# Java言語で学ぶデザインパターン入門

| | | | |
|---|---|---|---|
| **ノートブック:** | 240_開発リソース | | |
| **作成:** | 2015/10/24 20:45 | **更新:** | 2019/07/19 21:44 |
| **作成者:** | Tsutsui tomoaki | | |

*-*-*-* デザインパターンに慣れる *-*-*-*-*
◆Iterator
一つ一つ数え上げる

◆Adapter
一皮かぶせて再利用
既存のクラスには全く手を加えず、
目的のインターフェース(ＡＰＩ)に合わせようとするもの

*-*-*-* サブクラスに任せる *-*-*-*-*
◆Template Method
具体的な処理をサブクラスに任せる
superなabstractクラスを定義
処理内容はサブクラスで実装する
サブクラスはスーパークラスのメソッドを実装する責任を負う
subclass responsibility

◆Factory Method
インスタンス作成をサブクラスに任せる

*-*-*-* インスタンスを作る *-*-*-*-*
◆Singleton
たった一つのインスタンス
一つしか存在しないことを保証したい

◆Prototype （原型・模範)
コピーしてインスタンスを作る
クラスからインスタンスを生成するのではなく、インスタンスから別のインスタンスを作り出すパターン
Java言語では複製を作る操作を「clone」と呼んでいます。
--Prototype(原型)の役
cloneableインターフェースを実装してないといけない

※concrete(コンクリート)：具体的な。具体物

◆Builder
複雑なインスタンスを組み立てる

◆Abstract Factory
関連する部品を組み合わせて製品を作る

*-*-*-* 構造を渡り歩く *-*-*-*-*
◆Visitor
構造を渡り歩きながら仕事をする

◆Chain of Responsibility

責任のたらい回し

*-*-*-* シンプルにする *-*-*-*-*
◆Facade
シンプルな窓口

◆Mediator
相手は相談役一人だけ
＞相談役を作る
--役割
「相談役」mediator(調停者)
「各メンバー」colleague(同僚)

～ログインのGUIフレームを作成して学ぶ～

*-*-*-* 状態を管理する *-*-*-*-*
◆Observer
状態の変化を通知する
＞観察者
--役割
「観察される側」Subject(被験者)
「観察される側」ConcreteSubject(具体的な被験者)
「観察する側」Observer(観察者)
「観察する側」ConcreteObserver(具体的な観察者)

subject役はupdateメソッドを実装して、Observer役に状態の変更を通知する！

＋「観察」よりも「通知」になっている
本来の意味は観察者ですが、実際にはObserver役は能動的に観察するのではなく、
Subject役から通知されるのを受動的に待っていることになります。
Publish-Subscribeパターンと呼ばれることもある！！
こちらの呼ばれ方のほうが実体にあっているかも知れません。

◆Memento
状態を保存する
＞状態を保存する
インスタンスの状態を表す役割を導入して、カプセル化の破壊に陥ることなく
保存と復元を行う

Memento：「記念品」「形見」「思い出の種」
Memento パターンを利用すると
・undo「やり直し」
・redo「再実行」
・history「作業履歴の作成」
・snapshot「現在状態の保存」

◆State
状態をクラスとして表現する

*-*-*-* 無駄をなくす *-*-*-*-*
◆Flyweight
同じものを共有して無駄をなくす

◆Proxy
必要になってから作る

*-*-*-* クラスで表現する *-*-*-*-*
◆Command
命令をクラスにする

◆Interpreter
文法規則をクラスで表現する