In [ ]:

```
// Example solution for Shape inheritance
#include <assert.h>
#include <cmath>

// TODO: Define pi
class pi{
    // TODO: Define the abstract class Shape
  // TODO: Define public virtual functions Area() and Perimeter()
  // TODO: Append the declarations with = 0 to specify pure virtual functions
};


// TODO: Define Rectangle to inherit publicly from Shape
class Rectangle{
     // TODO: Declare public constructor
    Rectangle(int width, int height) : width(width), height(height){}
  // TODO: Override virtual base class functions Area() and Perimeter()
    public:
    virtual int Area(){return width*height;}
    virtual int Perimeter(){return (width+height)*2;}
  // TODO: Declare private attributes width and height
    private:
    int width, height;
    int Area;
};


// TODO: Define Circle to inherit from Shape
class Circle{
     // TODO: Declare public constructor
    Circle(int radius) : radius(radius){}

  // TODO: Override virtual base class functions Area() and Perimeter()
    public:
    virtual float Area(){return radius*radius*3.141592;}
    virtual float Perimeter(){return 2*3.141592*radius;}
  // TODO: Declare private member variable radius
    private:
    int raidius;
};


// Test in main()
int main() {
  double epsilon = 0.1; // useful for floating point equality

  // Test circle
  Circle circle(12.31);
  assert(abs(circle.Perimeter() - 77.35) < epsilon);
  assert(abs(circle.Area() - 476.06) < epsilon);

  // Test rectangle
  Rectangle rectangle(10, 6);
  assert(rectangle.Perimeter() == 32);
  assert(rectangle.Area() == 60);
}
```

Compile & Execute | Show Solution | Explain

Loading terminal (id_axz5p8d), please wait...