THE BUSINESS EXPERTS

THE FACADE PATTERN

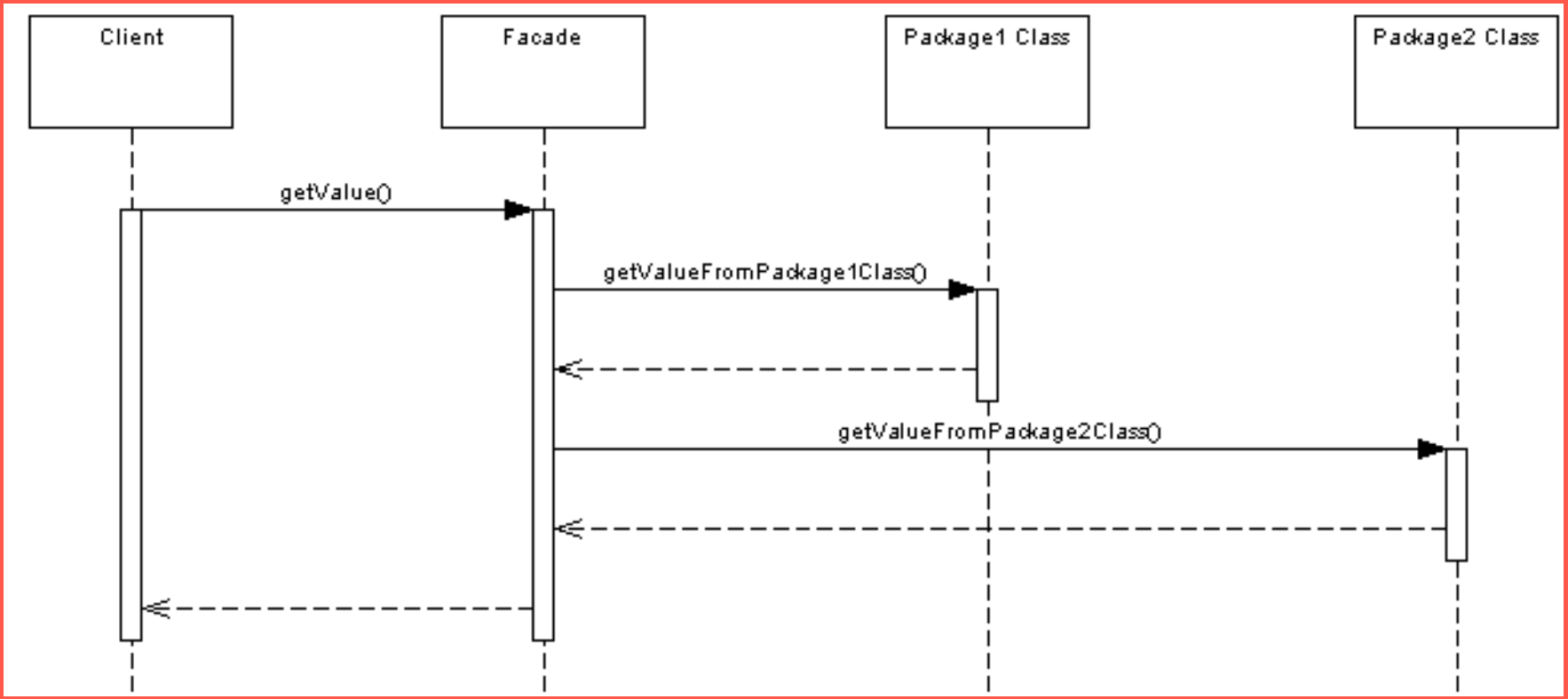
Looking down from on high!

KEY CONCEPTS OF FACADE

Example, Amazon Interface

- *The User of Amazon doesn't need to know how the sub class acts only what the output is, because the "facade" (Amazon) handles it.*
- *In other words basically, Amazon website does not need to know what the mail/stores and payment methods is doing, as long as the mail/store money is handled as expected.*

Facade pattern diagram



Facade pattern in code

```
public class Client{
  public static void main(String[] args){
    TravelFacade facade = new TravelFacade();
    facade.getFlightsAndHotels(from, to);
  }
}


public class TravelFacade{
  private HotelBooker hotelBooker;
  private FlightBooker flightBooker;

  public void getFlightsAndHotels(Date from, Data to){
   ArrayList<Flight> flights = flightBooker.getFlightsFor(from, to);
     ArrayList<Hotel> hotels = hotelBooker.getHotelsFor(from, to); //process and return   }}


public class FlightBooker{  public ArrayList<Flight> getFlightsFor(Date from, Date to)   {      //
returns flights available in the particular date range  }}

public class HotelBooker{  public ArrayList<Hotel> getHotelNamesFor(Date from, Date to)   {      //
returns hotels available in the particular date range  }}
```

Pros and Cons

Pro

The benefit of the Facade pattern is that it provides a simple interface to a complex system without reducing the options provided by the total system. This interface protects the client from an overabundance of options.

Pros and Cons

Con

Watch Out for the Downsides

By introducing the Facade into your code, you will be hardwiring subsystems into the Facade. This is fine if the subsystem never changes, but if it does, your Facade could be broken. Therefore, developers working on the subsystem should be made aware of any Facade around their code.

Pedro ♥ Reece

Sam