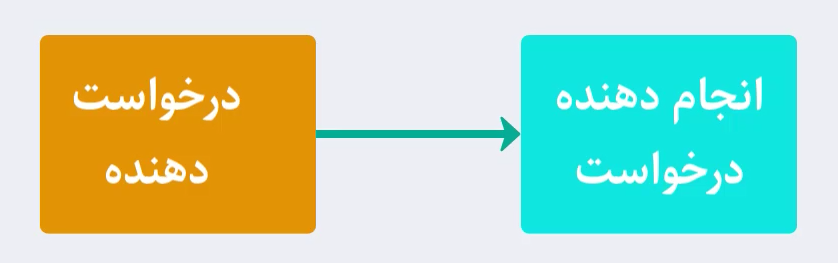
الگوی Command

در نرم افزار های کامپیوتری همیشه یک درخواستی داریم که توسط یک تکه کد، یک متد ، یک کلاس و یا یک ماژول باید توسط بخشی از نرم افزار ما انجام شود و یک تکه کد مسئول انجام دادن این درخواست است.

در نظر بگیرید در حالت عادی هر درخواست یک درخواست دهنده داردکه مستقیما مرحله انجام درخواست میرود:



الگوی command یک فاصله بین این دو می اندازد(یک کلاس). این کلاس درخواست را بصورت یک شی نگه میدارد:



فعالیت های این کلاس میتواند مانند زیر باشد:

* لاگ گیری فعالیت کاربر
* Undo ، Redo
* Script نویسی

نکات:

1. امکان استفاده از چند Receiver برای هر command
2. توانایی های کلاس Command در ساده ترین حالت Command یک واسط است
3. استفاده از واسط انتزاعی برای Receiver
4. ذخیره وضیعت در Command توسط متد execute
5. نگهداری Command ها برای انجام Undo, Redo
6. نگه داری لاگ Command ها بر روی دیسک برای یازیابی Crash نرم افزار
7. با این الگو میتوان Transaction ایجاد کرد(ذخیره دیتا در چند جدول )

مزایا:

1. Invoker و receiver کاملا از هم جدا هستند
2. تعریف فرمان جدید ساده است ( نیازی نیست Command های قبلی را تغییر بدیم،command های جدید تاثیری بر قبلی ها ندارند)
3. مستقل کردن Command از کسی که Command را اجرا میکند
4. پیاده سازی Undo و Redo
5. امکان پیاده سازی deferred execution
6. رعایت اصل SRP
7. رعایت اصل Open/Closed

معایب:

1. پیچیدگی کد

چه زمانی از این الگو استفاده کنیم؟

1. زمانی که میخواهیم درخواست های اجرا شده در اپلیکیشن را نگهداری کنیم.
2. زمانی که میخواهید برنامه قابلیت Undo و Redo اضافه کنیم
3. زمانی که میخواهیم برای command های اجرا شده در اپلیکیشن قابلیت لاگ گیری اضافه کنیم.

الگو های مرتبط با این الگو

* برای نگهداری وضعیت Command ها می توانیم از الگوی Memento استفاده کنیم.
* برای نمونه سازی از Command ها می توانیم از ProtoType استفاده کنیم