Facade Design Pattern and How it's Different from Proxy and Adapter Design Pattern.

Facade Design Pattern is used when we need to hide the complications of the system from the user.

(Use case 1)

```
EmployeeFacade {
  EmployeeDao employeedao;
public void setRow() {
    employeedao.setRow();
}
 public void getRow(){
   employeedao.getRow();
 }
}.
```

```
EmployeeDAO{
public void setRow();
public void getRow(){
}
  {10-50 methods}
}
```

Only two methods we are exposing to user.

(Use case 2) [Hiding the complexity of the System].

```
public class ProductDao {
  public Product getProduct(int productId){
  }
}
```

```
public class Payment {

  public boolean makePayment(){
  }
}
```

```
public class Invoice

public void generateInvoice(){

}
```

```
public class SendNotification

  public void sendNotification(){


  }
```

order_create

1. getProduct
2. makePayment
3. generateInvoice
4. SendNotification.

User does not want to know all these steps, It just give cash and wants the product

Lets say if tomorrow another step, step 5 gets added, the client side will need to make changes.

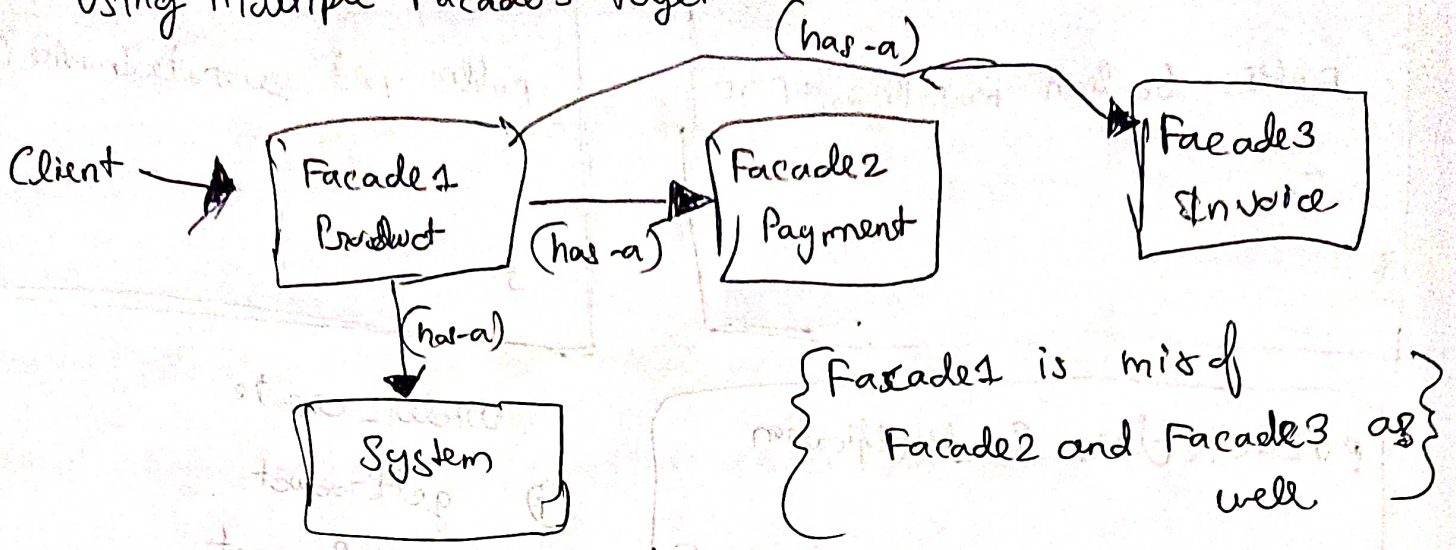
```
public class OrderFacade {
    ProductDao productDao;
    Invoice invoice;
    Payment payment;
    SendNotification sendnotification;

    Constructor();

    public void createOrder(){
        Product product = productDao.getProduct("11");
        payment.makePayment();
        invoice.generateInvoice();
        Sendnotification.SendNotification();
    }
}
```

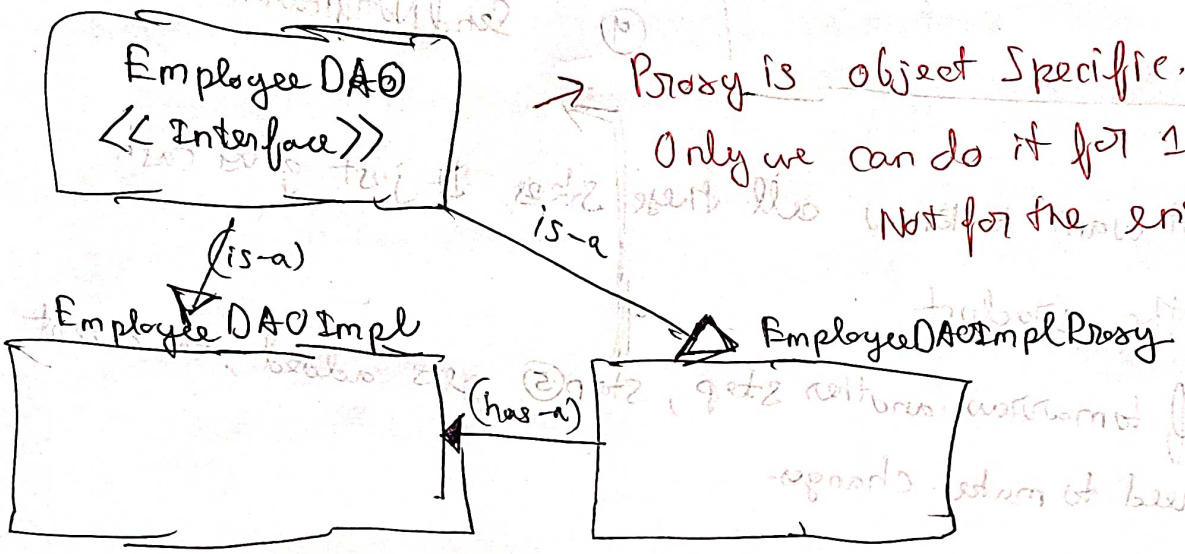
All functionality is hidden here

Using Multiple Facade's together



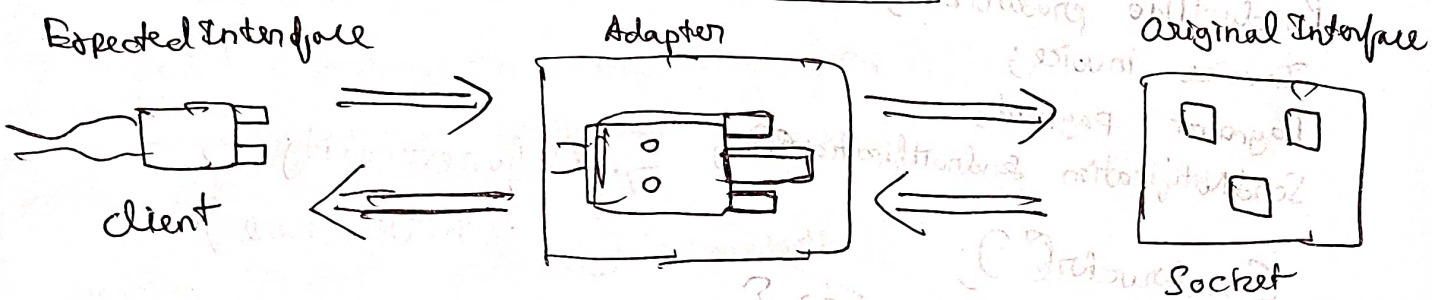
{ Facade1 is mix of Facade2 and Facade3 as well }

Facade V/s Proxy



→ Proxy is object Specific.
Only we can do it for 1 class and Not for the entire system.

Facade V/s Adapter,



Client & System interfaces are incompatible then adapter is used, whereas Facade hide's the complexity