New chat

```
class Rectangle : public Shape {
public:
    Rectangle(double w, double h) : width(w), height(h) {}
    double area() const { return width * height; }

private:
    double width;
    double height;
};
```

In this example, the `Rectangle` class derives from `Shape` and provides an implementation of the `area()` function. The implementation simply calculates the area of the rectangle using its width and height.

Here's an example of how you might use these classes:

cpp Copy code

```
int main() {
    // This won't compile, because Shape is an abstract class and can't b
    // Shape s;

    // This works fine, because Rectangle is a concrete class that provid
    Rectangle r(3.0, 4.0);
    std::cout << "The area of the rectangle is " << r.area() << std::endl

    return 0;
}
```

In this example, we try to create an object of type `Shape`, which won't compile because `Shape` is an abstract class. We then create an object of type `Rectangle`, which is a concrete class that provides an implementation of the `area()` function. We can then call the `area()` function on the `Rectangle` object to calculate its area.

I hope this explanation helps you understand how pure virtual functions work in C++