual double perimeter() = 0;
24     virtual double height()    = 0;
25     virtual double width()     = 0;
26     virtual void   rotate()    = 0;
27
28 };
29
30
31
32 ////////////////////////////////////////////////////////////////
33 //                        RECTANGLE                           //
34 ////////////////////////////////////////////////////////////////
35
36 class Rectangle : public Shape {
37 private:
38     double b;
39     double h;
40 public:
41     // Constructor
42     Rectangle(double x, double y);
43
44     // These methods are customised for this derived class
45     double area();
46     double perimeter();
47     double height();
48     double width();
49     void   rotate();
50 };
51
52
53
54
55
56
57
58
```

```
59 ////////////////////////////////////////////////////////////////////
60 //                          SQUARE                                //
61 ////////////////////////////////////////////////////////////////////
62
63 // Square is basically a specific rectangle so it inherits from
64 // Rectangle but it doesn't need to customize Rectangle's methods
65
66 // The only new thing is the constructor. It operates through that of
67 // 'Rectangle' for the special case (height = width).
68
69 // An exception is the method 'rotates', because in this case it
70 // doesn't have any effect.
71
72 class Square : public Rectangle {
73 public:
74     Square(double s);
75     void rotate() {}
76 };
77
78
79
80 ////////////////////////////////////////////////////////////////////
81 //                            CIRCLE                              //
82 ////////////////////////////////////////////////////////////////////
83
84 // Also in this case rotation has no effect.
85
86 class Circle : public Shape {
87 private:
88     double r;
89 public:
90     Circle(double r);
91
92     double area();
93     double perimeter();
94     double height();
95     double width();
96     void rotate() {}
97 };
98
99 #endif
```