



# Design Patterns

## Module 1

# Proxy Pattern



**To act as a virtual  
proxy**

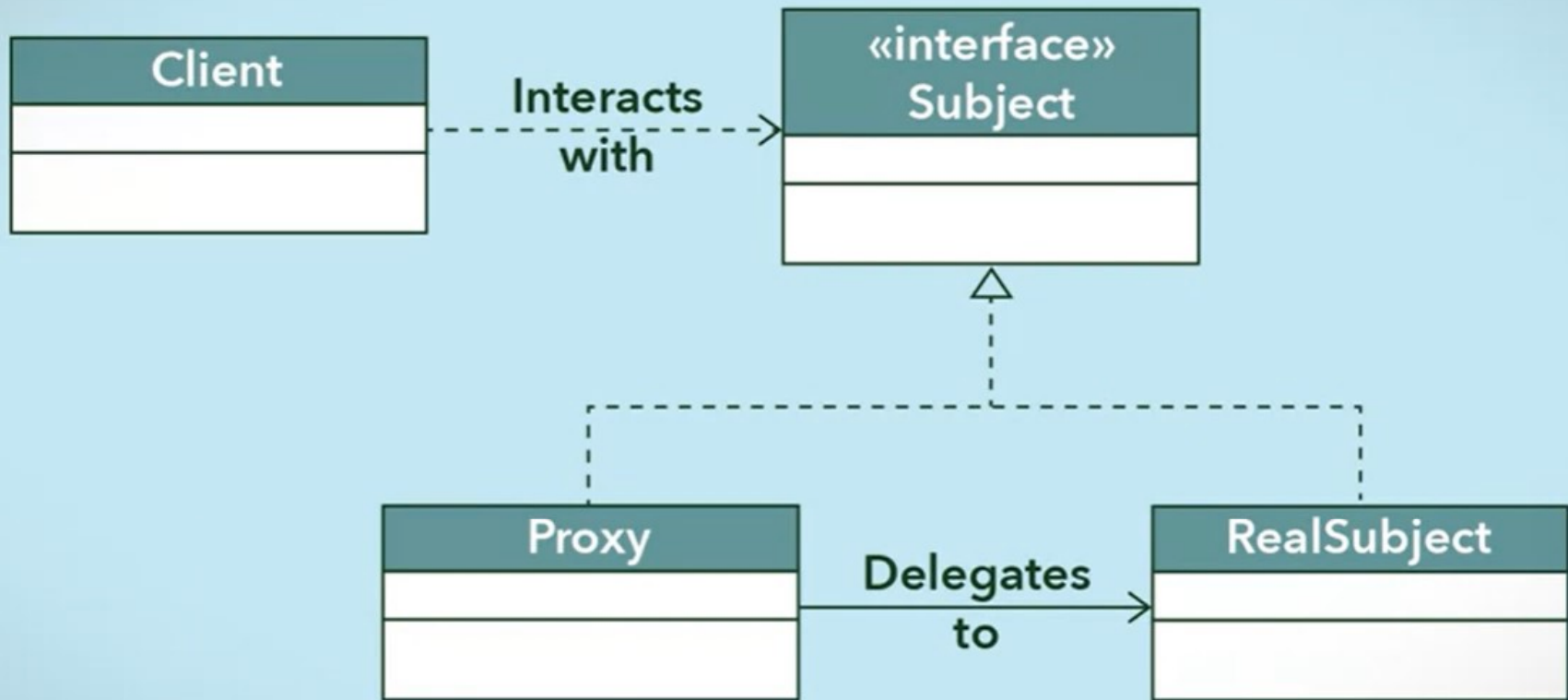


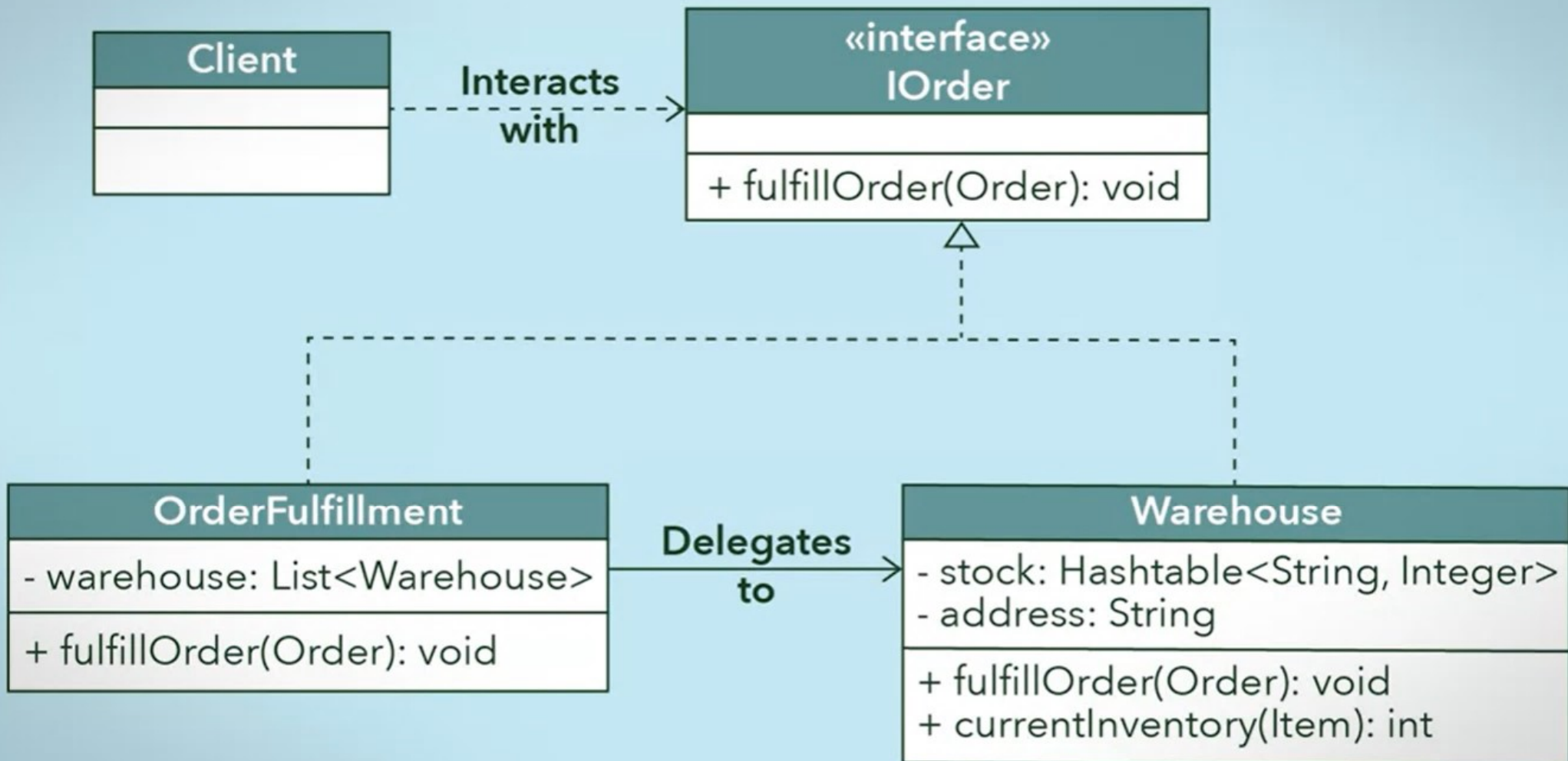
**To act as a protection  
proxy**





**To act as a remote  
proxy**





## Step 1: Design the subject interface

```
public interface IOrder {  
    public void fulfillOrder(Order);  
}
```



## Step 2: Implement the real subject class

```
public class Warehouse implements IOrder {
    private Hashtable<String, Integer> stock;
    private String address;

    /* Constructors and other attributes would go here */
    ...

    public void fulfillOrder(Order order) {
        for (Item item : order.itemList)
            this.stock.replace(item.sku, stock.get(item)-1);

        /* Process the order for shipment and delivery */
        ...
    }

    public int currentInventory(Item item) {
        if (stock.containsKey(item.sku))
            return stock.get(item.sku).intValue();
        return 0;
    }
}
```

```

public class OrderFulfillment implements IOrder {
    private List<Warehouse> warehouses;

    /* Constructors and other attributes would go here */

    public void fulfillOrder(Order order) {
        /* For each item in a customer order, check each warehouse
           to see if it is in stock.

           If it is then create a new Order for that warehouse. Else
           check the next warehouse.

           Send the all the Orders to the warehouse(s) after you finish
           iterating over all the items in the original Order. */
        for (Item item: order.itemList) {
            for (Warehouse warehouse: warehouses) {
                ...
            }
        }
        return;
    }
}

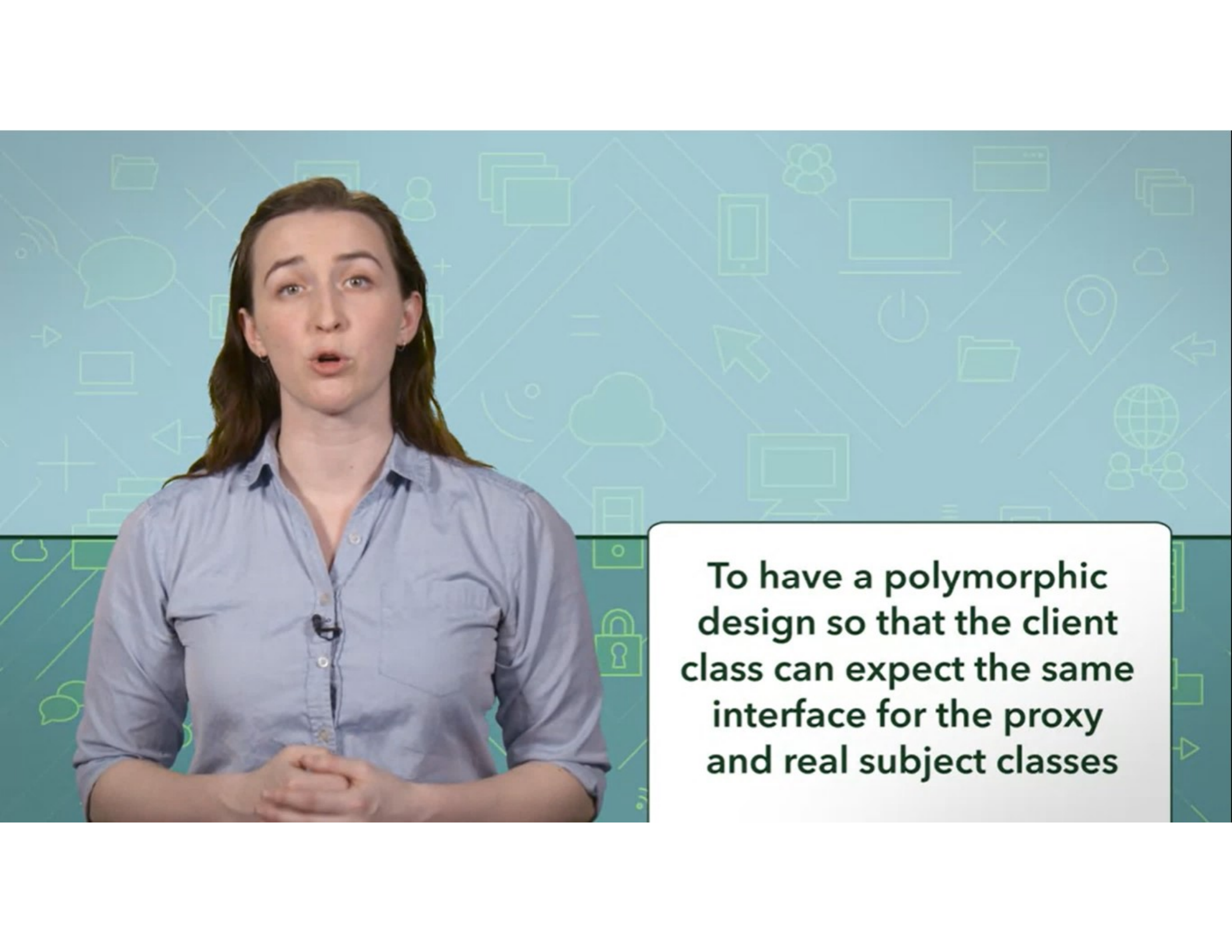
```

**Step 3: Implement  
the proxy class**



**To use the proxy  
class to wrap the real  
subject class**



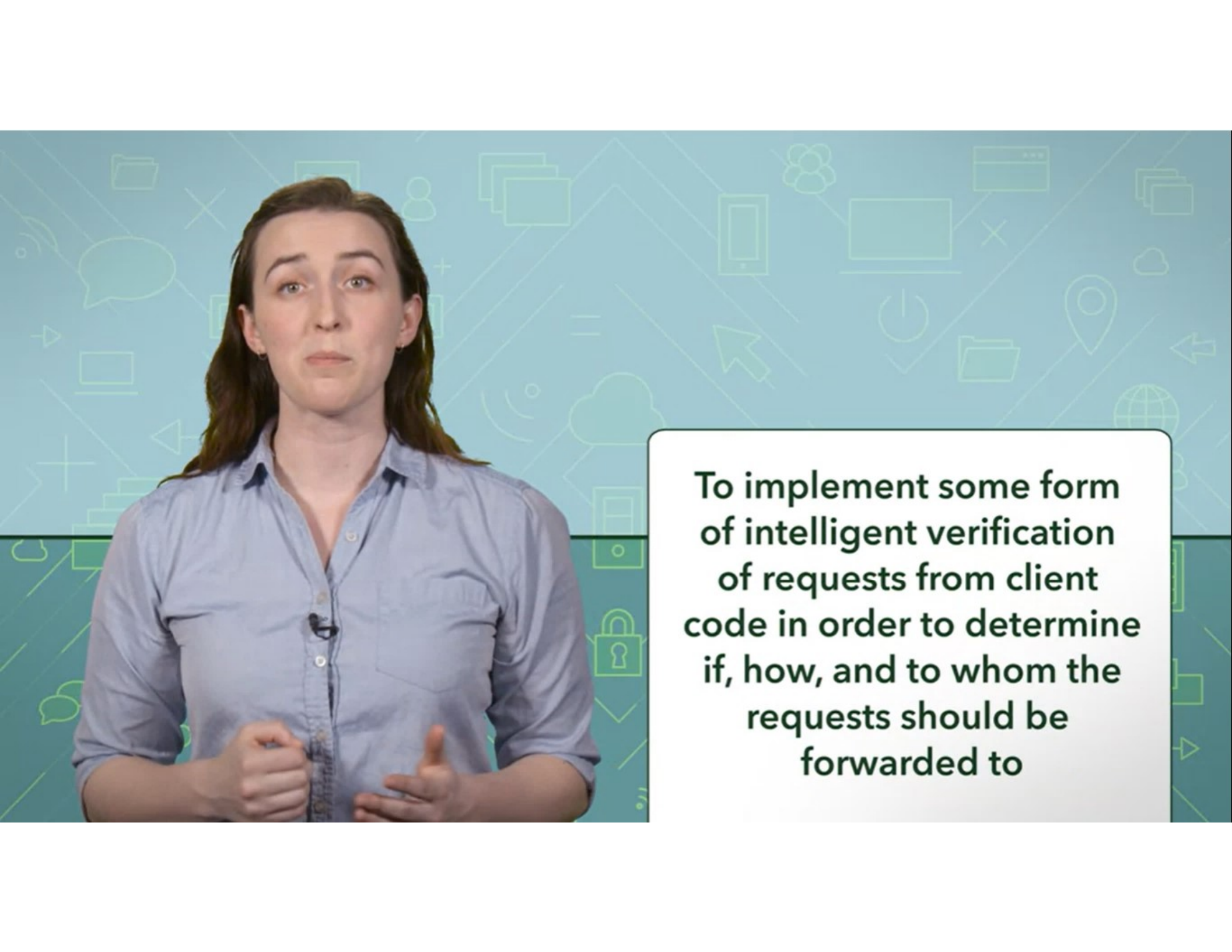


**To have a polymorphic design so that the client class can expect the same interface for the proxy and real subject classes**





**To use a lightweight proxy in place of a resource intensive object until it is actually needed**



To implement some form of intelligent verification of requests from client code in order to determine if, how, and to whom the requests should be forwarded to

