تاریخچه مختصری بر الگوها

ایده اصلی الگوهای نرم افزاری ریشه در حوزه معماری دارد. کریستوفر الکساندر (معمار) (Christopher Alexander) در اواخر دهه ۱۹۷۰، دو کتاب انقلابی در تشریح الگوهایی برای معماری ساختمانها (Building Architecture) و برنامه ریزی شهری (Urban Planning)نوشت که عبارتند از:

Pattern Language: Towns, Buildings, Construction (Oxford University Press, 1977)

The Timeless Way of Building (Oxford University Press, 1979)

البته ایده پس پرده الگوهای نرم افزاری ربطی به معماری نداشته و به حوزه ای خارج از آن مربوط می شوند، که عبارت است از توسعه و تولید نرم افزارها.

در سال ۱۹۸۷، وارد کانینگهام (Ward Cunningham) و کنت بک (Kent Beck) از ایده ها الکساندر در ساخت پنج الگوی طراحی واسط کاربر (user interface) استفاده کردند. این دو جزوه ای را پیرامون الگوهای واسط کاربر تحت عنوان Using Pattern Languages for Object-Oriented Programsدر OOPLSA-87 منتشر کردند.

در اواخر ۱۹۹۰، اریک گاما(Erich Gamma) ، ریچارد هلم(Richard Helm) ، جان ولسایدز (John Vlissides) و رالف جانسون (Ralph Johnson) کار خود را بروی یکی از مؤثرترین کتابهای دو دهی اخیر شروع کردند .Design Patterns : کتاب مذکور در سال ۱۹۹۴ به چاپ رسید و اغلب آن را کتاب گروه چهار (Gang of Four) یا به اختصار کتاب GoF می نامند. این کتاب موجب گسترش ایده الگوها شده و هم اکنون نیز منبع بسیاری از کتابها و مقالات مرتبط با الگوها می باشد.

قسمت اول

کاتالوگFundamental

الگوهای موجود در این کاتالوگ:

1. Delegation
2. Interface
3. Abstract Superclass

4. Interface and Abstract Class
5. Immutable
6. Marker Interface
7. Proxy

دانستن الگوهای این کاتالوگ از اهمیت بسزایی برخوردار است، چراکه شما شاهد کاربرد فراوان این الگوها در سایر الگوها خواهید بود.

الگوهایDelegation ، Interface، Abstract Superclassو Interface and Abstract Class در نحوه سازماندهی ارتباطات بین کلاسها مورد استفاده قرار می گیرند. اکثر الگوها حداقل یکی از این الگوها را مورد استفاده قرار می دهند. این الگوها آنقدر شناخته شده اند که اغلب در بخش الگوهای مرتبط بیشتر الگوها (بخشی جهت اشاره به سایر الگوهای مرتبط با الگوی مورد بحث) بدانها اشاره ای نمی شود.

از الگوی Immutable به عنوان روشی جهت پیشگیری از باگها و تاخیر، در هنگام دسترسی چندین شیئ به یک شیئ مشابه استفاده می شود. این الگو اگرچه بطور واضح در اکثر الگوها مورد استفاده قرار نمی گیرد، ولیکن می توان از مزایای آن در بسیاری از الگوها بهره برد.

الگوی Marker Interface روشی را جهت تسهیل در طراحی کلاسهایی پیشنهاد می کند که یک ویژگی ثابت بولی دارند (constant boolean attribute).

الگوی Proxy اساس و بنیان الگوهاییست که موضوعات (عملکردها) کلی را به اشتراک می گذارند. به عبارتی دیگر هنگامی که شیئی دسترسی به شیئ دیگری را با یک روش ارتباطی مدیریت می کند.

کاتالوگCreational

الگوهای موجود در این کاتالوگ:

1. Factory Method
2. Abstract Factory
3. Builder

4. Prototype
5. Singleton
6. Object Pool

از الگوهای Creational در ساخت اشیائی استفاده می شود که در هنگام ساخت به اتخاذ تصمیمهایی نیاز دارند. این تصمیمها می توانند تعیین دینامیکی کلاسی جهت نمونه سازی(instantiate) ، و یا تعیین اشیائی باشد که یک شیئ می بایست امور خود را بدانها واگذار نماید. الگوهای Creational به ما می گویند که چگونه این تصمیمات را ساختاردهی و پنهان سازی (encapsulate)نماییم.

در اکثر موقعیتها بیش از یک الگوی Creational جوابگوی نیاز کاری خواهند بود که انتخاب یکی از آنها خود می تواند کار هوشمندانه ای باشد. گاهی اوقات می توان چندین الگو را به نحو شایسته ای با یکدیگر ترکیب نمود، ولی در سایر موارد می بایست از بین الگوهای مناسب یکی را انتخاب کرد. بنابر همین دلایل بسیار مهم است که الگوهای این کاتالوگ را به خوبی بشناسید.

اگر تنها برای مطالعه یکی از الگوهای این کاتالوگ وقت دارید، پرکاربرد ترین آنها الگوی Factory Method است. الگوی مذکور روشیست که اشیاء با استفاده از آن اشیاء دیگری را بدون اطلاع از کلاس آنها نمونه سازی می کنند.

اشیاء با استفاده از الگوی Abstract Factory می توانند اشیاء گوناگون و متفاوتی را بدون اطلاع از کلاس آنها نمونه سازی کنند، البته کلاسهای مذکور می بایست در ترکیب درستی قرار داشته باشند.

از الگوی Builder جهت تعیین کلاس شیئی استفاده می شود که براساس مضمون (content) یا مفهوم (context) آن ساخته شده است.

الگوی Prototype به اشیاء اجازه می دهد تا اشیاء اختصاصی دیگری را بدون اطلاع دقیق از کلاس یا جزئیات ساخت آنها بسازد.

با استفاده از الگوی Singleton ، چندین شیئ می توانند یک شیئ مشترک را، بدون اطلاع از وجود فعلی آن، مورد استفاده قرار دهند.

الگوی Object Pool روشیست جهت استفاده مجدد از اشیاء، به جای ساخت اشیاء جدید.

کاتالوگ Partitioning

الگوهای موجود در این کاتالوگ:

1. Filter
2. Composite
3. Read-Only Interface

یکی از روشهای رایج حل مسائل *خرد کردن و غلبه* (divide and conqure)نام دارد. در این روش مسئله ای که حل آن مشکل است به چند مسئله کوچکتر، که حل آنها ساده تر است، خرد می شود. از الگوهای این کاتالوگ در راستای تسهیل و رسیدن به یک طراحی خوب در دسته بندی کلاسها و واسطها (interface) استفاده می شود.

از الگوی Filter در مدیریت محاسبات بروی جریانی از اطلاعات استفاده می شود .این روش قابلیت انعطاف فراوانی داشته و به شما اجازه می دهد تا بروی یک جریان اطلاعات مشابه، محاسبات متفاوتی را با یکدیگر ترکیب کرده و تطبیق دهید.

الگوی Composite روشی را جهت سازماندهی سلسله مراتبی (hierarchy) از اشیاء ارایه می کند.

از الگوی Read-Only Interface در دسته بندی اشیائی استفاده می شود که از یک شیئ استفاده می کنند. در این روش به اشیائی که اجازه تغییر شیئ مذکور را داشته باشند اجازه داده شده و به آنهایی که مجاز نیستند اجازه داده نمی شود.

کاتالوگ Structural

الگوهای موجود در این کاتالوگ:

1. Adapter
2. Iterator
3. Bridge
4. Façade
5. Flyweight
6. Dynamic Linkage
7. Virtual Proxy
8. Decorator
9. Cache Management

الگوهای این کاتالوگ روشهای رایجی را فراهم می کنند که به کمک آنها می توان اشیائی از انواع (type) متفاوت را جهت همکاری با یکدیگر سازماندهی نمود.

الگوی Adapter توضیح می دهد که چگونه کاربر یک شیئ می تواند انتظار پیاده سازی شدن یک واسط خاص را از آن داشته باشد، حال آنکه واسط مذکور توسط شیئ مورد نظر پیاده سازی نشده است.

الگوی Iterator توضیح می دهد که چگونه شیئی می تواند بدون اطلاع از ساختار یا کلاس یک مجموعه(collection) ، به اشیاء عضو مجموعه دسترسی داشته باشد.

الگوی Bridge روشیست برای مدیریت سلسله مراتبهای موازی از خلاصه سازیها (abstractions) و پیاده سازیها (implementations).

از الگوی Façade جهت مخفی کردن پیچیدگیهای عملکرد گروهی از اشیاء مرتبط با یکدیگر در یک شیئ منفرد استفاده می شود.

به منظور پیشگیری از ساخت چندین نمونه از شیئی که به حافظه زیادی احتیاج دارد، می توان از الگوی Flyweight استفاده نمود. در این الگو از اشیاء مشترکی استفاده می شود که حاوی مقادیر مشابه اند.

از الگوی Dynamic Linking به منظور افزودن دینامیکی (dynamic adding) کلاسهایی به برنامه، در زمان اجرا (runtime)استفاده می شود.

الگوی Virtual Proxy روشیست جهت به تاخیر انداختن ساخت اشیاء، به گونه ای که این موضوع از دید کاربران اشیاء مورد نظر مخفی باشد.

از الگوی Decorator به منظور افزایش یا تغییر رفتارهای اشیاء موجود بصورت دینامیک، استفاده می شود.

الگوی Cache Management روشیست جهت جلوگیری از ساخت تکراری اشیاء مشابه، در این روش از اشیاء ساخته شده استفاده مجدد می شود.

کاتالوگ Behavioral

الگوهای موجود در این کاتالوگ:

1. Chain of Responsibility
2. Command
3. Little Language
4. Mediator
5. Snapshot
6. Observer
7. State
8. State
9. Null Object
10. Strategy
11. Template Method
12. Visitor

از الگوهای این کاتالوگ به منظور سازماندهی، مدیریت و ترکیب رفتارها (behavior) استفاده می شود.

کاتالوگConcurrency

الگوهای موجود در این کاتالوگ:

1. Single Threaded Execution
2. Lock Object
3. Guarded Suspension
4. Balking
5. Scheduler
6. Read/Write Lock
7. Producer-Consumer
8. Tow-Phase Termination
9. Double Buffering

10. Asynchronous Processing
11. Future

از الگوهای این کاتالوگ در مدیریت و کنترل پردازشهای همزمان استفاده می شود. در اصل این الگوها به دو گونه از مشکلات پاسخ می دهند:

منابع مشترک .(Shared Resources) هنگامی که چندین پردازش همزمان از داده ها یا انواع دیگری از منابع بصورت اشتراکی استفاده می کنند، درصورت مراجعه همزمان آنها به این منابع ممکن است مشکلاتی پیش آید. به منظور اطمینان از درستی پردازشهای انجام شده بروی منابع مشترک، پردازشها می بایست به اندازه کافی در دسترسی همزمان به داده ها محدود شده و منابع مشترک در هر لحظه تنها در اختیار یکی از پردازشها قرار بگیرند .البته اعمال محدودیتهای بیش از حد می تواند پردازشها را به بن بست (deadlock)رسانده و مانع از خاتمه آنها شوند.

بن بست به وضعیتی گفته می شود که در آن دو پردازش منتظر انجام کاری توسط دیگری می شوند. با توجه به اینکه هریک از این پردازشها منتظر دیگری می ماند، هیچ کاری انجام نشده و هردوی آنها تا بینهایت منتظر خواهند ماند (مثل جریان کار پیدا کردن آقا پسرها و تهیه جهیزیه دختر خانومها به منظور ازدواج، خلاصه اینکه بد بلاییه.(

مراحل انجام پردازشها .(Sequence of Operations) هنگامیکه پردازشها در دسترسی به منابع مشترک بصورت یکی در هر لحظه (one-at-a-time) محدود می شوند، می بایست اطمینان حاصل نمود که این دسترسی ها با ترتیب خاصی انجام شوند. به عنوان مثال نمی توان شیئی را پیش از ورود به یک ساختمان داده از آن حذف نمود (باز هم مثال لازمه؟.(

Single Threaded Executionمهمترین الگوی این کاتالوگ است. بیشتر مسائل مربوط به منابع مشترک را می توان تنها با استفاده از این الگو حل کرد، این الگو اطمینان می دهد که در هر لحظه بیش از یک روند (thread) به یک منبع مشترک دسترسی نخواهد داشت. البته از این الگو در شرایطی استفاده می شود که ترتیب دسترسی به منابع از اهمیت کمی برخوردار باشد، در چنین شرایطی می بایست از الگوی Scheduler استفاده شود.

الگوی Guarded Suspension در شرایطی که روندی علارقم داشتن دسترسی انحصاری به یک منبع درمی یابد که بنابه دلایلی نمی تواند کار خود را به پایان برساند (مثلا چیزی هنوز آماده نیست)، راهنمایی می کند که چه کاری می بایست انجام شود

(این الگوی خیلی خوبیه.(!!!

هنگامیکه پردازشی نیازمند دسترسی انحصاری به چندین منبع می باشد، از الگوی Lock Objectبه عنوان روشی جهت تسهیل در هماهنگ سازی دسترسی به چندین منبع استفاده می شود.

از الگوی Balking زمانی استفاده می شود که یک پردازش می بایست یا سریعا انجام شود و یا اصلا انجام نشود.

الگوی Read/Write Lock شکل دیگریست از دسترسی یکی در لحظه .(one-at-a-time) در این حالت برخی پردازشها می توانند منبعی را به اشتراک بگذارند و برخی دیگر خیر.

از الگوی Producer-Consumer به منظور ایجاد هماهنگی بین اشیاء تولید کننده یک منبع و اشیاء مصرف کننده آن منبع استفاده می شود.

از الگوی Tow-Phase Termination جهت خاتمه دادن به روندها، در قالب یک ترتیب منظم استفاده می شود.

الگوی Double Buffering یکی از اشکال خاص الگوی Producer-Consumer است. در این الگو منابع پر کاربرد پیش از نیاز بدانها ساخته می شوند.

الگوی Asynchronous Processing روشیست به منظور جلوگیری از انتظار برای نتیجه یک پردازش، در شرایطی که نتیجه آن سریعا مورد نیاز نیست.

الگوی Future توضیح می دهد که چگونه می توان باعث شد تا کلاسهای فراخوانی کننده یک پردازش، از هماهنگ بودن (synchronous)یا هماهنگ نبودن (asynchronous)آن مطلع نشوند .