الگوهای طراحی (Design Patterns)

عرفان زیده‌سرایی

فهرست مطالب

- الگو چیست؟
- تاریخچه الگوهای طراحی
- فرمت استاندارد الگوها
- معرفی و پیاده‌سازی الگوهای طراحی
- دسته‌بندی الگوها
- مزایای الگوهای طراحی
- منابع

بالا بردن قابلیت استفاده مجدد

- **هدف اصلی** مهندسی نرم‌افزار استفاده مجدد از طراحی و محصولات مرتبط است :
  - ✓ **چارچوب‌ها (Frameworks)**
  - ✓ **الگوها (Patterns)**

تعریف Framework از دیدگاه شئ‌گرایی

- یک Framework عبارت است **از مجموعه‌ای از کلاس‌هایی که با ترکیب و کار در کنار هم، یک طراحی با قابلیت استفاده مجدد برای یک کلاس خاص از نرم‌افزار ایجاد می‌کنند**. (گاما و همکارانش)

تاریخچه الگوهای طراحی

- الگوها به‌طور کلی نشأت گرفته از **علم معماری** و حاصل مطالعات **الکساندر** می‌باشند (۱۹۷۷).

Gang of Four

- (گاما و همکاران، ۱۹۹۵)←(باچمن و همکاران، ۱۹۹۶)←(اشمیت و همکاران، ۲۰۰۰)←(کرچر و جین، ۲۰۰۴)←(باچمن و همکاران، ۲۰۰۷a)←(باچمن و همکاران، ۲۰۰۷b)

تاریخچه الگوهای طراحی

- کتاب **گاما و همکارانش (GoF)** بیشترین تاثیر را در پیشرفت الگوهای طراحی داشته است :

Design Pattern: Elements of Reusable Object-Oriented Software, 1995.

- این کتاب تا سال **۲۰۰۷** بیش از **۳۶** بار تجدید چاپ شده است.

الگو چیست ؟

- سامرویل : الگو **توصیفی** از **مساله** و اساس **راه حل** آن است، به‌طوری که این راه حل می‌تواند به شکل‌های **مختلفی** مورد استفاده قرار گیرد.

الگو = توصیف مساله + بهترین راه حل

- الگوها و زبان الگوها روشی برای توصیف **بهترین عمل**، **طراحی‌های خوب** و **انتقال تجربه** است.

الگو چیست ؟

- الگوهای طراحی اختصاص به دنیای نرم‌افزار ندارند و در برخی صنایع نیز ممکن است از طرح‌ها و الگوها استفاده کنند : **الگوی خیاطی**

- الگوهای طراحی در مهندسی نرم‌افزار معمولاً به طراحی **شئ گرا** مربوط می‌شوند و اثر شگرفی بر طراحی نرم‌افزار شئ گرا داشته‌اند.

الگو چیست ؟

- زمانی یک الگوی طراحی مقبولیت عمومی پیدا می کند که دست کم در **سه** مورد از یک صورت مساله مشابه **نتیجه‌بخش** باشد.

- تعداد الگوهای طراحی بسیار زیاد است ولی تنها تعداد کمی از این الگوها شهرت جهانی پیدا کرده‌اند. زیرا دیگر الگوها یا **بسیار ساده‌اند** که نمی‌توان نام الگوی طراحی را بر آن‌ها گذاشت و یا **بسیار بزرگ و خاص** هستند.

مثال ۱ :

- محاسبه کتانژانت ۲ یک زاویه بر حسب رادیان :

```
double cos = Math.Cos(2);
double sin = Math.Sin(2);
double cot = cos / sin;
```

مثال ۲ :

- اعتبارسنجی آدرس ایمیل توسط عبارت منظم :

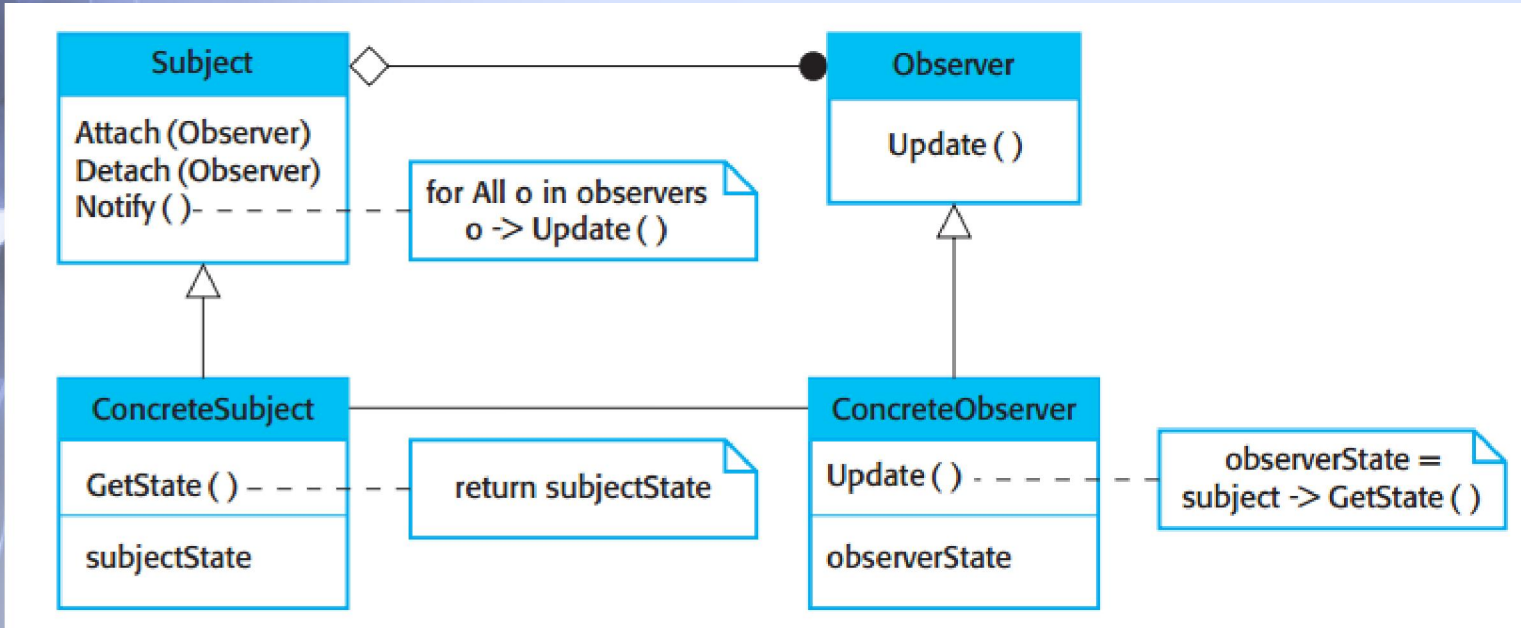
```
string re=@"^([0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*@([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$";
 if (RegularExpressions.Regex.IsMatch(رشته ورودی,re))
        MessageBox.Show("Email is valid");
else
        MessageBox.Show("Email is not valid");
```

فرمت استاندارد الگوها

نام الگو	یک نام خوب و مفید برای الگو
هدف	یک جمله کوتاه و مختصر درباره چیزی که الگو انجام می‌دهد. (تعریف مساله و راه حل به صورت مختصر و مفید)
نام مستعار	نام‌های دیگری که الگو با آن شناخته می‌شود.
ساختار	یک نمایش گرافیکی از الگو
اجزا تشکیل دهنده	کلاس‌ها و اشیای که در الگو شرکت دارند (وجود دارند).
همکاری‌ها	چگونه اجزای تشکیل دهنده با هم همکاری می کنند تا وظایفشان را انجام دهند.
نتایج	نتایج استفاده از الگوی مورد نظر
پیاده‌سازی	تکنیک‌های پیاده‌سازی الگوی مورد نظر
نمونه کد	تکه کدی برای پیاده‌سازی یک نمونه
الگوهای مرتبط	الگوهای طراحی دیگری که ارتباط نزدیگ با الگوی مورد نظر دارند.

نمایش گرافیکی الگوها

- الگوی Observer (بیننده) :



دسته‌بندی الگوها

- الگوهای GoF از لحاظ **هدف** به **سه** دسته تقسیم شده‌اند :

1. **الگوهای آفرینشی (Creational)** : این الگوها در فرآیندهای **ایجاد اشیاء** استفاده می‌شوند.

دسته‌بندی الگوها

2. **الگوهای ساختاری (Structural)** : این الگوها در **ترکیب کلاس‌ها و اشیاء** مورد استفاده قرار می گیرند.

3. **الگوهای رفتاری (Behavioral)** : این الگوها به **چگونگی تعامل میان کلاس‌ها یا اشیاء و نیز نحوه توزیع مسئولیت میان آن‌ها** می‌پردازد.

دسته‌بندی الگوها

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

معرفی الگوهای طراحی پر کاربرد

- **Singleton Pattern :** برای اینکه از یک کلاس فقط **یک نمونه یا شئ** ایجاد شود، از الگوی سینگلتون استفاده می‌شود.
  شرکت X **فقط** می‌تواند یک مدیر عامل داشته باشد.

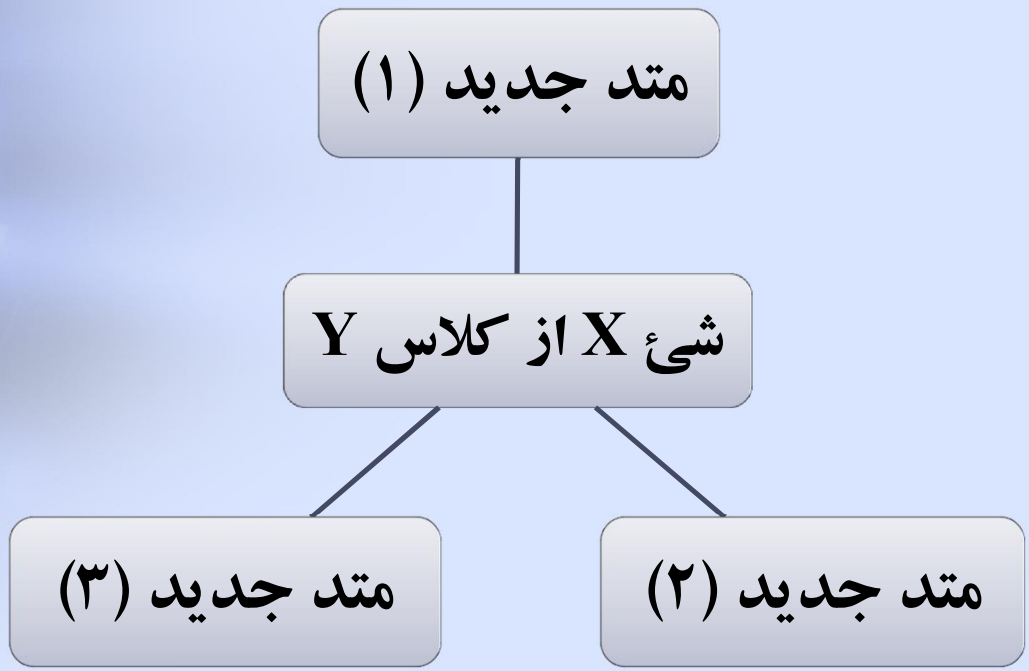
معرفی الگوهای طراحی پر کاربرد

- **Factory Method Pattern** : این الگو، کلاسی است که **نمونه‌های کلاس‌های دیگر را با توجه به پارامترهای که برای آن ارسال می‌شود، ایجاد می‌نماید.**

معرفی الگوهای طراحی پر کاربرد

- **Decorator Pattern** : این الگو، امکان افزودن قابلیت‌های جدید به یک شئ را به صورت **پویا و در زمان اجرا** ممکن می‌سازد.
- به‌عبارت دیگر، توسط این الگو می‌توانیم یک قابلیت را تنها به یک شئ خاص از یک کلاس اختصاص بدهیم **بدون آنکه سایر اشیاء آن کلاس تغییر کنند.**

معرفی الگوهای طراحی پر کاربرد

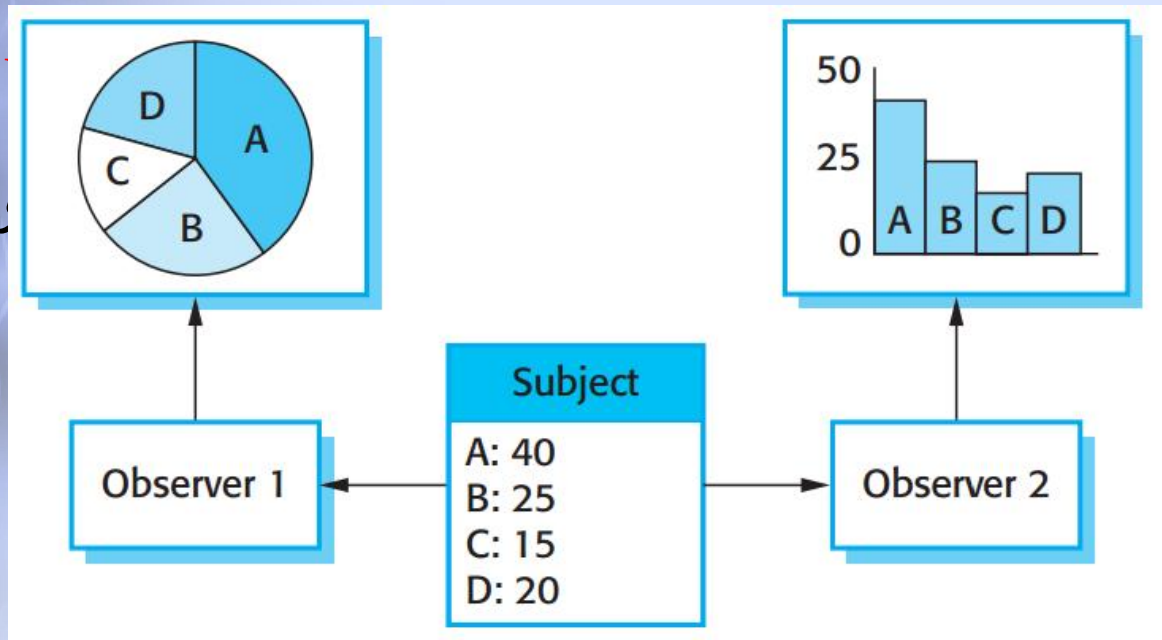


معرفی الگوهای طراحی پر کاربرد

- **Observer Pattern** : این الگو یک وابستگی، **یک به چند** بین اشیاء می‌باشد، هنگامیکه شئ طرف یک، وضعیتش **تغییر** کند، اشیاء وابسته مطلع می‌شوند و وضعیت خودشان را با توجه به وضعیت طرف یک **بروزرسانی** می‌کنند.
- به‌عبارت دیگر، **الگوی بیننده نمایش شئ را از خود شئ جدا می‌کند.** (سامرویل)

معرفی الگوهای طراحی پر کاربرد

- در ا
  می‌ن
  هست



الگوی Observer

پیاده‌سازی الگوی سینگلتون – روش اول

۱-یک شئ از کلاس سینگلتون در همان ابتدای آغاز به کار برنامه باید ایجاد شود.

۲-کلاس سینگلتون باید یک متد سازنده از نوع private داشته باشد تا نمونه جدید نتواند در خارج از این کلاس ایجاد شود.

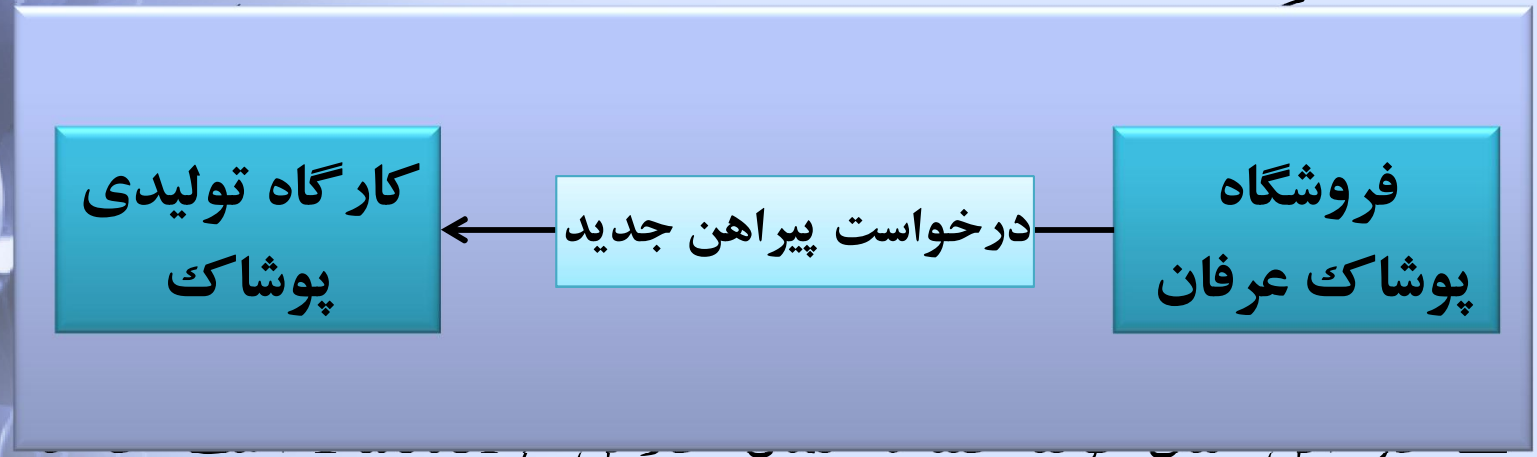
۳-یک متغیر از نوع static private جهت نگهداری تنها شیءِ کلاس سینگلتون در داخل متد تعریف می‌شود که نهایتاً شیء را بر می گرداند.

پیاده‌سازی الگوی سینگلتون – روش دوم

- آسان‌ترین راه برای ایجاد کلاسی که فقط بتواند یک نمونه از آن ایجاد شود استفاده از یک **متغییر استاتیک** در داخل کلاس می‌باشد. اولین نمونه‌ای که از این کلاس ایجاد می شود ما متغییر استاتیک را مقدار true می‌دهیم و در سایر دفعات این متغییر استاتیک در سازنده تست می‌شود، اگر مقدارش false باشد یک شئ از کلاس ایجاد خواهد شد در غیر این صورت شئ جدیدی از آن کلاس ایجاد نخواهد شد.

پیاده‌سازی الگوی Factory Method

- تصورِ کنید که ما یک فروشگاه پوشاک داریم، هر وقت که



توجه به پارامترهای ارسال شده (لیست پوشاک مورد نظر فروشگاه) لباس مورد نظر (شئ مورد نظر) را تولید می‌کند و به مشتری ارسال می‌کند.

چرا الگوهای طراحی را مطالعه کنیم؟

- **استفاده مجدد از راه حل‌ها**
- **از تجربیات دیگران بهره می‌گیریم و نیازی به ابداع مجدد این راه حل‌ها که تثبیت شده‌اند نیست.**
- **استقرار واژگان مشترک :** ارتباطات و کار گروهی نیازمند واژگان و فرهنگ لغات مشترک است. الگوهای طراحی این واژگان را برای فاز تحلیل و طراحی پروژه مهیا می‌کنند.

سایر مزایای الگوهای طراحی

- همان‌گونه که طراحی شئ‌گرا ادعا می‌کند که استفاده مجدد از کتابخانه‌ها و قطعات را افزایش می‌دهد، ادعا می‌شود که **استفاده از الگوهای طراحی، استفاده مجدد از کتابخانه‌ها و قطعات را افزایش می‌دهد.**
- یکی از دلایل مهم استفاده از الگوها، **افزایش سرعت در طراحی سیستم‌ها** می‌باشد.

سایر مزایای الگوهای طراحی

- **کارآیی** و **مقبولیت** الگوهای طراحی GoF به‌حدی رسیده است که امروزه ابزارهای حرفه‌ای CASE به‌طور مستقیم از آن‌ها پشتیبانی می‌کنند و امکان اعمال الگوهای طراحی GoF را فراهم می‌کنند.

  از جمله این ابزارها می‌توان از Rational XDE، Rational Rose و Borland Together نام برد.

هدف الگوهای طراحی

- هدف الگوها در جامعه نرم‌افزاری ایجاد یک **ادبیات پایه‌ای** است تا توسعه‌دهندگان نرم‌افزار در حل مشکلات تکراری، در کل جریان توسعه از آن یاری گیرند.

الگوهای طراحی در مقایسه با Frameworkها

- هر دو ابزاری برای استفاده مجدد در فرایندهای تولید نرم‌افزار هستند.
- الگوهای طراحی در مقایسه با Frameworkها **کلی‌تر و انتزاعی‌تر** هستند.
- یک Framework **دارای معماری بزرگتری است** و ممکن است در معماری خود از چندین الگوی طراحی استفاده کند.

الگوهای طراحی در مقایسه با Frameworkها

- Framework یک **پیاده‌سازی واقعی** از یک یا گروهی از الگوهای طراحی است.

جایگاه الگوهای طراحی در مهندسی نرم‌افزار

منابع

[۱]-سامرویل، یان؛ جعفرنژاد قمی، عین الله؛ مهندسی نرم‌افزار؛ ویراست نهم، علوم رایانه : بابل؛ ۱۳۹۱.

[۲]-پرسمن، راجر اس؛ سالخورده، محمد مهدی؛ مهندسی نرم‌افزار؛ ویراست هفتم، خراسان، باقانی : مشهد؛ ۱۳۹۰.

[3]-Gang of Four; Design Pattern: Elements of Reusable Object-Oriented Software; 1995.

[4]-www.uml.org

[5]-www.hillside.net

[6]-www.30sharp.com

[7]-www.ooa.blogfa.com

منبع پیشنهادی

نصیری، وحید؛ الگوهای طراحی برنامه‌نویسی شئ‌گرا در سی‌شارپ؛ ناقوس : تهران؛ ۱۳۸۵.

اگر من شخصی بودم که فقط می‌خواست برنده شود، تاکنون باید وارد عرصه دیگری می‌شدم. فکر نمی‌کنید که اگر من در ذهنم خط پایانی را متصور بودم سال‌ها پیش می‌بایست از آن گذر کرده باشم؟ (بیل گیتس)