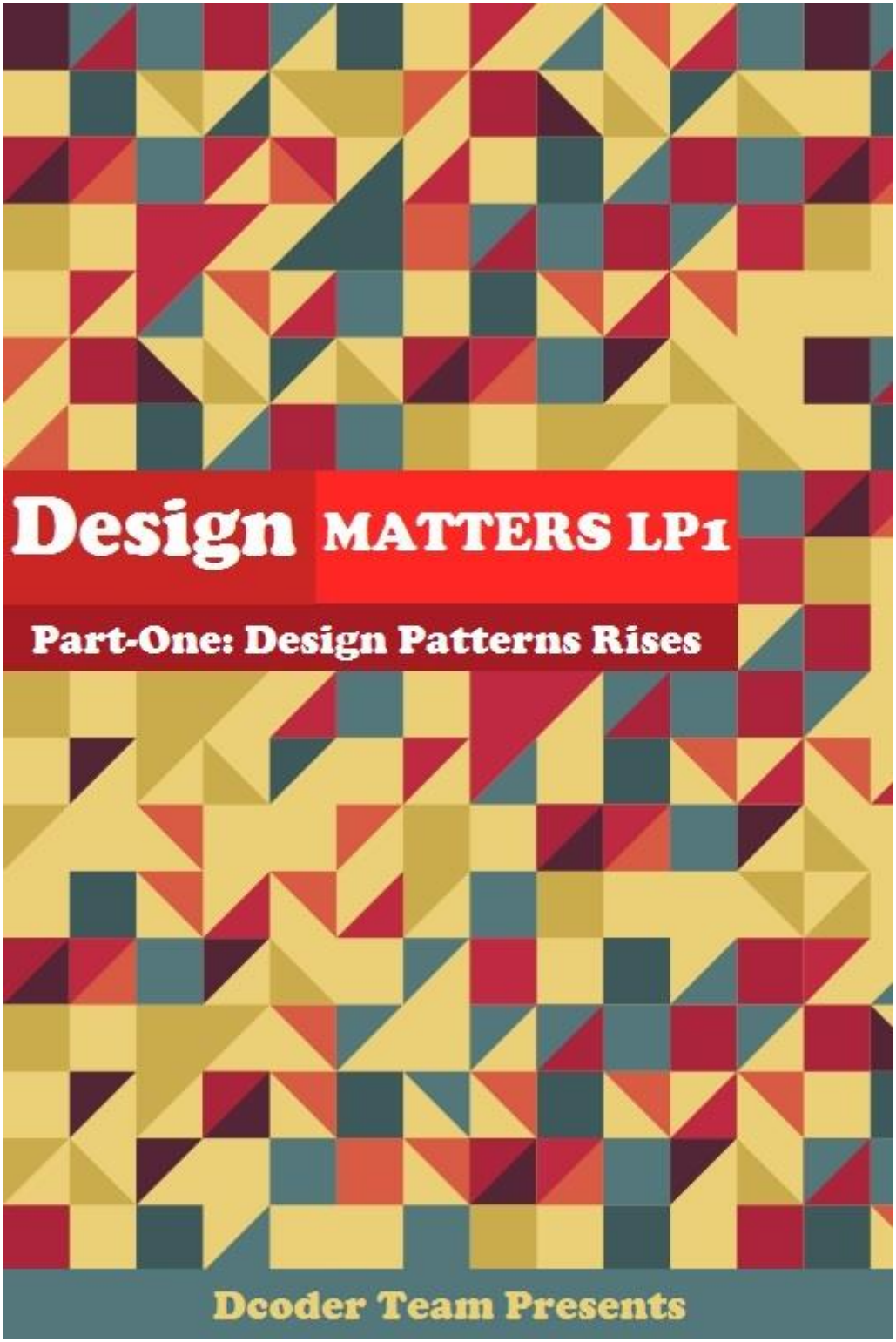
Design MATTERS LP1
Part-One: Design Patterns Rises
Dcoder Team Presents

Information

Article : Design Matters LP 1

Authors : Mhmv and Pilofil and Joulook

Prerequisite : Object-Oriented Concepts

Published By : Dcoder Team

Follow us on Telegram : https://t.me/de_coder

The Cover of Article is Adapted From “Design Patterns Explained Simply By Alexander Shvets” Cover Page

Preface

در ابتدا جا داره بگیم که این متن اقتباس شده از چند کتاب مرجع در مبحث Design Pattern هستش که در آخر این مطلب براتون قرار خواهیم داد و نویسندگان این مطلب هیچ دخل و تصرفی بر محتوای علمی آن نداشته اند و توصیه اکید ما بر مطالعه منابع می باشد نه صرف یک گزارش از این منابع . از ترجمه کردن کلمات فنی اکیدا خودداری شده است

Part-One : Design-Pattern Rises

Why Design Pattern ?

طراحیِ Object-Oriented یک نرم افزار کار دشواری است و طراحی reusable آن کاری دشوار تر! اگر شما چیزی از Object-Oriented نمی دانید خواندن این مطلب کمک چندانی به شما نخواهد کرد و می توان گفت که پیش نیاز یادگیری بحثِ Design pattern محیط بودن بر Concept های Object-Oriented می باشد . حال طراحی reusable چیست ؟ اگر شما برنامه نویسی کرده باشید متوجه این مسئله شده اید که در پایان برنامه نویسی ، شما توانسته اید یک solution و یا یک راه حل برای مشکلی که در پیش روی شما بوده بدست آورید و به اصطلاح Problem Solved کرده اید. و اگر این تجربه را بار ها کسب کرده باشید متوجه این قضیه شده اید که گاها با همان Problem Domain ای در برنامه نویسی مواجه شده اید که در تجربه های قبلی تان بهش برخورده بودید و توانسته بودید راه حلی را برایش بدست آورید . حال ممکن است این Problem Domain تکرای باشد اما بخاطر متفاوت بودن شرایط و Situation Type ، شما نتوانسته اید که همان راهکار قبلی را برای این مشکل تکراری به کار ببرید و مجبور شده اید که راه حل تازه ای برایش بدست آورید . به عبارتی میتوان گفت که طراحی شما Flexible و Reusable نبوده و موجب شده که شما برای مسائل تکراری راه حل های متفاوتی ارائه دهید و این چیز خوبی نیست . طراحی ما باید یک طراحی Reusable باشد و این تنها ویژگی یک طراحی خوب نیست عوامل دیگری مانند Elegant بودن طراحی یا Flexible بودن و efficiency جزو معیار های دیگر یک Design خوب می باشد . حال سوالی پیش می آید و آن این است که چگونه در برخورد با یک مسئله یا Problem Domain جدید ، یک Design با معیار های ذکر شده انجام بدهیم که وقتی دوباره با همان مشکل در کار های بعدی مواجه می شویم بتوانیم با حداقل تغییرات در Solution قبلی خود راه حل مطلوب را بدست آوریم ؟ در جواب باید گفت که این کار به هیچ عنوان کار راحتی نیست و نیاز به تجربه های زیاد و موفق دارد . یعنی با گذشت زمان مویسر می شود . شما باید بعد از هر بار انجام یک design تجربیات و نتایج بدست آمده را Record کنید و در هر بار طراحی از تجربیات قبلی استفاده کنید و قدرت

طراحی خود را ارتقاع ببخشید که به اصطلاح می گویند Sprint Retrospective و در آخر تبدیل به یک Expert Object-Oriented Designer شوید . قطعا شما این همه وقت را ندارید که بخواهید خرج به دست آوردن تجربیات کنید تا بتوانید یک Design با معیار های ذکر شده انجام دهید . اینجاست که بحث Design Pattern مطرح می شود

What is Design Pattern ?

ابتدا Design Pattern را تعریف می کنیم : در دنیای ما یک سری مسائل و Problem ها تکرار شونده هستند و بار ها به آن ها برخورد میکنیم . یک بار برای همیشه آمده اند برای آن Problem ها Solution ای را پیدا کرده اند . هر Pattern بیانگر یک Problem با Solution خاص خودش است به عبارتی می گویند یک Design Pattern باید Best Practice عه مشکلِ مربوطه باشد (یعنی هیچ Solution بهتری برای آن Problem Domain وجود نداشته باشد) و وظیفه یک Designer این است که Pattern Matching انجام بدهد یعنی بیاید بررسی بکند که کدام Design Pattern به Problem ای که پیش رو دارد، می خورد و می تواند آن را تطبیق دهد.

Who is Design Pattern ?

حال به بررسی Characteristics یا همان خواص Design Pattern ها می پردازیم :

1.باید جوری باشد که دیگر لازم نباشد که ما راجع به مسئله فکر کنیم و تنها کار ما Pattern Matching باشد به این خاصیت Smart می گویند

2. وابسته به یک نوع سیستم خاص یا زبان برنامه نویسی نباشد باید مختص به یک Problem خاص به اما به طور کلی و عام در آن محدوده باشد که اصطلاحا به این خاصیت می گویند Generic

3. باید اثبات شده باشد که Solution ای که پیشنهاد می دهد به طور حتم مشکل ما را حل خواهد کرد و نیازی به اثبات مجدد نداشته باشد که اصطلاحا می گویند Well-Proven باشد
4. باید Simple باشد
5. باید Reusable باشد که پیشتر توضیح دادیم به چه معناست
6. تمام Solution های Design Pattern ها استوار بر پارادایم Object-Oriented می باشد

Attention Please!

تا اینجای کار توصیه می کنیم یک بار دیگر از ابتدا دوباره متن را بخوانید و سپس ادامه دهید

Design Patterns Structure

هر Design Pattern از چهارتا Basic Part تشیکل شده است که به بررسی آن ها می پردازیم

1. هر Design Pattern دارای اسمی یک یا دو کلمه ای است که بتوان از همان اسمش تعریفی از Problem و Solution و result ای که دارد ، را متوجه شد (یعنی یه اسم مرتبط نسبت به کاری که میکند رویش بگذارند نه اینکه اسم سازنده اش را رویش بگذازند) پس اولین Basic Part آن Name است
2. دومین بخش آن Problem است که باید توضیح دهنده مشکل و زمینه آن باشد مثلا اینکه چگونه یک الگوریتم را در قالب یه Object می توانیم Represent کنیم . گاهی اوقات Problem شامل یک لیست از Condition هایی می شود که باید بررسی شود آن ها وجود دارند یا نه و سپس مطابق به نتیجه آن تصمیم بگیریم که این Design Pattern به کارمان می آید یا خیر

3. سومین بخش آن Solution است که Element هایی را معرفی می کند که وظیفه آن ها Make up کردنِ Relationship ها و Responsibility ها و Collaboration های یک Design است . Solution ها یک پیاده سازی خاص ندارند به این خاطر که آن ها مانند یک Template هستند که در شرایط مختلف مورد استفاده قرار می گیرند و بیشتر شامل Diagram های UML ای هستند
4. چهارمین بخش Consequences & Trade-off میباشد که نشان دهنده Result یک Pattern هستند در اصل نقش Evaluating یک Design را دارد و هزینه ها و مزایای استفاده از یک Design Pattern را مشخص می کنند

Design Patterns Classification

برای Classify کردن Design Pattern ها از دو Item استفاده می شود یکی Purpose که بازتاب گر این است که یک Design Pattern از نظر کاری که انجام می دهد در دسته های مختلف قرار بگیرد که کل آن ها در سه دسته قرار میگیرند Creational و Structural و Behavioral که در جلوتر خصوصیات هر دسته را بیان خواهیم کرد . آیتم بعدی Scope می باشد که مشخص گر این است که یک Pattern در کاربرد اصلیش در قالب Class مورد استفاده گیرد یا در اندازه Object . آن هایی که در Class Scope قرار می گیرند با ارتباط بین class ها و subclass ها کار میکنند مانند بحث Inheritance همچنین آن ها Static هستند و در Compile-time به صورت Fixed هستند اما آن هایی که در Object Scope هستند با ارتباط بین Object ها کار میکنند و میتوانند در Run-time تغییر کنند و Dynamic هستند

حال می رویم سراغ سه بخشِ Purpose ، Creational Patterns زمانی استفاده می شوند که برای ما Instantiation Process اهمیت ویژه ای دارد و در Design از این نوع Design Pattern ها بسته به شرایط و موقعیت های مختلف استفاده می کنیم که این نوع Design Pattern ها از Inheritance استفاده زیادی در ساخت Object می کنند . این نوع Pattern به شما Flexibility زیادی میدهد در مواردی چون : چه چیزی Create شده

و چه کسی آن را Create کرده است و چگونه Create شده است و چه وقتی Create شده است

دومین نوع یعنی Structural Patterns متمرکز است بر چگونگی Compose شدن Class ها و Object ها برای ساخت Structure های بزرگتر. آن Structural Pattern هایی که در Class Scope قرار دارند از Inheritance برای Compose کردن Interface ها استفاد ه می کنند این نوع Design Pattern ها به طور خاص به شدت برای Develop کردن Class Library های از هم Independent که بتوانند با هم کار کنند کاربرد دارد

و دسته آخر یعنی Behavioral Patterns با الگوریتم ها و Responsibility بین Object ها درگیر است این نوع Pattern نه تنها توصیفگر Class و Object است بلکه Communication بین آن ها را هم مشخص می کند این نوع Pattern ها مناسب زمانی است که Control Flow ما پیچیده است و Follow کردن آن در Run-time کار دشواری است . این نوع Pattern ها تمرکز ما را از Control Flow دور می کنند و اجازه می دهند که ما فقط به نحوه ی Interconnect شدنِ Object ها متمرکز شویم

در شکل زیر میتوانید نام انواع Design Pattern ها را در کتگوری هایی که قرار دارند ببینید

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method (107)	Adapter (class) (139)	Interpreter (243) Template Method (325)
	Object	Abstract Factory (87) Builder (97) Prototype (117) Singleton (127)	Adapter (object) (139) Bridge (151) Composite (163) Decorator (175) Facade (185) Flyweight (195) Proxy (207)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Observer (293) State (305) Strategy (315) Visitor (331)

Table 1.1: Design pattern space

Where You Are Now ?

بحث Introduction و Concept های Design Pattern در ایجا تمام نمی شود منتها توضیحات بیشتر از این در این مقال نمی گنجد . توصیه اکید ما به شما این است که حتما حتما حتما ... حتما این فصل ها را از کتاب های معرفی شده بخوانید تا هم بیشتر در جریان مطالب قرار بگیرید و هم مطالب اضافه تری یاد بگیرید

کتاب اول : فصل 1

کتاب دوم : فصل 7 از وسط صفحه 258

کتاب سوم : فصل 1

Our Plan For Next Session

در جلسات آینده شروع به معرفی Design Pattern ها خواهیم کرد و هر یک را در یک مثال به زبان های Java , C# , C++ پیاده سازی میکنیم و سورس کدش را برایتان قرار خواهیم داد

Bibliography

1. **Design Patterns Elements of Reusable Object-Oriented Software -Gmma and Helm and Johnson and Vlissides -Addison Wesley**

   Download Link : https://t.me/debrary/525

2. **UML 2 Toolkit -Eriksson and Penker and Lyons and Fado –Wiley**

   Download Link : https://t.me/debrary/527

3. **Java Design Patterns -Rohit Joshi -Exelixis Media**

   Download Link : https://t.me/debrary/529

جهت دریافت قسمت های بعدی با کانال زیر مراجعه کنید

https://t.me/de_coder