الگوهای طراحی

DESIGN PATTERNS

مقدمه

هدف اصلی مهندسی نرم افزار: بهبود قابلیت استفاده مجدد

کسی وجود دارد که قبلاً مسئله شما را حل کرده است .

مروری بر الگو

- ایجاد الگوهایی برای تسهیل و تسریع فرایند طراحی نرم افزار.
- ایجاد ساختارها و روش ها با انتزاع بالا
- استفاده مجدد از طراحی ها (طراحی نکردن یک راه حل از ابتدا)
- سازگاری با نیازمندی های مخصوص نرم افزارهای مختلف

الگوهای طراحی و شی گرایی

- تاکید شی گرایی بر استفاده از نمادها
- تمرکز شی گرایی بر رسم نمودارها
- بیشترین استفاده مجدد درهنگام طراحی

استفاده از الگوها = ارتقای سطوح شی گرایی(تجرید - محصورسازی- واحد بندی- سلسله مراتب)

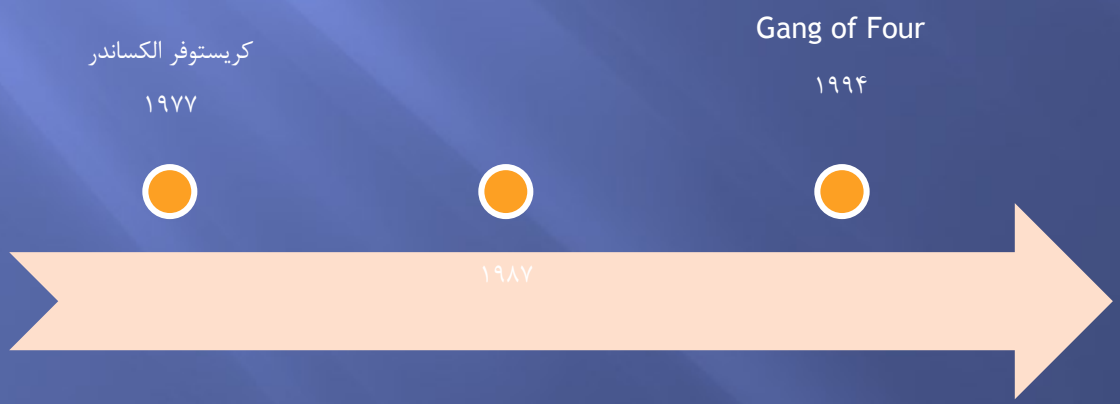
طبقه بندی الگوها (GoF)

استاندارد الگوها برگرفته از GoF (Gang of Four)

" Design Patterns : Elements of Reusable Object Oriented Software

ایده استفاده از الگوها در طراحی نرم افزار: ایجاد یک فرمت استاندارد برای مستندسازی الگوها

دسته بندی ۲۳ نوع از الگوها



طبقه بندی الگوها (ادامه)

الگوهای آفرینشی (Creational Pattern)

مورد استفاده فرآیندهای ایجاد اشیاء

الگوهای ساختاری (Structural Pattern)

مورد استفاده در تر کیب اشیاء و کلاسها

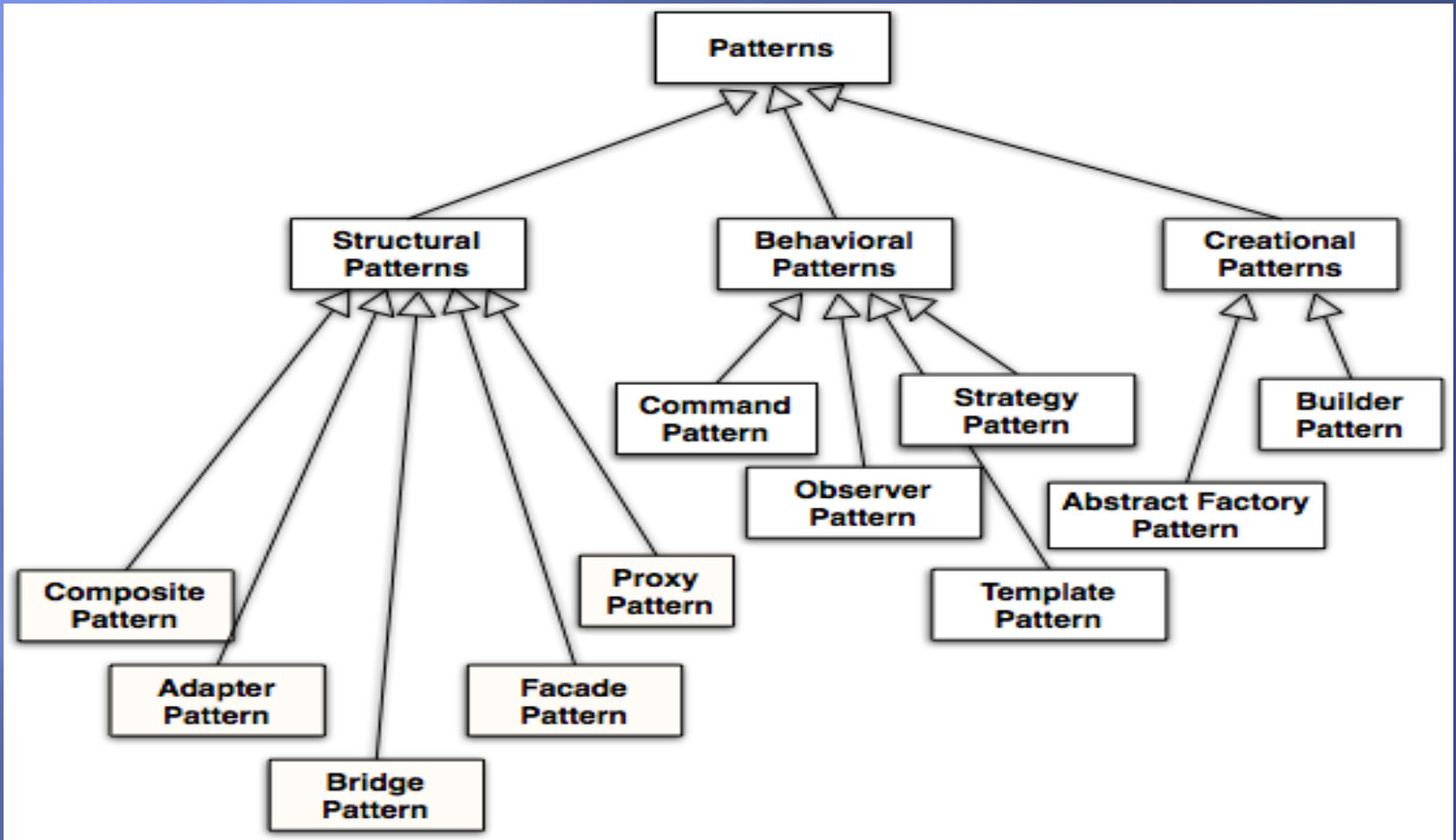
الگوهای رفتاری (Behavioral Pattern)

تمرکز روی ارتباط اشیاء با یکدیگر و نحوه توزیع مسئولیت میان آنها

طبقه بندی الگوها (ادامه)

Behavioral	Structural	Creational
Chain of Responsibility	Adapter	Abstract Factory
Command	Bridge	Builder
Interpreter	Composite	Factory Method
Iterator	Decorator	Prototype
Mediator	Façade	Singleton
Memento	Flyweight	
Observer	Proxy	
State		
Strategy		
Template Method		
Visitor		

طبقه بندی الگوها (ادامه)

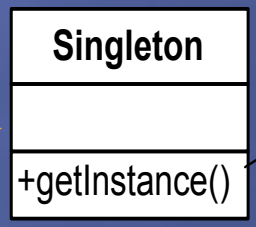


شناسائی الگو

- تعریف مسئله
- شناسایی و بررسی زمینه (context)، سابقه و راه حل های مسئله
- تعیین بهترین راه حل از بین راه حل های موجود

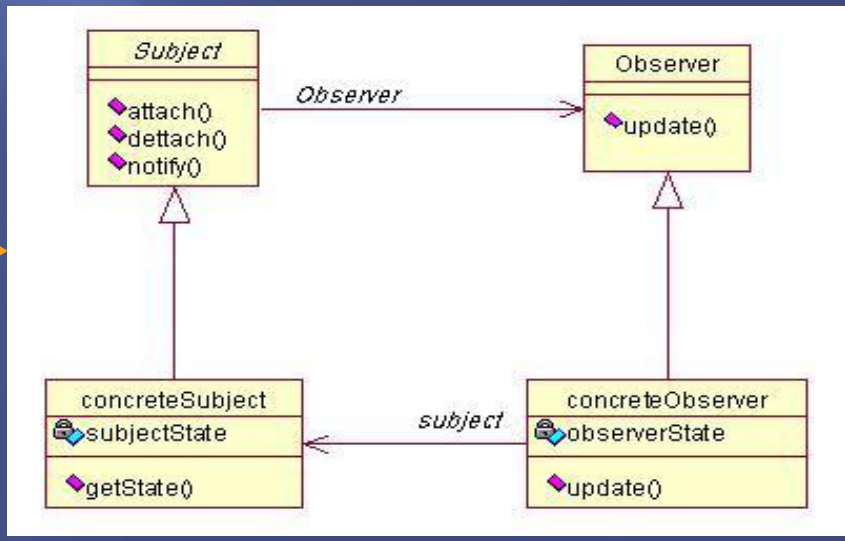
شناسائی الگو (ادامه)

چگونه میتوان یک کلاس تک و یگانه ایجاد نمود؟




```
If (this ==null)
        this= new Singleton();
```

چگونه میتوان اشیا وابسته به یک کلاس را از تغییرات آن آگاه نمود؟



الگوی منفرد (Singleton) Creational /

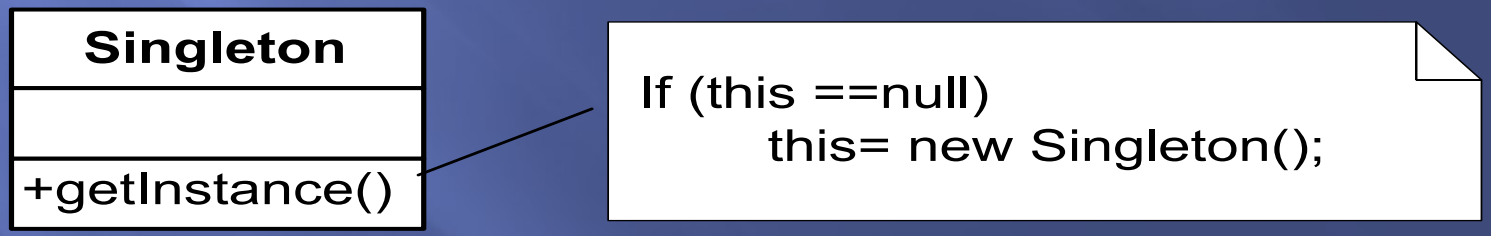
زمینه:

اطمینان می دهد که تنها یک نمونه (Instance) از یک کلاس وجود دارد.

راه حل:

یک سازنده خصوصی (Private Constructor) وجود دارد که اطمینان می دهد هیچ کلاس دیگری قادر به ایجاد مجدد (Recreate) یک نمونه دیگر نیست.

مثال : شرکت اطلس فقط میتواند یک مدیرعامل داشته باشد.



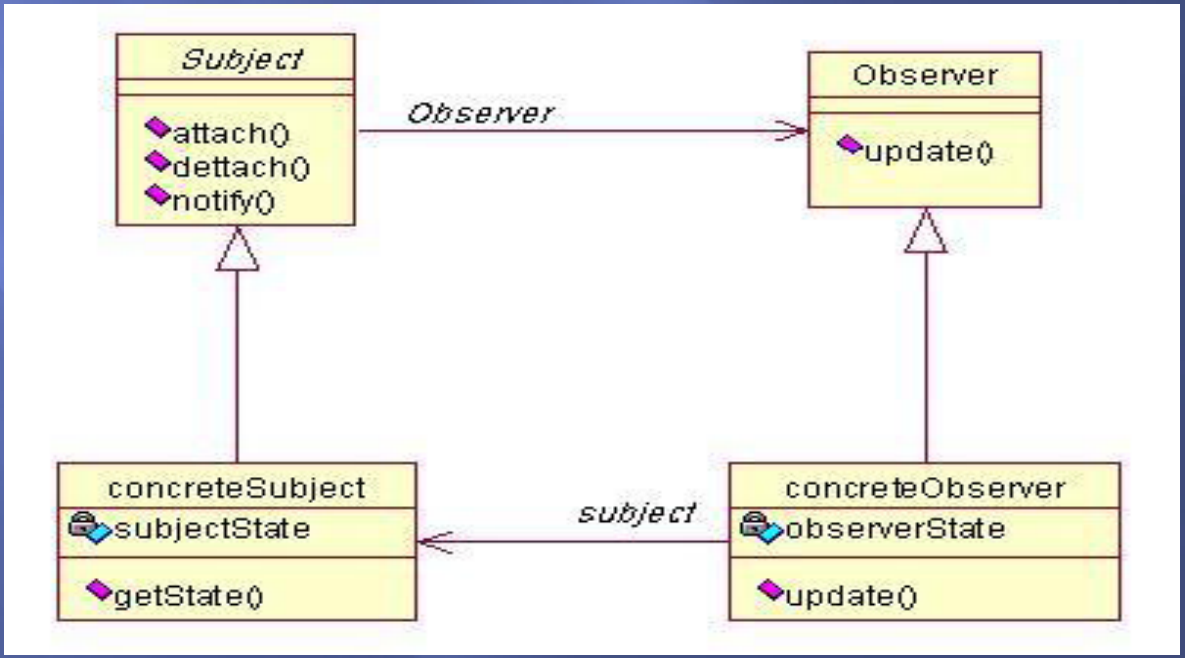
Behavioral / (Observer) الگوی بیننده

زمینه :

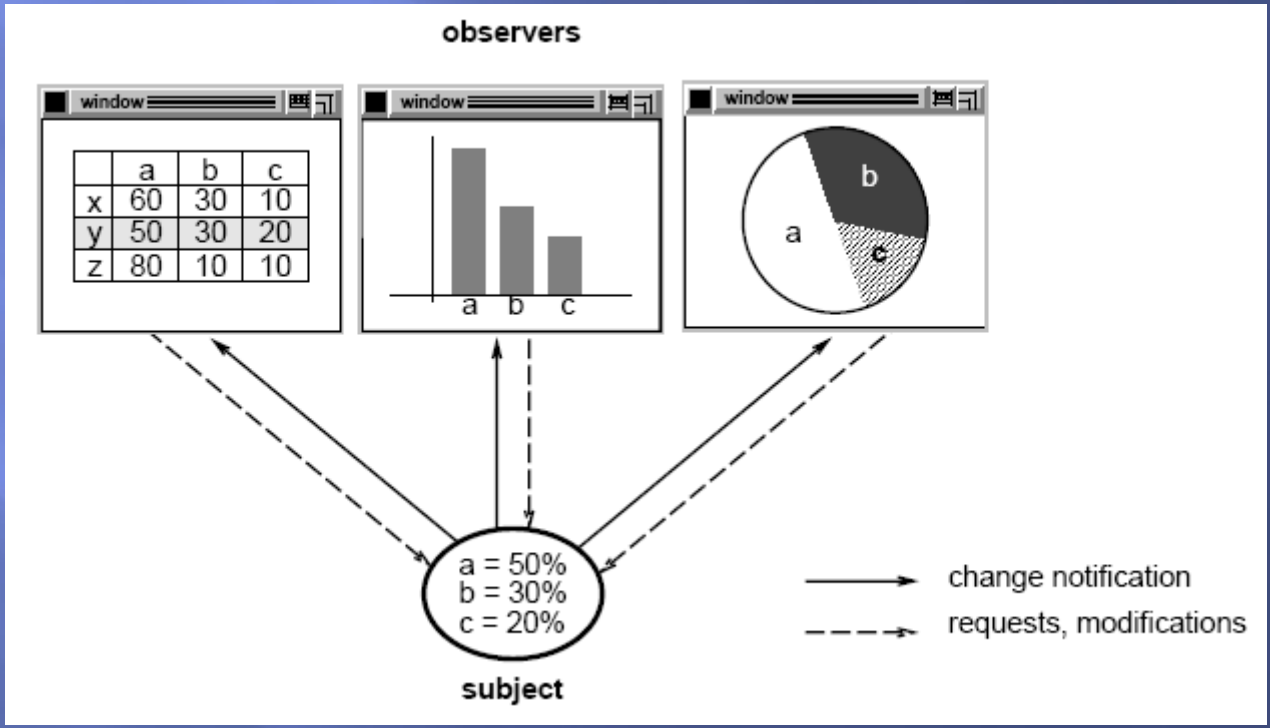
تعدادی شی به یک شی دیگر وابستگی یک به چند دارند. اعمال بروزرسانی در اشیاء وابسته درهنگام تغییر یک شی

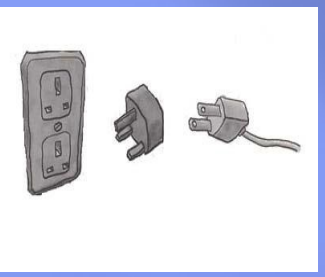
راه حل:

ایجاد یک شی بنام subject که مجموعه ای از اشیاء به آن وابسته هستند . هر زمانی که وضعیت شی مورد نظر تغییر می کند اشیاء دیگر(observer) از آن تغییر آگاه می شوند و بروزرسانی می شوند.



الگوی بیننده (ادامه)





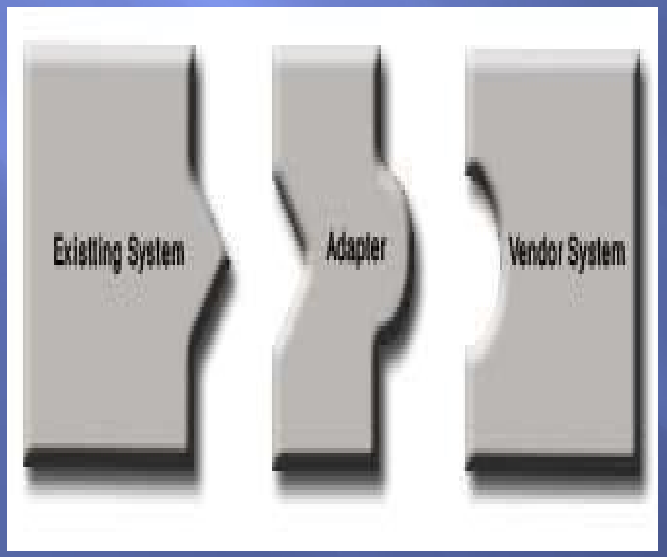
الگوی تطبیق گر (adapter) / structural

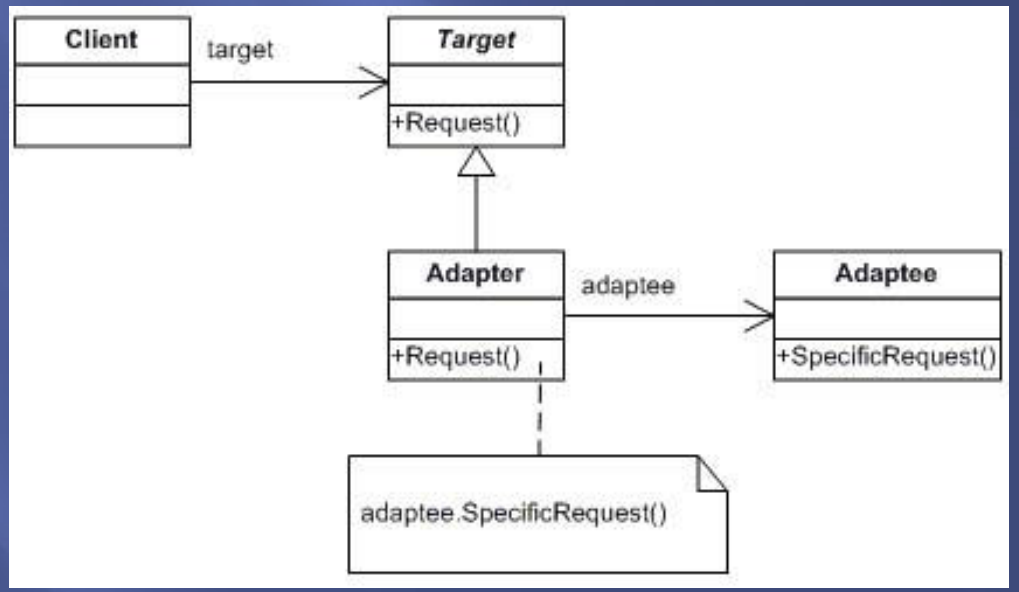
زمینه :

تفاوت اینترفیس های کلاسهای جدید و کلاسهای موجود در سیستم

راه حل :

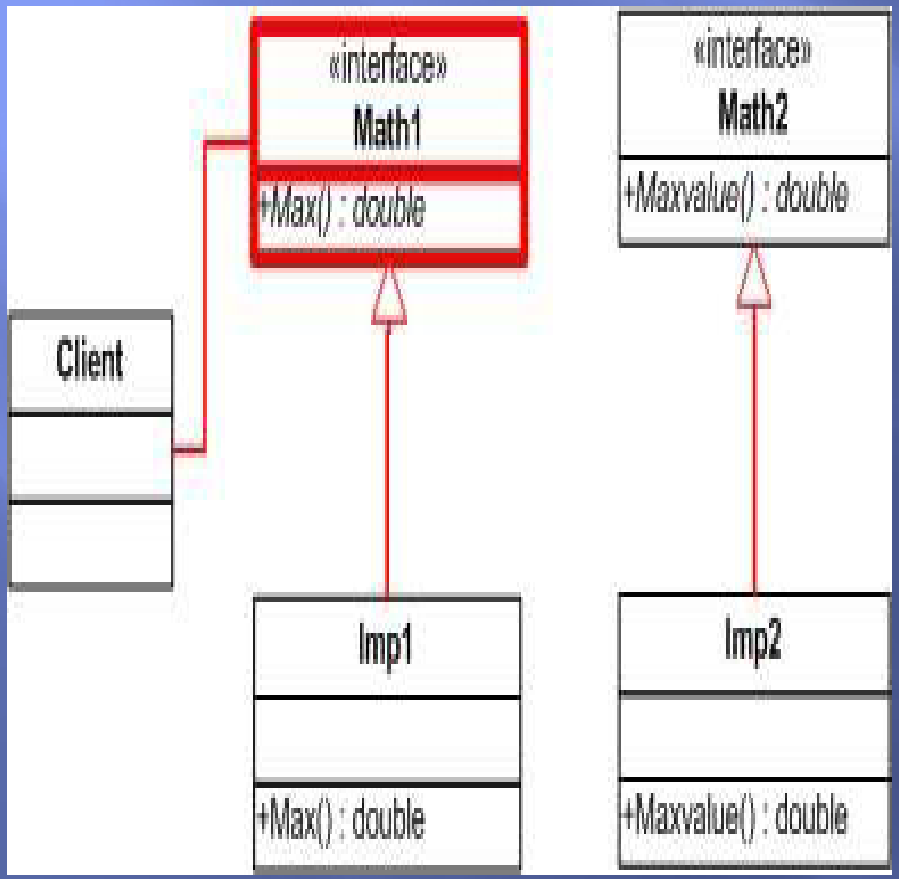
تعریف یک کلاس جدید(Adapter) بطوری که اینترفیسی را کلاس کلاینت می شناسد پیاده سازی می کند.

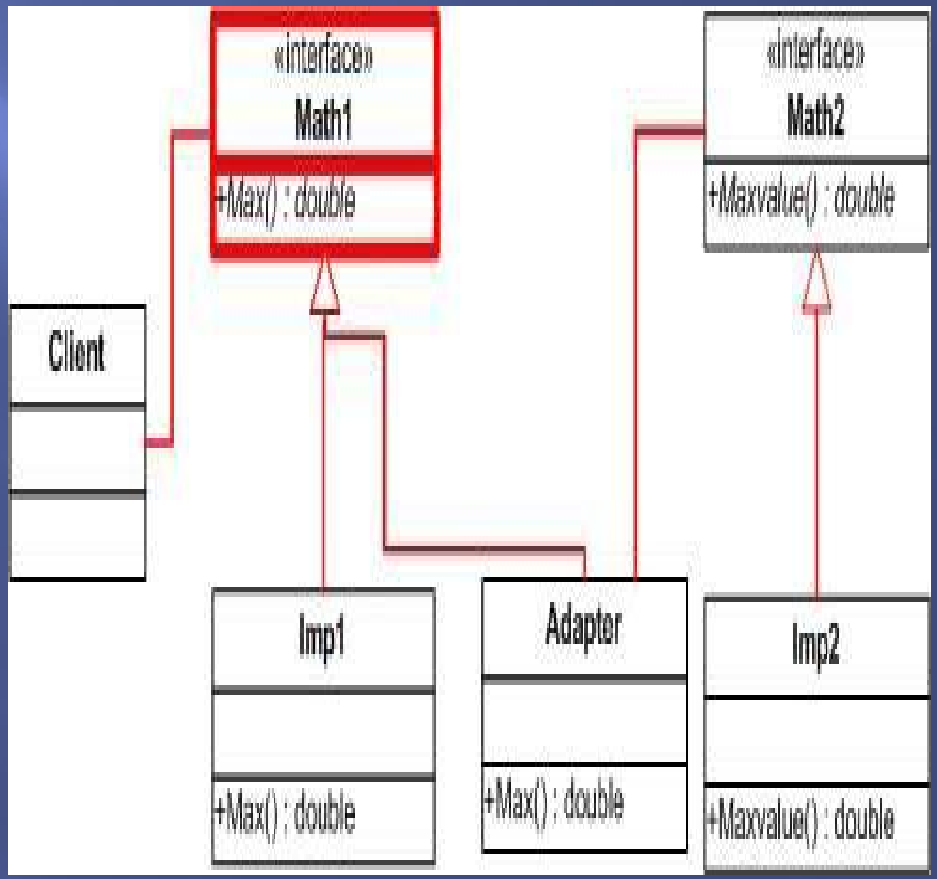




الگوی تطبیق گر (ادامه)

فرض کنید که شما یک سیستم نرم افزاری دارید که نیاز دارد با کتابخانه ی از کلاس های جدید که مربوط به فروشندگان است کار کند، اما کلاس های جدید به گونه ای طراحی شده اند که واسط متفاوتی را نسبت به واسط قبلی دارند .





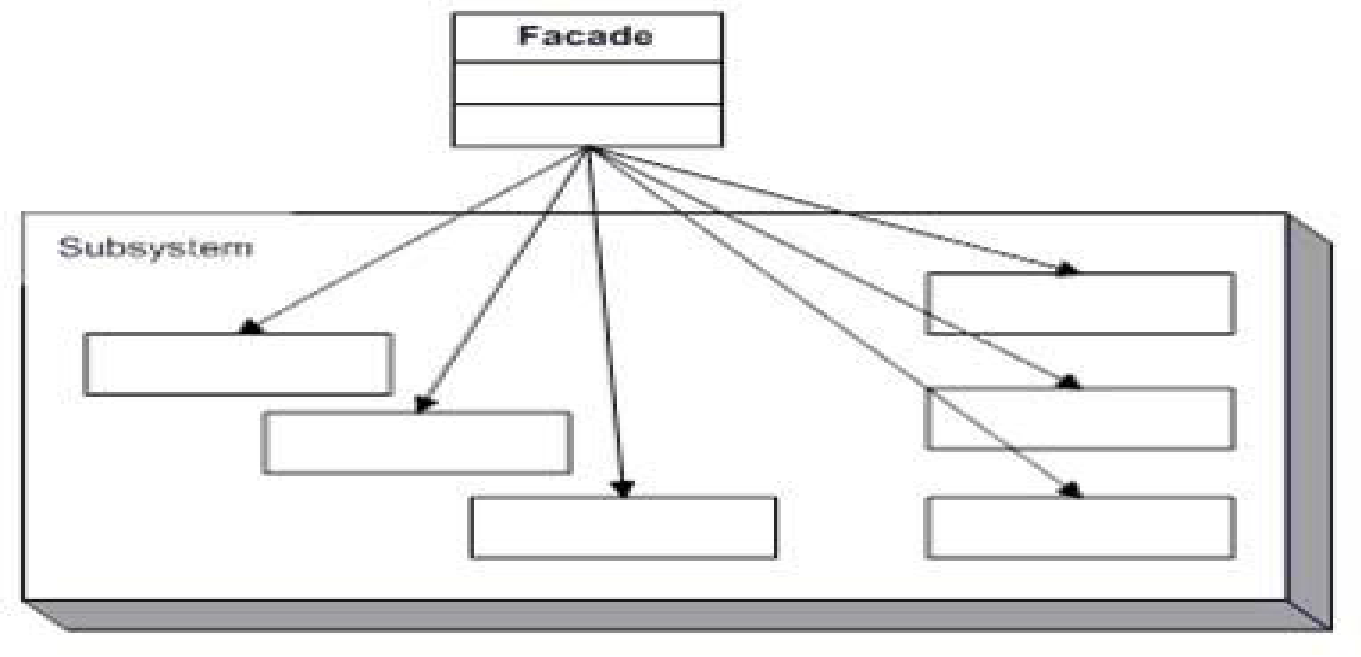
Structural / (Facade) الگوی دریچه

زمینه :

توسعه دهندگان بخواهند پیچیدگیهای سیستم را در یک یا چند کلاس (واسط) مخفی کنند.

راه حل :

ایجاد کلاس facade، که واسطی را به زیر سیستم ایجاد می کند و این کلاس به عنوان نماینده زیرسیستم مورد نظر با زیر سیستم های دیگر عمل می کند.



Behavioral / (Strategy) الگوی استراتژی

زمینه :

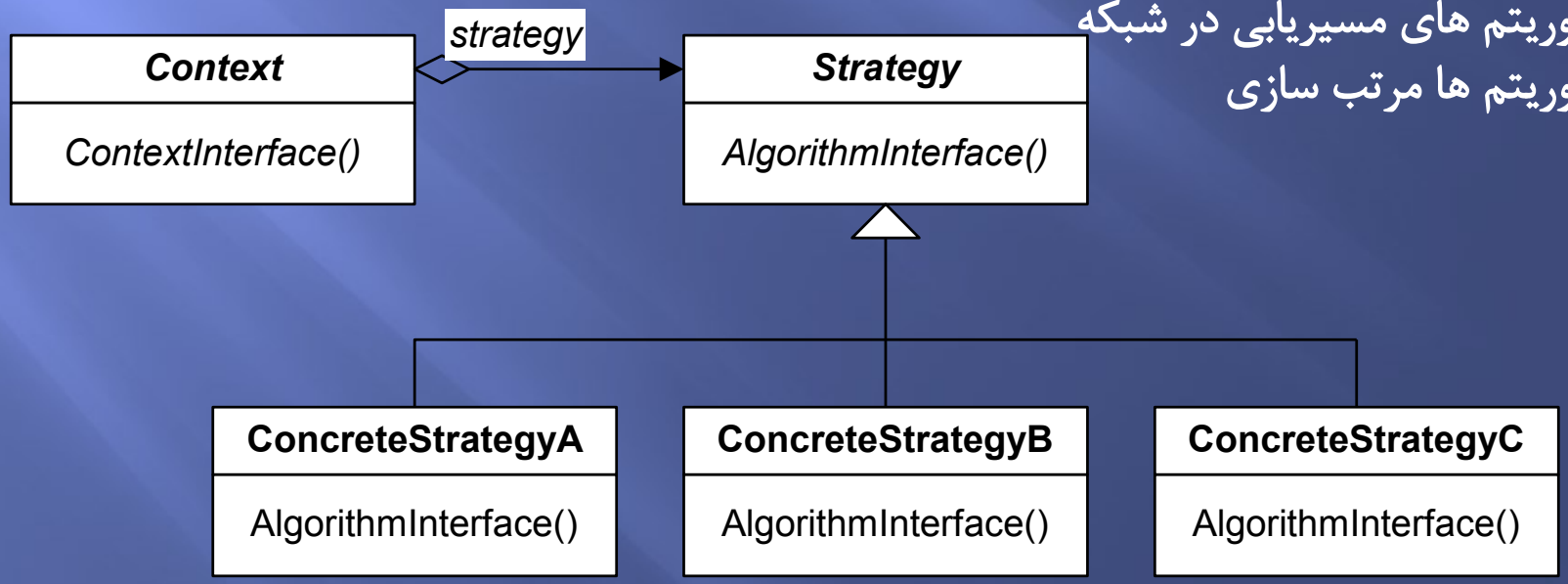
وجود چندین الگوریتم یا راه حل مختلف برای مسئله

راه حل :

به تعداد استراتژی ها کلاس **concrete strategy** در نظر گرفته و ویژگیهای مشترک آنها را در کلاس استراتژی قرار داده و رابطه ارث بری ایجاد می گردد.

مثال :

- الگوریتم های مسیریابی در شبکه
- الگوریتم ها مرتب سازی



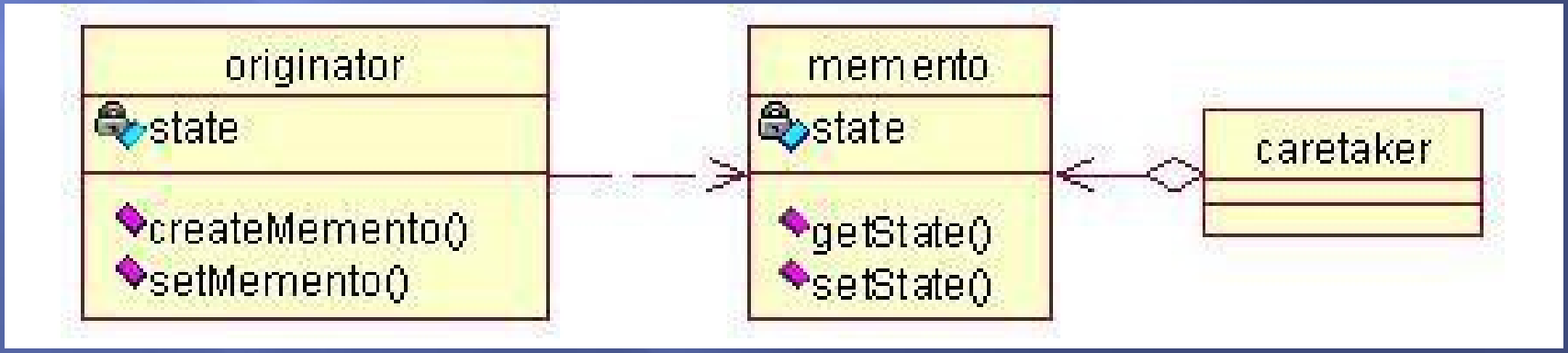
Behavioral :Memento الگوي

زمینه :

ذخیره حالت موجود اشیاء و سپس بازیابی آن حالت در صورت نیاز

راه حل :

زمانی که برنامه درخواست ذخیره کردن از شی originator میکند این شی تمام صفات مورد نیاز جهت بازیابی را در شی memento ذخیره می کند. تمام اشیا memento در کلاس caretaker قرار می گیرند.



Behavioral :Momento الگوي

مثال :
بازی کامپیوتری
خرید ناتمام اینترنتی
مشابه clipboard برنامه word

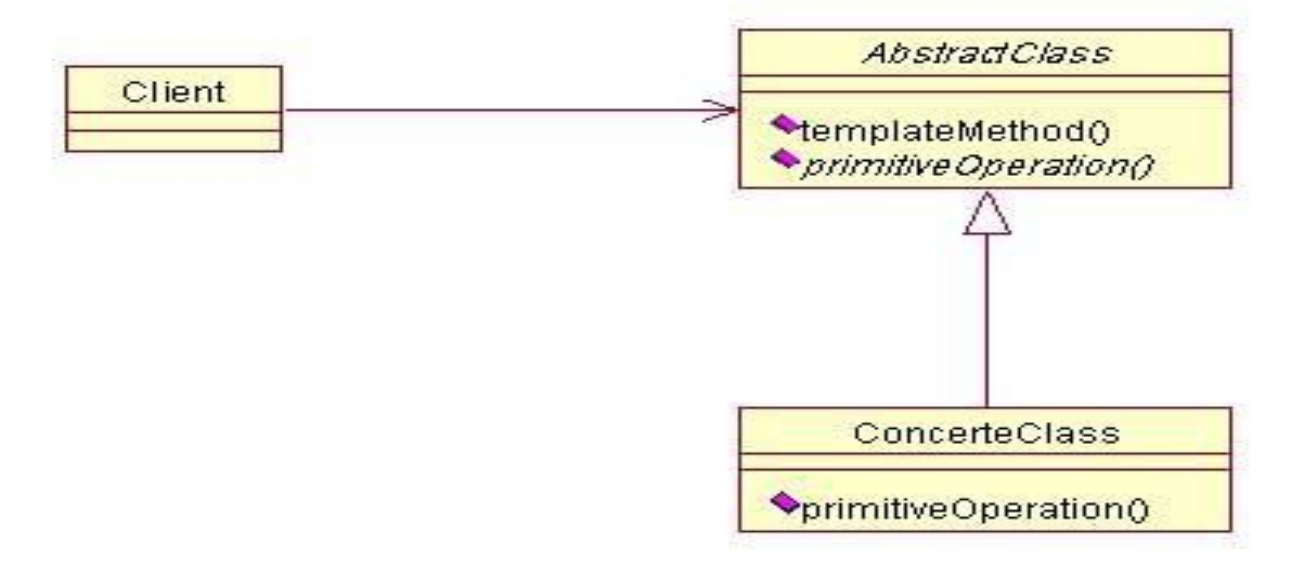
Behavioral:Template method الگوي

زمینه :

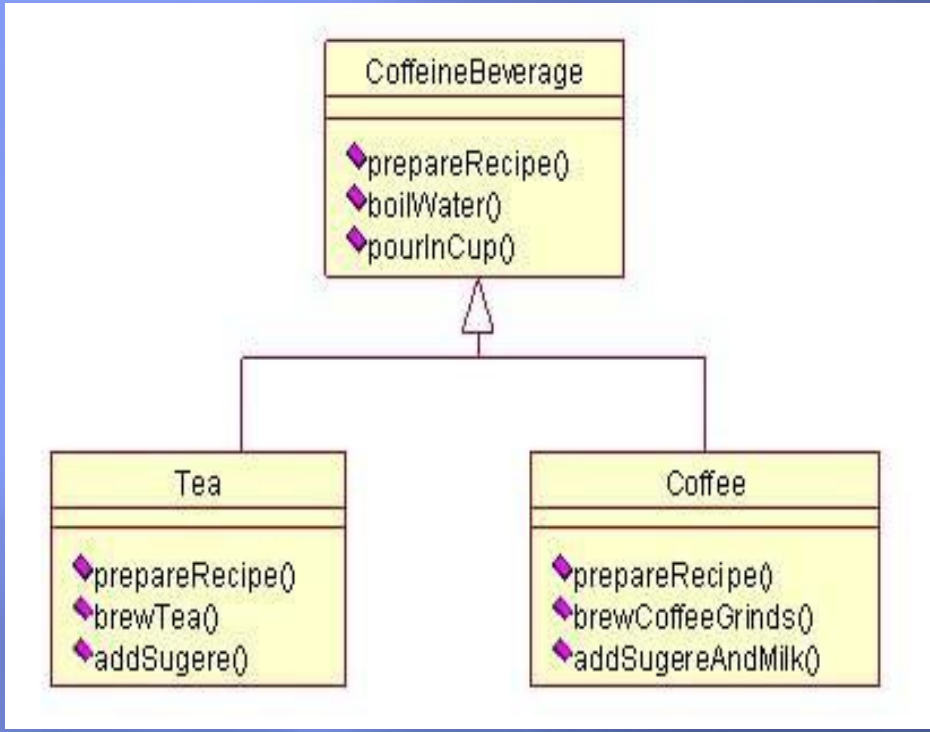
ایجاد یک قالب و اسکلت برای یک الگوریتم

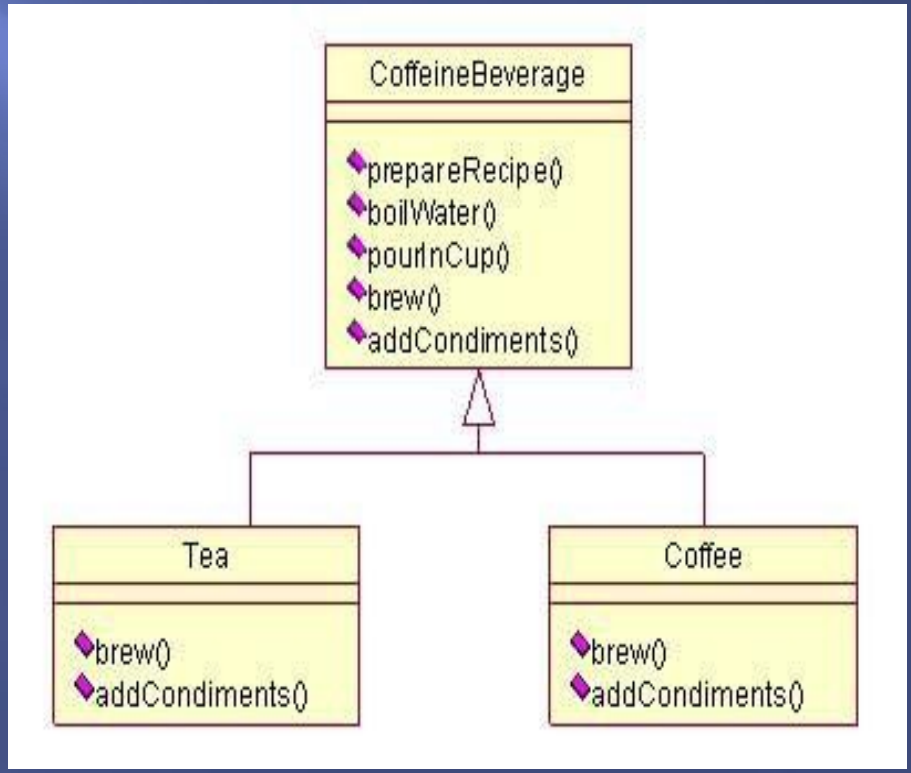
راه حل :

یک الگوریتم را در یک متد در کلاس پایه abstract class تعریف می کند و اجازه می دهد زیر کلاس ها (concerte class) یک یا چند مرحله از الگوریتم را پیاده سازی کنند .



Behavioral:Template method الگوي





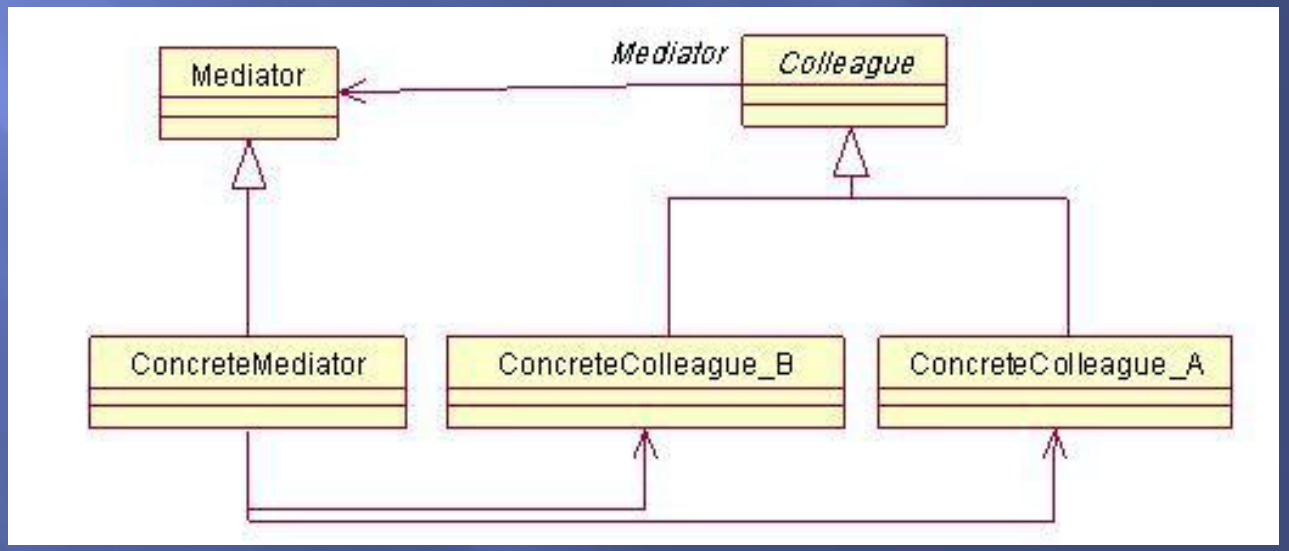
Behavioral:Mediator الگوي ميانجي

زمینه :

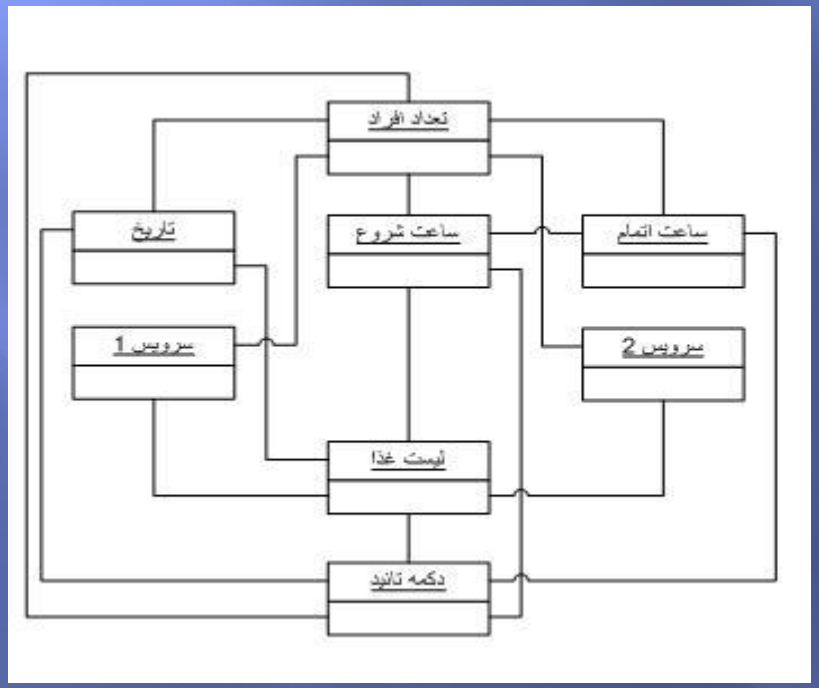
وابستگی اشیاء به یکدیگر باعث پیچیده شدن منطق برنامه خواهد شد و این نگهداری سیستم و قابلیت استفاده مجدد را مشکل تر خواهد کرد.

راه حل :

ایجاد یک کلاس جدید (modiator)، به جای اینکه کلاس ها بایکدیگر بطور مستقیم ارتباط برقرار کنند، با کلاس جدید ارتباط بر قرار کنند.



Behavioral:Mediator الگوي ميانجي



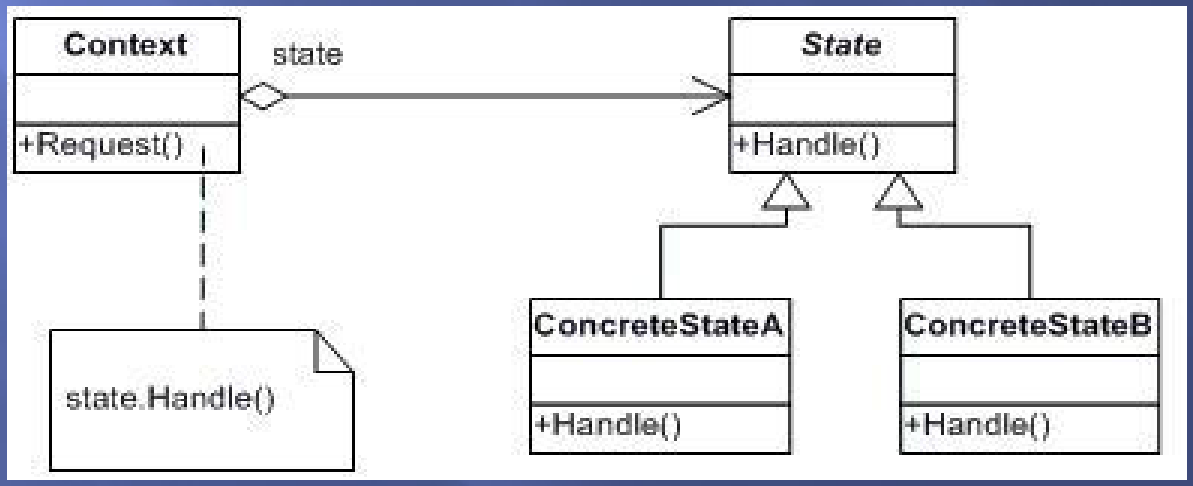
الگوی حالت (State) / Behavioral

زمینه :

پیاده سازی یک شی که دارای چندین حالت است و اضافه و یا تغییر حالت شی درخواست شود.

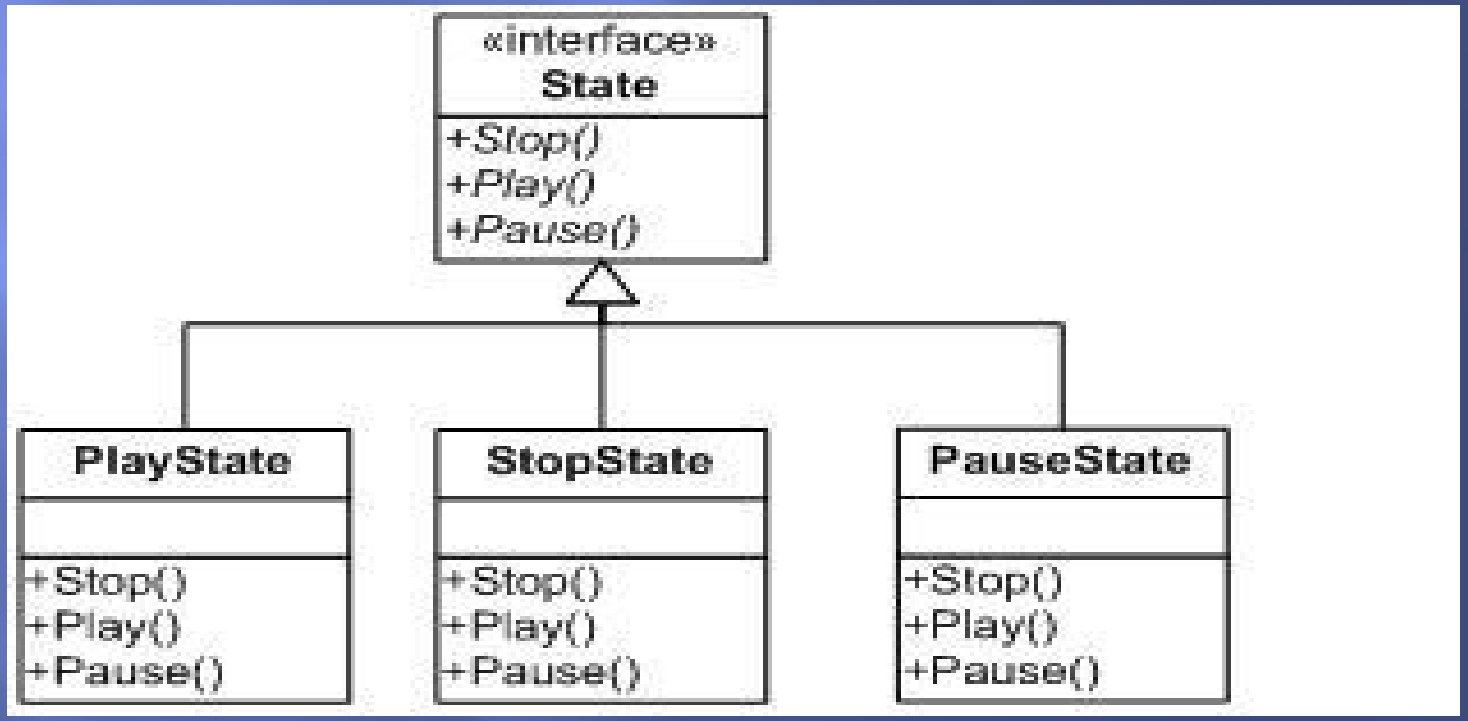
راه حل :

- یک واسط بنام state تعریف می گردد که شامل یک متد برای هر عملی است که شی مورد نظر می تواند ا نجام دهد.
- برای هر حالت شی، یک کلاس جهت پیاده سازی آن ارائه می گردد.



الگوی حالت (ادامه)

برنامه Windows Media Player



ضرورت توجه به الگوها

- **جداسازی کلاسها از یکدیگر و جلوگیری از این مورد که کلاسها اطلاعات زیادی از یکدیگر داشته باشند.**
- افزایش استفاده مجدد از کتابخانه ها و قطعات آماده و راه حل ها
- افزایش سرعت در طراحی سیستم ها
- بهره گیری از تجربیات دیگران و عدم نیاز به ابداع مجدد راه حلهای تثبیت شده
- اسقرار واژگان مشترک