دانشکده فنی و مهندسی
کارشناسی ارشد مهندسی کامپیوتر نرم‌افزار
گروه مهندسی کامپیوتر و فناوری اطلاعات

معماری نرم افزار

موضوع :

پیاده سازی الگوی طراحی chain of responsibility

در جاوا اسکریپت

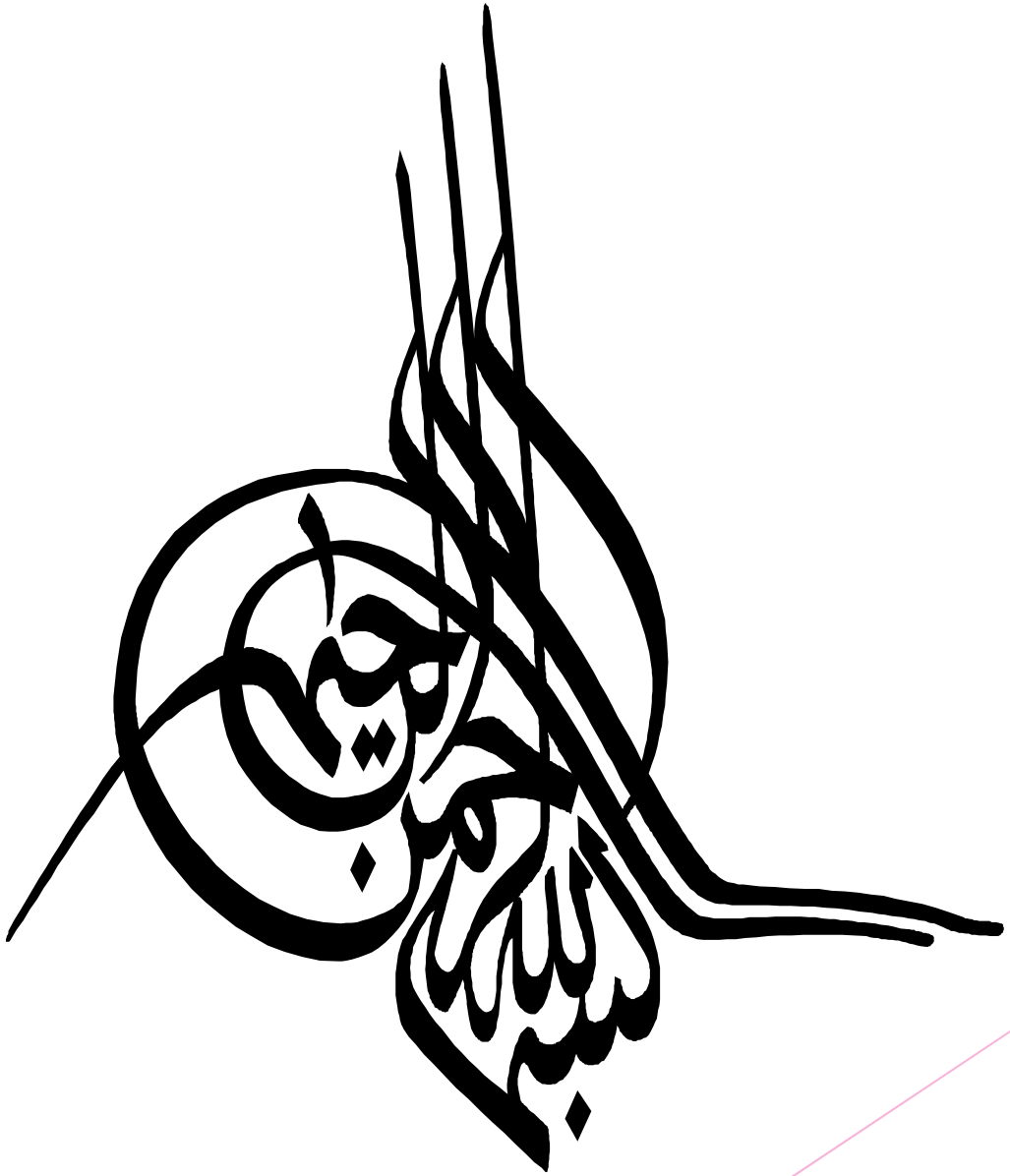
دانشجو::

امیر حسین روشن مهر

استاد راهنما:

دکترسید علی ابراهیمی رضوی

دی۱۳۹۹


بسم الله الرحمن الرحيم

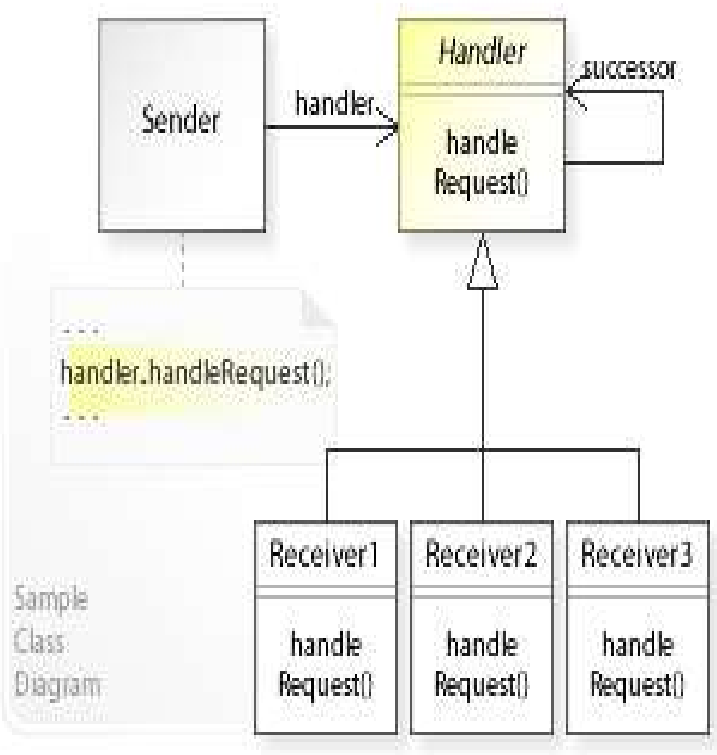
الگوی طراحی
chain of responsibility
در جاوا اسکریپت

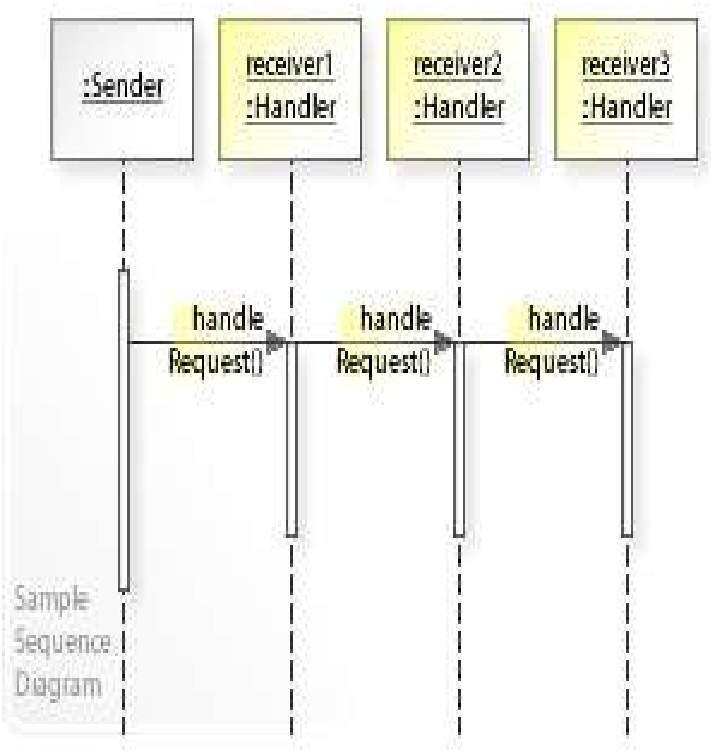
- **تا به حال پیش آمده به دستگاه عابر بانک مراجعه کرده باشید و بعد از انجام یک کار دستگاه ATMاز شما سوال کند که آیا درخواست دیگری برای انجام دارید ؟ اگر این اتفاق برای شما افتاده بدون آنکه خود شما بدانید از الگوی chain of responsibilityاستفاده کردید در ادامه به طور مفصل توضیح خواهم داد که این الگو چیست و چه ارتباطی با دستگاه ATMدارد و چگونه در زبان جاوا اسکریپت آن را پیدا سازی کنیم.**

- **الگوی** chain of responsibility**راهی برای عبور دادن درخواست به اشیا بصورت زنجیر وار است به شکلی که سه عنصر در این سناریو شرکت دارند :**

- **فرستند** sender
- **گیرنده** reciever
- **درخواست** request
- **که روند سناریو به شکل زیر میباشد**

الگوی طراحی chaining





- همانطور که در شکل بالا میبینید sender فقط از اولین شی موجود در زنجیر اطلاع داره پس درخواست رو برای اولین شی موجود در زنجیر میفرسته
-
- اولین شی زنجیر هم فقط از شی بعدی اطلاع داره و میتونه درخواست رو برای نفر بعدی بفرسته و این روند اونقدر ادامه پیدا میکنه تا وقتی که یک شی درخواست رو برای شی بعدی نفرسته
-
- به عبارت دیگر chain of responsibility به همه اشیای موجود در زنجیر شانس پردازش request رو میدهد و با این کار وابستگی coupling بین sender و یک گیرنده خاص را کم میکند
-

نحوه پیاده سازی الگوی طراحی
chain of responsibility
در جاو اسکریپت

- **حالا که تعریف الگو رو توضیح دادم به مثال دستگاه عابر بانک برمیگیردم، فرض میکنیم دستگاه عابر بانک فقط واحد های ۲، ۵، ۱۰ و ۵۰ هزار تومانی رو به مراجعه کننده میده و از طرفی ما به ۲۷ هزار تومان احتیاج داریم.**
- **در این مثال دستگاه عابر بانک یک** sender **و سیستم دریافت پول متناسب با واحد های پولی ذکر شده** reciever **های موجود در یک زنجیر و در خواست ما برای ۲۷ هزار تومان پول به منزله** request **در این مثال میباشد.**
- **برای پیاده سازی آن من یک کلاس به نام** Request **در نظر میگیرم که یک مقدار داده به نام** amount **دارد که میزان پولی است که ما احتیاج داریم و همچنین یک متد به نام** get**دارد که وظیفه پول دادن به مراجعه کننده بر اساس نوع نیاز (۱۰–۵–۲) را دارد و نقش** reciever **را ایفا میکند.**

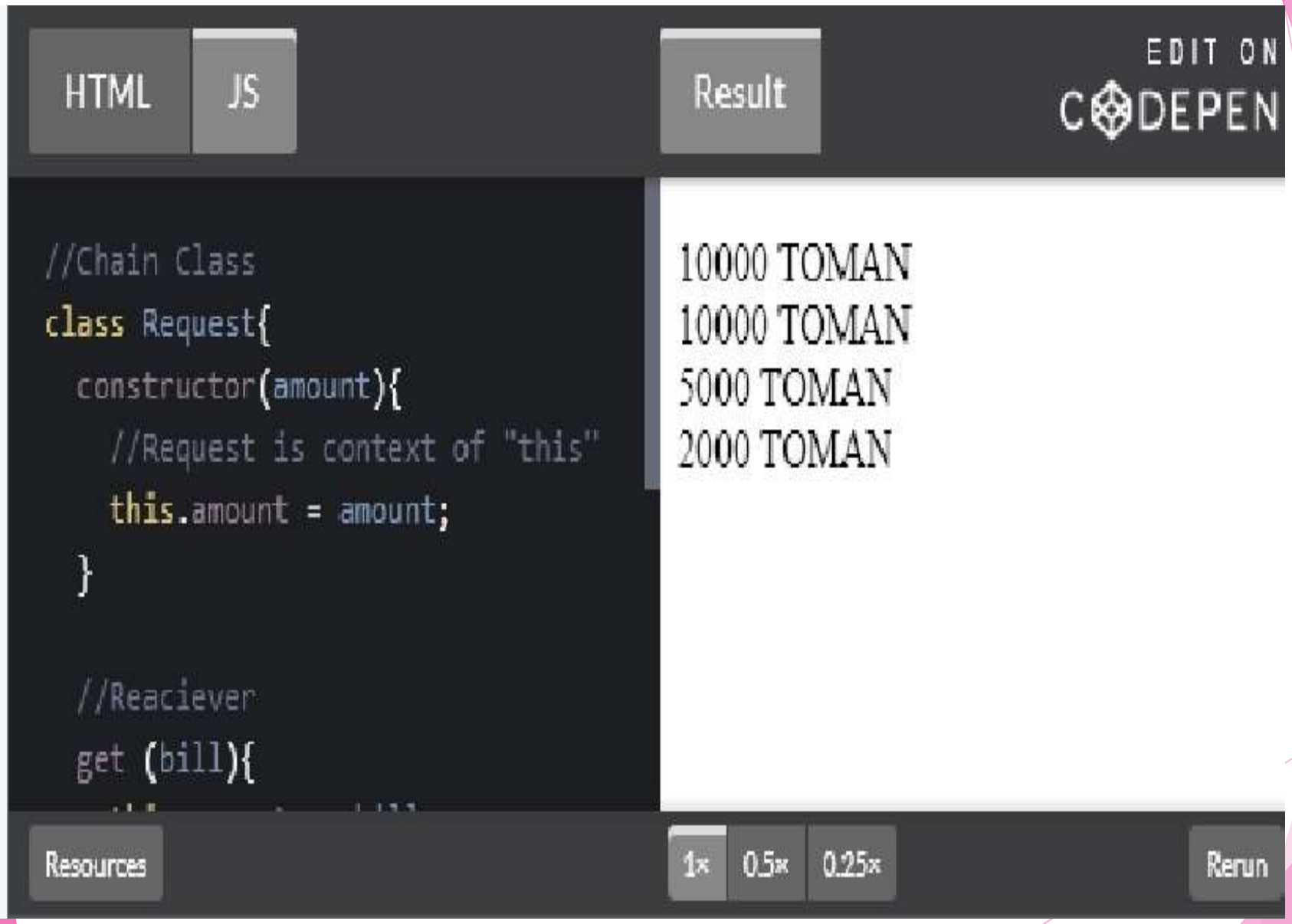
```
//Chain Class
class Request{
  constructor(amount){
    //Request is context of "this"
    this.amount = amount;
  }


  //Reaciever
  get (bill){
    this.amount -= bill;
    document.write(bill + '000 TOMAN
')
    return this;
  }
}

//Sender
let ATM = new Request(27);
ATM.get(10).get(10).get(5).get(2);
```

- **حالا با ساختن شی از این کلاس** Requestو **تعیین میزان پولی که ما نیاز داریم یک شی از آن میسازیم و داخل متغیر** ATMمیریزیم، **و برای دریافت ۲۷ هزار تومان پول باید ۴ بار درخواست پول بدیم دو تا ۱۰ هزار تومان، یک ۵ هزار تومان و یک ۲ هزار تومان و این کار را از طریق** ATM.get(10).get(10).get(5).get(2); **انجام میدهیم**

- **در ابتدا درخواست دریافت پول از طریق** senderبه **اولین** recieverیعنی get(10) **ارسال میشه به عبارت دیگه کاربر از دستگاه میخواد که ۱۰ هزار تومن پول به او بدهد، وقتی دستگاه پول را داد از کاربر میپرسه که درخواست دیگه ای برای انجام داری؟ این یعنی درخواست به** recieverبعدی **فرستاده شده و این کار اونقدر ادامه پیدا میکنه که دیگه کسی درخواست دریافت پول رو مورد پردازش قرار نده**

- **نکته بسیار مهم این است که هر بار که درخواستی برای** reciever **یعنی متد** getارسال **میشود از طریق** return thisدرخواست **را به** recieverبعدی **فرستاده میشود تا اگر خواست پردازش و اگر نه به نفر بعدی ارسال کند**

- **و در نهایت نتیجه اجرای کد بالا را در زیر میتوانید مشاهده کنید که دریافت ۲۷ هزار تومان پول میباشد و این کار از طریق ارسال درخواست برای اشیای دیگر در یک زنجیر محقق شد**


HTML
JS
Result
EDIT ON
CODEPEN
//Chain Class
class Request{
constructor(amount){
//Request is context of "this"
this.amount = amount;
}
//Reaciever
get (bill){
10000 TOMAN
10000 TOMAN
5000 TOMAN
2000 TOMAN
Resources
1×
0.5×
0.25×
Rerun

مهمترین کاربرد استفاده از chain of responsibility pattern کم کردن وابستگی بین sender و یک requestو کم کردن خط کد میباشد به عنوان مثال در دستگاه عابر بانک در صورت عدم استفاده از این الگو باید به شکل زیر عمل میکردیم

```
ATM.get(10);
ATM.get(10);
ATM.get(5);
ATM.get(2);
```

پایان