Challenging Tomorrow's Changes

初めてやってみたシリーズ
デザインパターン編(Template Method)

技術戦略室
先端技術開発部
三井　省

伊藤忠テクノソリューションズ株式会社

初めてやってみらシリーズ(デザインパターン編)の展望

前回と同じ(今回から初めても分かるように残しておく)

目的

マイクロサービスで活用されるDDDを理解するには、「オブジェクト指向」の理解が必要。
このシリーズを通して「オブジェクト指向」の使い方を学ぶ。

施策

シリーズでのコード展開、及び、説明補足。また不明点の共有と解消。

展望

マイクロサービスを活用したアジャイル開発等に興味を持って頂く。
各自、能動的にDojoでの発言やコード開発を実践して頂ければ幸いです。

アジェンダ

1) 前提/目的
2) オブジェクト指向って？
3) デザインパターンって？
4) Template Methodパターンって？
5) 余談(クラス/インスタンス/オブジェクトの違い)

1. 前提/目的

前提/目的

前提

・選定言語はJava
・デザインパターンを全て学ぶわけではなく、代表的なものを数個学ぶこととする
--Singleton, Template Method, Factory, Strategy, Composite等

背景

現在、マイクロサービスの活用に伴い、DDD(ドメイン駆動設計)が活用されることが多い。
DDDではDIを活用したオブジェクト指向の利用が前提（ヘキサゴナルアーキテクチャ等）となっており、DDD自体理解が難しいので、基本となるオブジェクト指向の理解を目指す。

目的

各種デザインパターンを通して、「オブジェクト指向」の使い方を学ぶ。
今回はTemplate Methodパターンを通して倫理観にも若干触れながら理解を進める。

複数のデザインパターンを通してオブジェクト指向の活用方法を学ぶため、
今回の資料のみで理解するものではないことご了承ください。

2. オブジェクト指向って？

オブジェクト指向って？

前回と同じ(今回から初めても分かるように残しておく)

皆さん、「オブジェクト指向」って現場で使えてますか？

オブジェクト指向の設計者は以下のように定義してるようです。

- 1, *Everything Is An Object.*
- 2, *Objects communicate by sending and receiving messages (in terms of objects).*
- 3, *Objects have their own memory (in terms of objects).*
- 4, *Every object is an instance of a class (which must be an object).*
- 5, *The class holds the shared behavior for its instances (in the form of objects in a program list).*
- 6, *To eval a program list, control is passed to the first object and the remainder is treated as its message.*

— Alan Kay

どれも重要そうですが、実際現場で「使う」という観点ではそんなに多くを気にしてません。

オブジェクト指向って？

前回と同じ(今回から初めても分かるように残しておく)

現場で「使う」という観点に絞ると、実はとてもシンプル。DDDで特に重要なのはオブジェクト指向の内、ポリモーフィズムです。以下の一言が言える方は問題なく使えています。

「インタフェース」を使う

3. デザインパターンって？

デザインパターンって？

前回と同じ(今回から初めても分かるように残しておく)

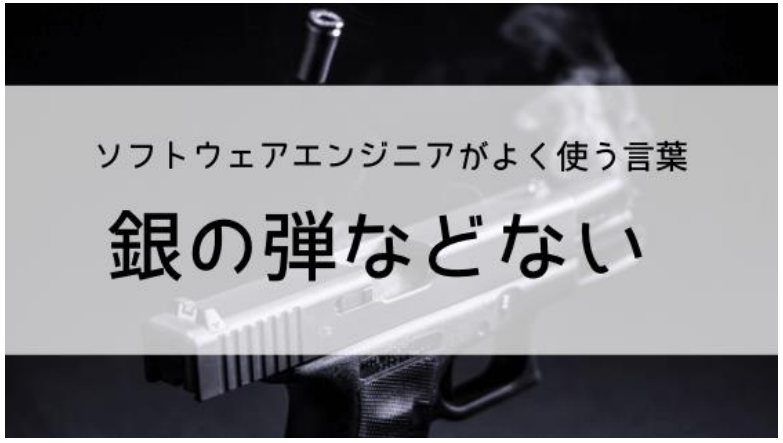
Wikiでは以下となってます。
共通的な問題を解決するパターン手法と簡単に理解して頂ければ。

Design patterns are formalized **best practices** that the programmer can use to **solve common problems** when designing an application or system.

【注意】

デザインパターンはよく使われる設計を一般化してまとめたものに過ぎず、現場の問題を全て解決するものではないです。

※IT業界に「銀の弾丸」はないのでアリマス！



4. Template Methodパターンって？

Template Methodパターンって？

Template Methodパターンのコードを以下GitHubに格納しております。

https://github.com/sho-kenny/hogehoge/tree/master/Java/DP/template

Template Methodパターンは、ソースを見て頂くと分かるように継承(extends)を利用しています。

ここで皆さんに質問です。

Q.「Template Methodパターン」と「継承」の違いは答えられますか？
A.考え方をパターンにするか否かで、結局どちらも同じ。
A.仰々しくパターンといってるだけでは？

Template Methodパターンって？

答えは、**「final」の有無**、です！
Template Methodの親クラスの抽象メソッドはfinalメソッドです。
つまり、子クラスで親クラスの抽象メソッドが拡張されるのを防ぎます。
※拡張しようとするとコンパイルエラーが発生

勘のいい人は、ここで分かったと思います。
「子クラスで拡張できない」＝「**親クラスで書く内容は不変的な内容**」

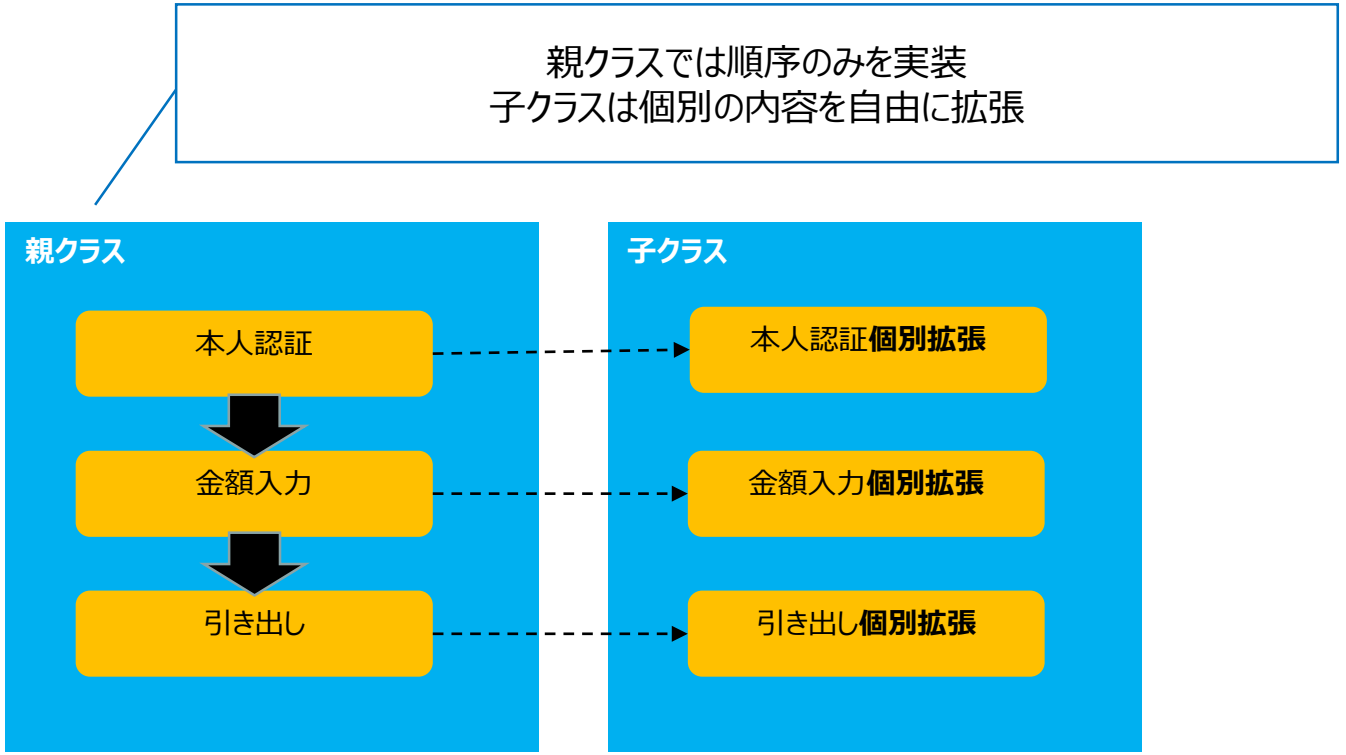
例) お金を銀行口座から引き落とすユースケースを想定

【Template Methodパターンの親クラスで記載すべき内容】
「本人認証⇒金額入力⇒引き出し」の**順序プロセス**を記載
※いかに技術が発達してもこの順序性は不変

【Template Methodパターンの子クラスで記載すべき内容】
本人認証等の中身が、時代によって変更されるのでその具体的な内容を記載

Template Methodパターンって？

例) お金を銀行口座から引き落とすユースケースを想定



Template Methodパターンって？

【Template Methodパターンの有用性】

・修正箇所を局所化できる(基本子クラスの修正でいい)

⇒親クラスは修正しなくていい

【Template Methodパターンの注意点】

・親子関係は関係が密になるのであまり好ましくない。

⇒ソースの解釈が親と子をいったりきたりする

⇒子生成すると親生成というようにメモリ上でも密

・親の内容を充実させ過ぎると子の自由度が利かない

【Template Methodパターンの条件】

・親に不変的な内容を記載し、子に具体的な内容を記載

※その際に継承(extends)を使う

5. 余談(クラス/インスタンス/オブジェクトの違い)

余談(クラス/インスタンス/オブジェクトの違い)

私は、プログラムやり始めに以下の違いがよく分からず、適当に使ってました。

1. クラス
2. インスタンス
3. オブジェクト

これではいかんと思い何とか調べてみました！

最初にタイ焼きをイメージして
1. クラス：タイ焼きの鋳型
2. インスタンス：作成したタイ焼きそのもの
3. オブジェクト：抽象的な概念（クラスもインスタンスもオブジェクトの一種）

余談(クラス/インスタンス/オブジェクトの違い)

クラスとインスタンスは、タイ焼きの鋳型とタイ焼きそのもの
--クラスは開発時にツラツラ書くコードそのもの
--インスタンスは実際にクラスが動く時に、**newして**メモリ上に載ったもの
※newした数分インスタンスが作成される（下のタイ焼きも3つ!!）