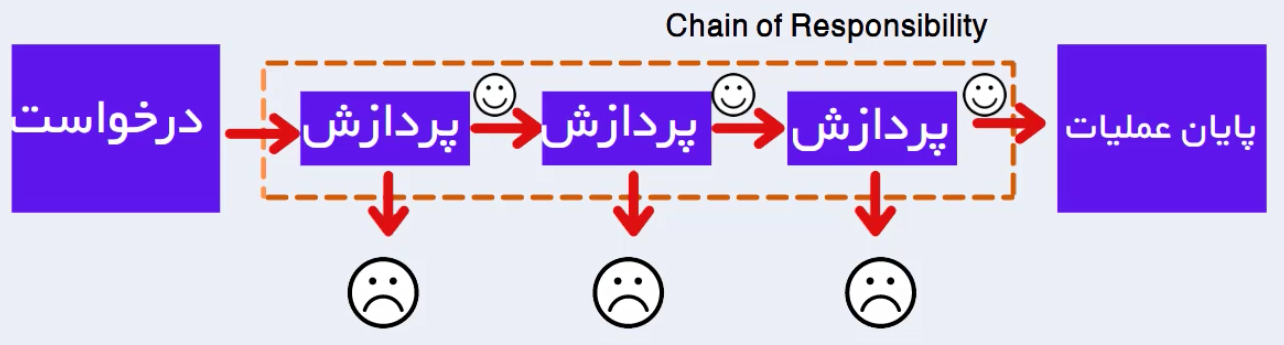
الگوی Chain Of Responsibility یا زنجیره مسئولیت

این الگو درخواست کاربر را در مراحل یا دسته های مختلفی پردازش میکند تا به پایان برسد.



یک در خواست از سمت کاربر ارسال می شود در این Chain Of Responsibility دارای 3 پردازشگر داریم که هرکدام کاری را بر روی درخواست انجام میدهد. در هرکدام از مراحل اگر به خطا برخورد کند درخواست Failed می شود.

برای مثال اسنپ :

* از کلاینت شما مبدا و مقصد را دریافت کرده -ثبت سفارش - ارسال به راننده

این پردازش ممکن است براساس شرایطی در بازار تغییر کند برای مثال کاربر درخواست دهنده کاربری فعال است یا خیر

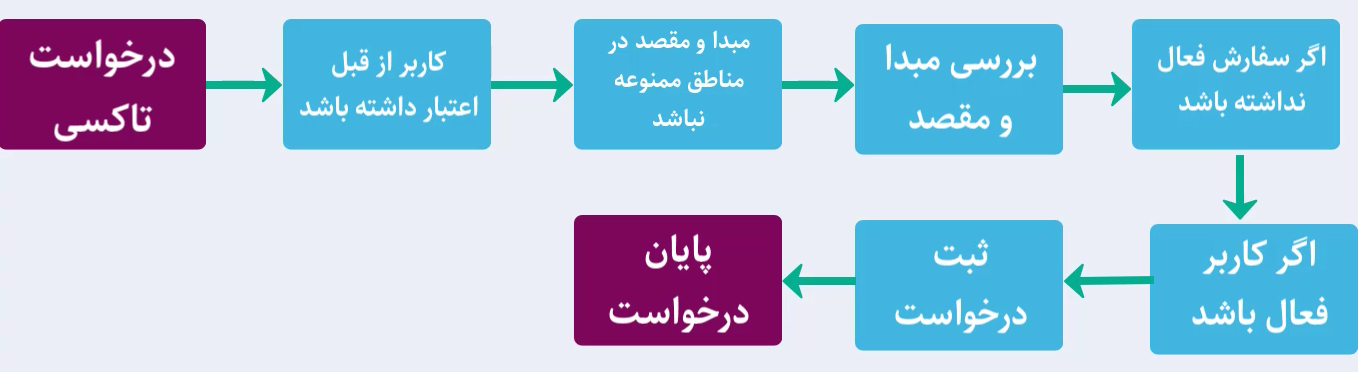
* کاربر فعال است-مبدا و مقصد – ثبت سفارش-ارسال به راننده

یا یا چک کرده محدوده سرویس دهی

* مبدا و مقصد را دریافت کرده -چک کرده محدوده سرویس دهی -ثبت سفارش - ارسال به راننده

یا کاربر سفارش جاری دارد یا خیر

* کاربر سفارش فعال ندارد- مبدا و مقصد را دریافت کرده - ثبت سفارش - ارسال به راننده



نکات

1. استفاده از چندین handler
2. استفاده از interface
3. تضمین 100 درصدی برای به پایان رسیدن درخواست کاربر را نمیدهد.(باید مراقب چیدمان زنجیره باشیم)
4. استفاده از handler پیشفرض

مزایا:

1. به راحتی میتوان ترتیب مراحل انجام یک درخواست را مدیریت کرد
2. جدا سازی کلاس هایی که عملیات را انجام میدهند از کلاس هایی که عملیات را فراخوانی میکنند.
3. کاهش اتصال محکم
4. اصل Open/Close : میتوانیم chain های جدیدی توسعه بدهیم

معایب:

1. اگر چیدمان زنجیره را به درستی رعایت نکنیم برخی از درخواست ها بدونه رسیدگی به پایان میرسند.

چه زمانی از این الگو استفاده کنیم:

1. زمانی که برای رسیدگی به یک درخواست شرایط و مراحل زیادی داشته باشیم
2. زمانی که میخواهید handler ها و یا ترتیب اجرای درخواست ها در زمان اجرا قابل تغییر باشند.