

パターンクラス図コレクション

Ver. 1.0 (1999/10/16)

このパターンクラス図コレクションは、ソフトバンク・パブリッシング社刊の月刊誌「Cマガジン」1999年4月号から11月号まで著者(大城正典)が連載した「JavaとC++によるデザインパターン」の補足として作成したものです。よく知られるようになったパターンをUML記法で記述しています。

【注意】

このパターンクラス図コレクションに収録されているパターンに関する説明は、「Cマガジン」の上記連載記事を参照して下さい。誤字脱字など、お気づきの点がありましたら著者までe-mail(GAH03155@nifty.ne.jp)にてご連絡下さると助かります。なお、このパターンクラス図コレクションに関するその他のご質問は、著者もCマガジン編集部もお答え致しかねますので、ご了承下さい。

このパターンクラス図コレクションの著作権は、著作者の私(大城正典)に帰属します。個人的な学習のための複製は許可しますが、ネットワーク、CD-ROMなどの記録メディアおよび印刷媒体を介しての登録・公開・複製を行うことは、これを固く禁止致します。

1999年10月16日 大城正典

基本構造

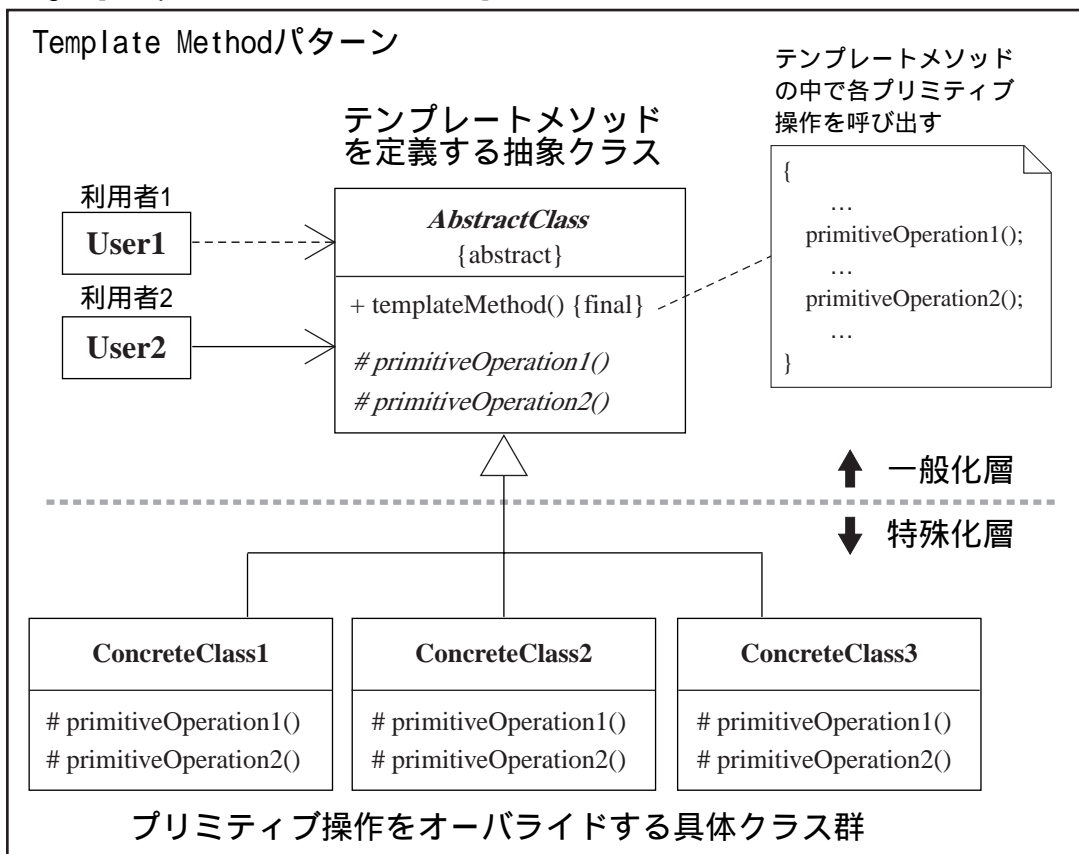
Template Methodパターン

自己参照型パターン

Strategyパターン

- 多重度0以上の自己参照
Compositeパターン, Interpreterパターン
- 多重度1以下の自己参照
Decoratorパターン, Chain of Responsibilityパターン, Proxyパターン

Fig.1[Template Methodパターン]



finalとは、“サブクラスでオーバーライドされない”という意味で使用しています。

Fig.2[Strategyパターン]

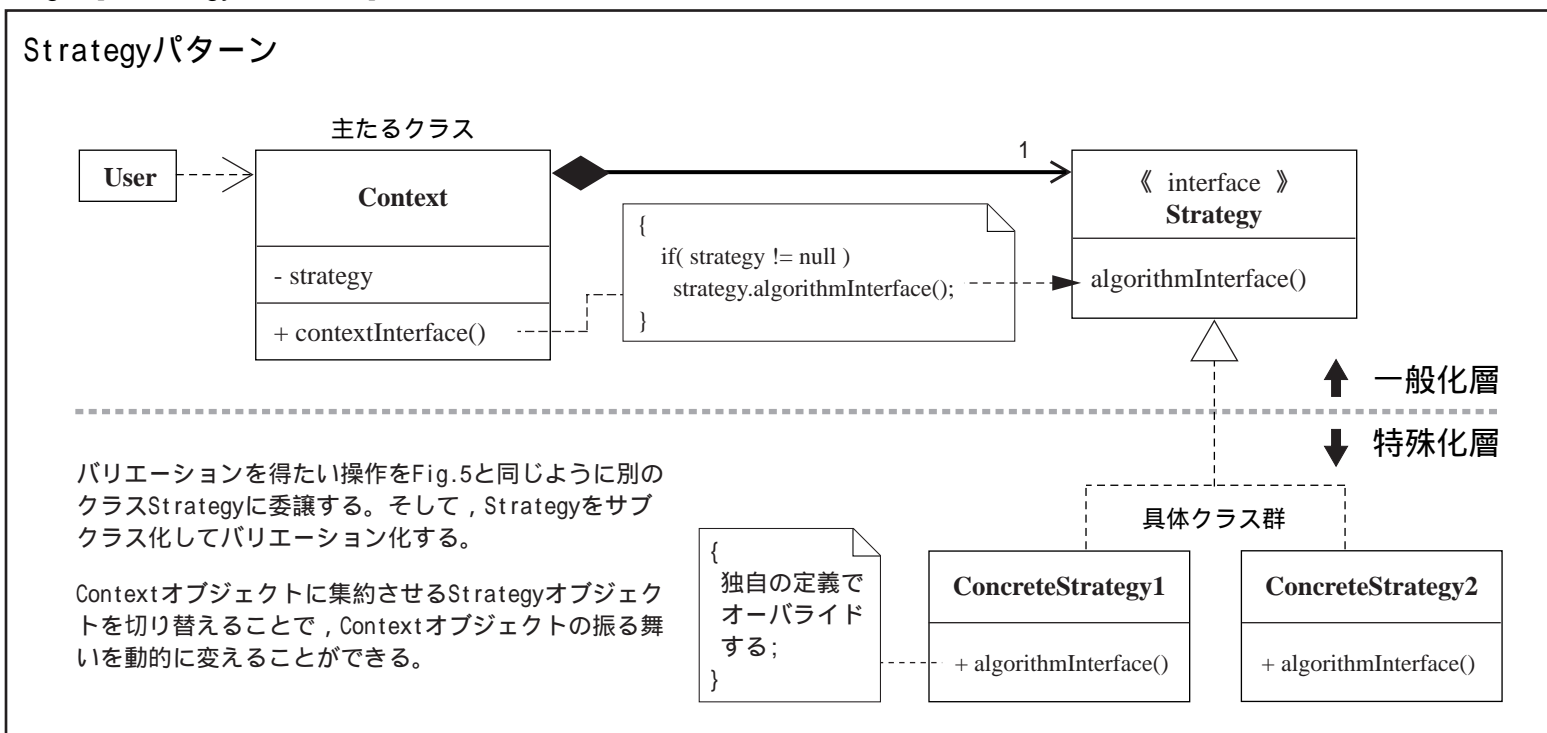


Fig.3[Compositeパターン(透過性優先版)]

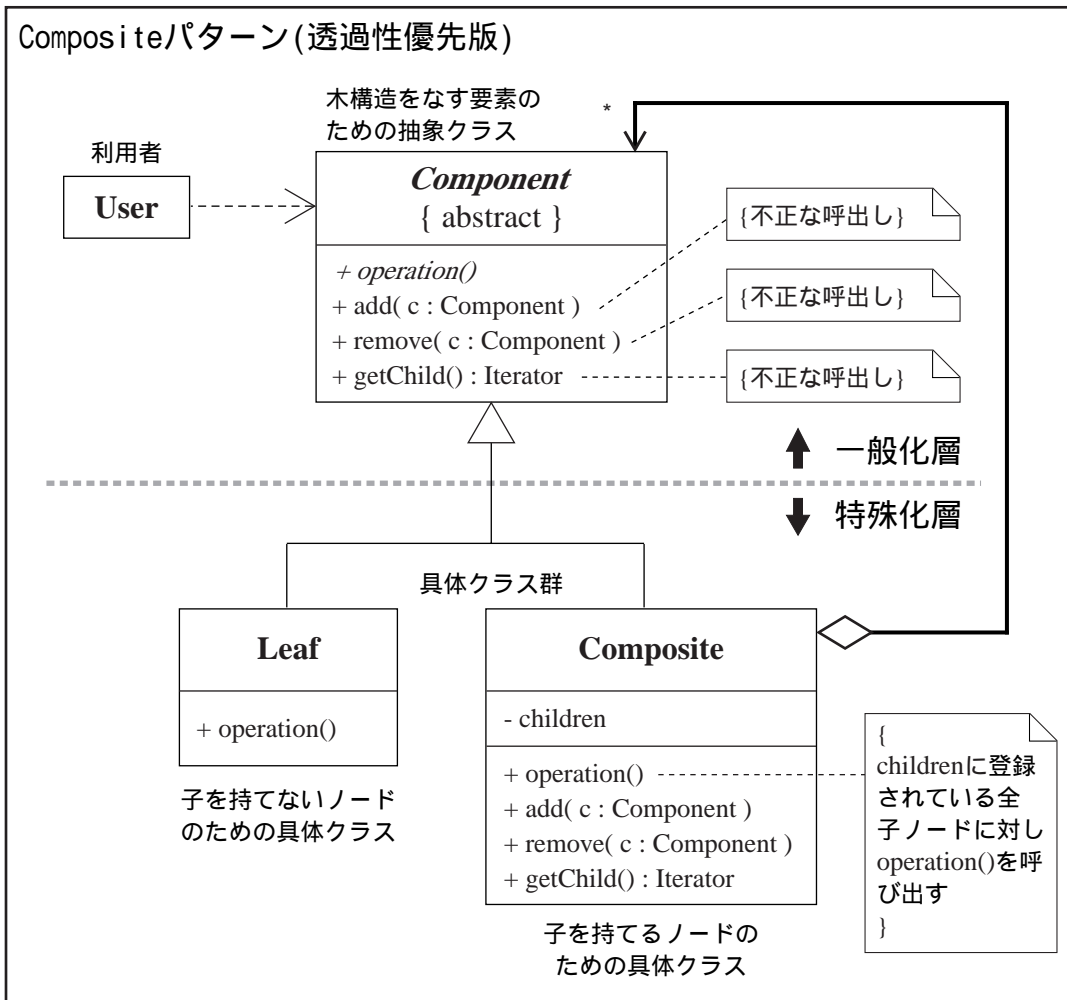


Fig.4[Compositeパターン(安全性優先版)]

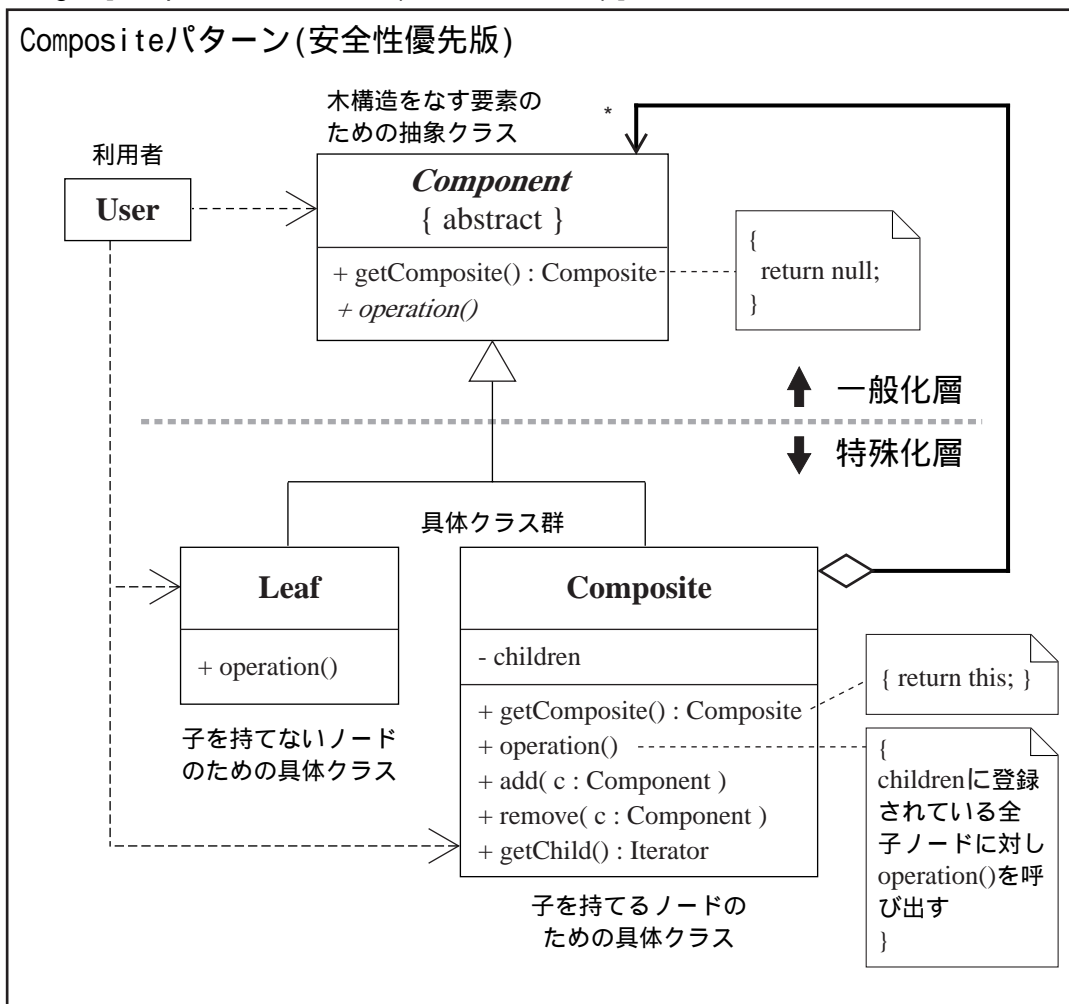


Fig.5[Interpreterパターン]

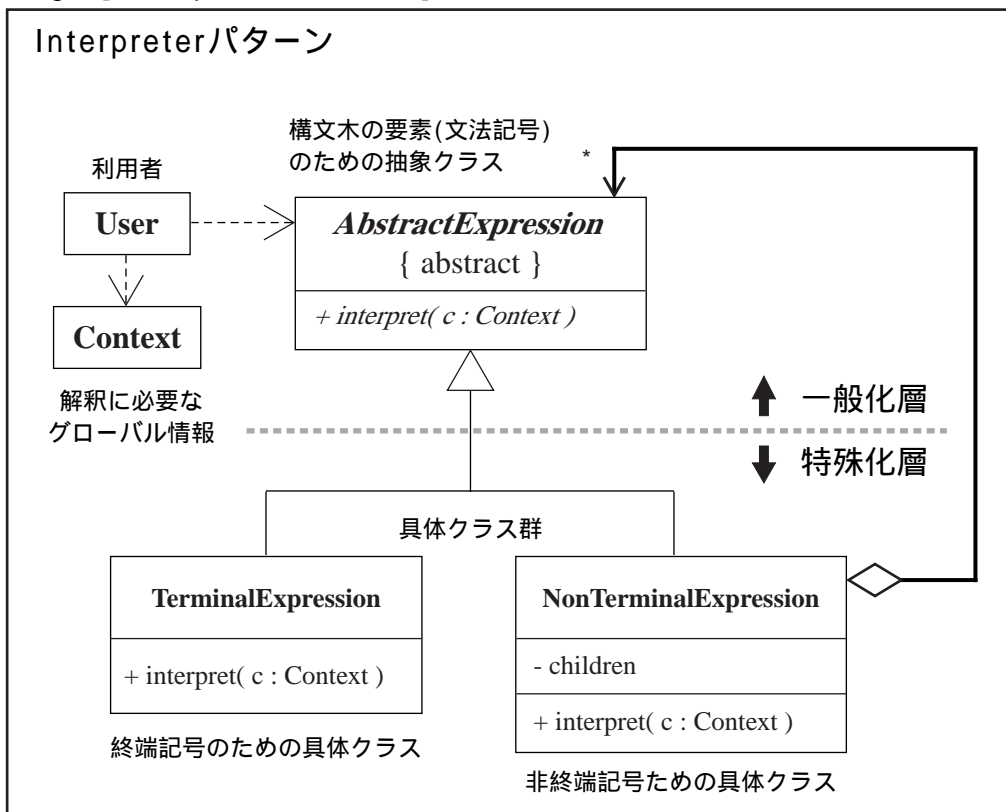


Fig.6[Decoratorパターン]

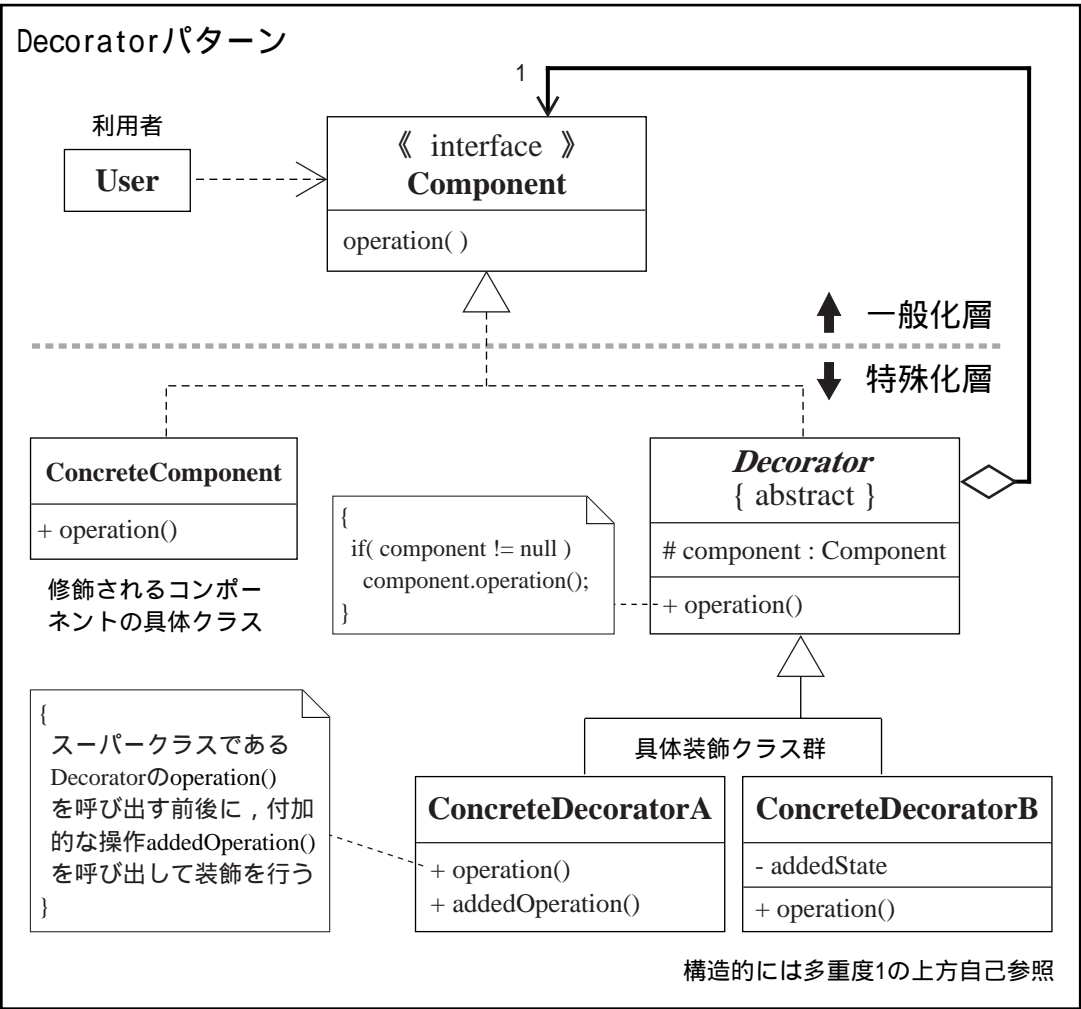


Fig.7[Chain of Responsibilityパターン]

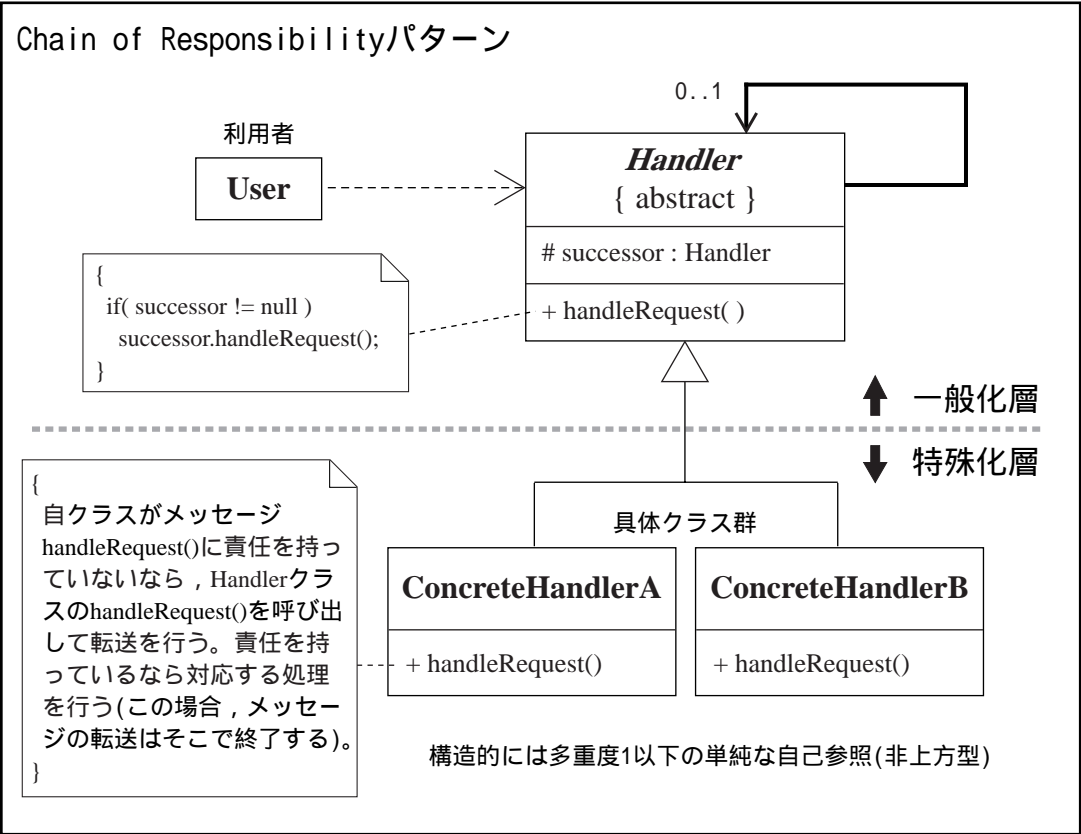
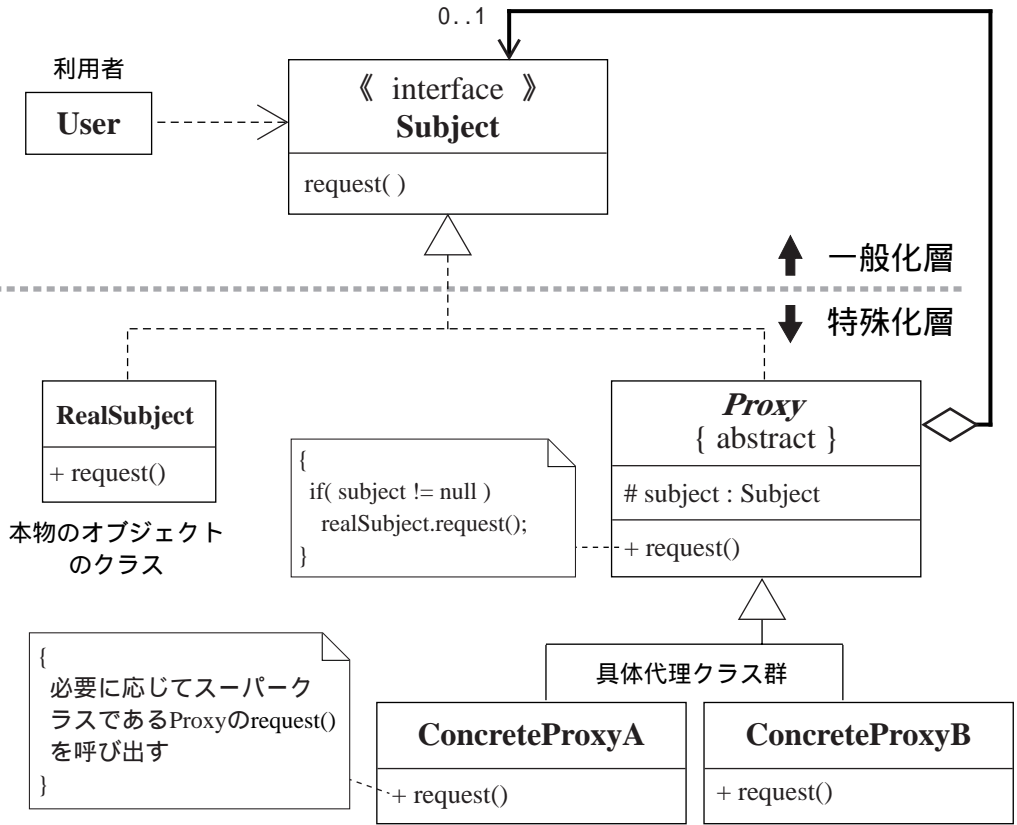


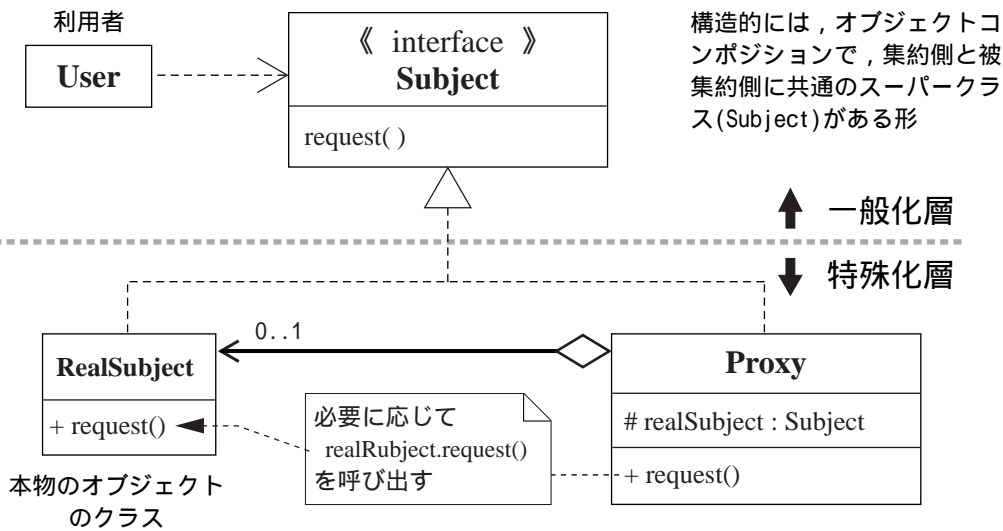
Fig.8[Proxyパターン]

Proxyパターン

(a)複数の代理オブジェクトを連鎖させる場合(多重度1の上方自己参照)



(b)代理オブジェクトを1個に限定する場合(自己参照ではない)



オブジェクト生成パターン

Factory Methodパターン

コンポジション・パターン

Abstract Factoryパターン, Builderパターン, Prototypeパターン

Singletonパターン

Singletonパターンのクラス図は用意していません

Fig.9[Factory Methodパターン]

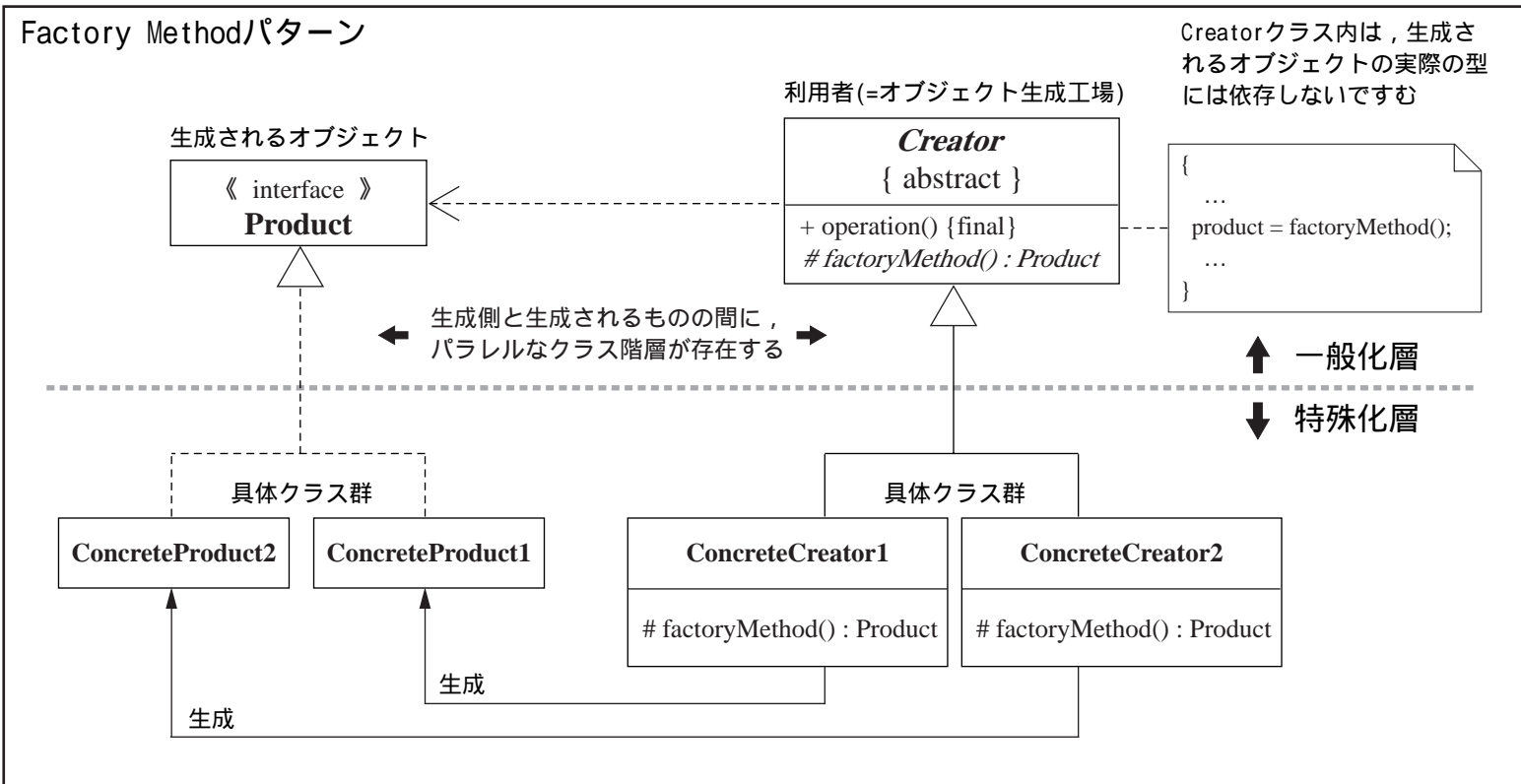


Fig.10[Abstract Factoryパターン]

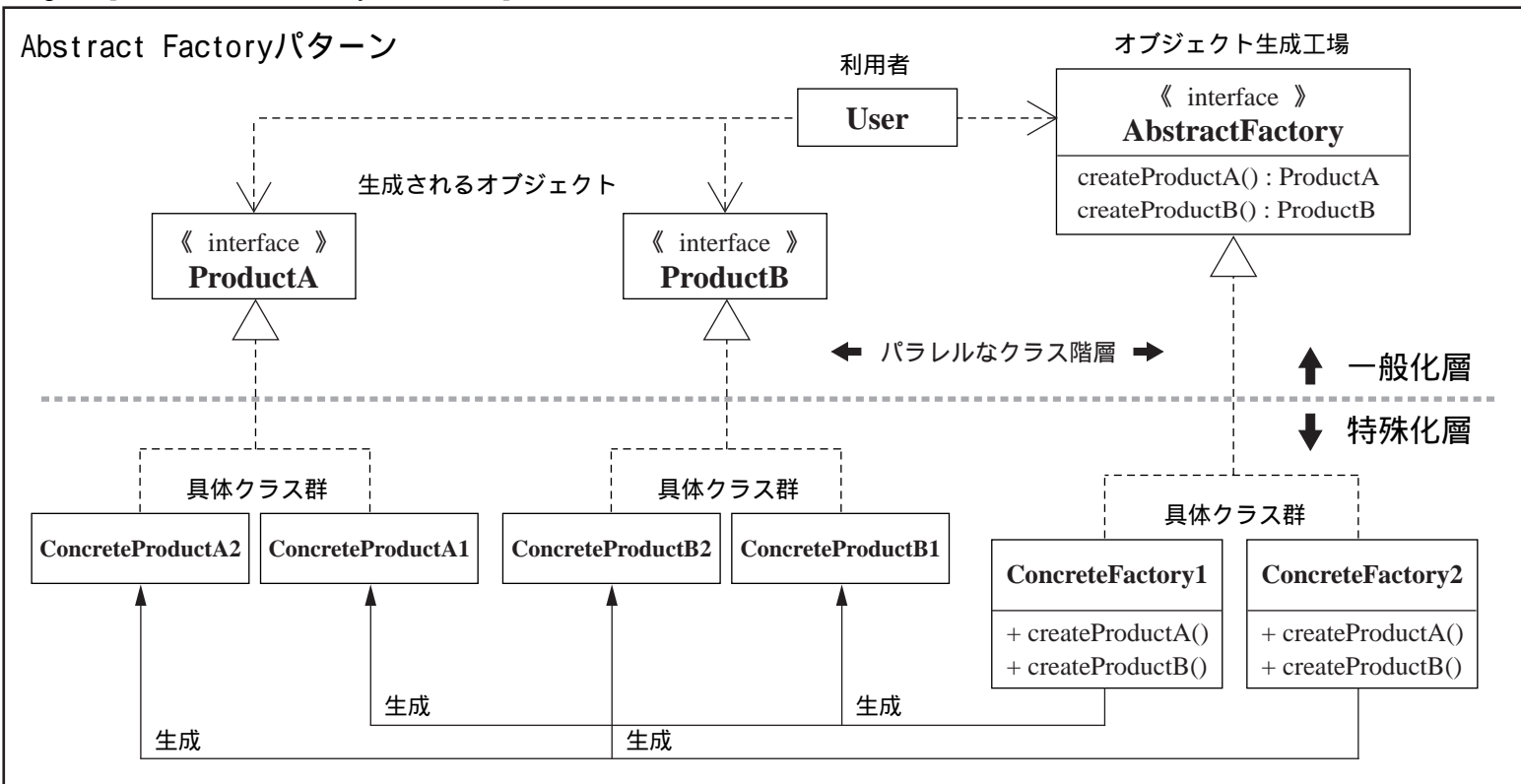


Fig.11[Builderパターン]

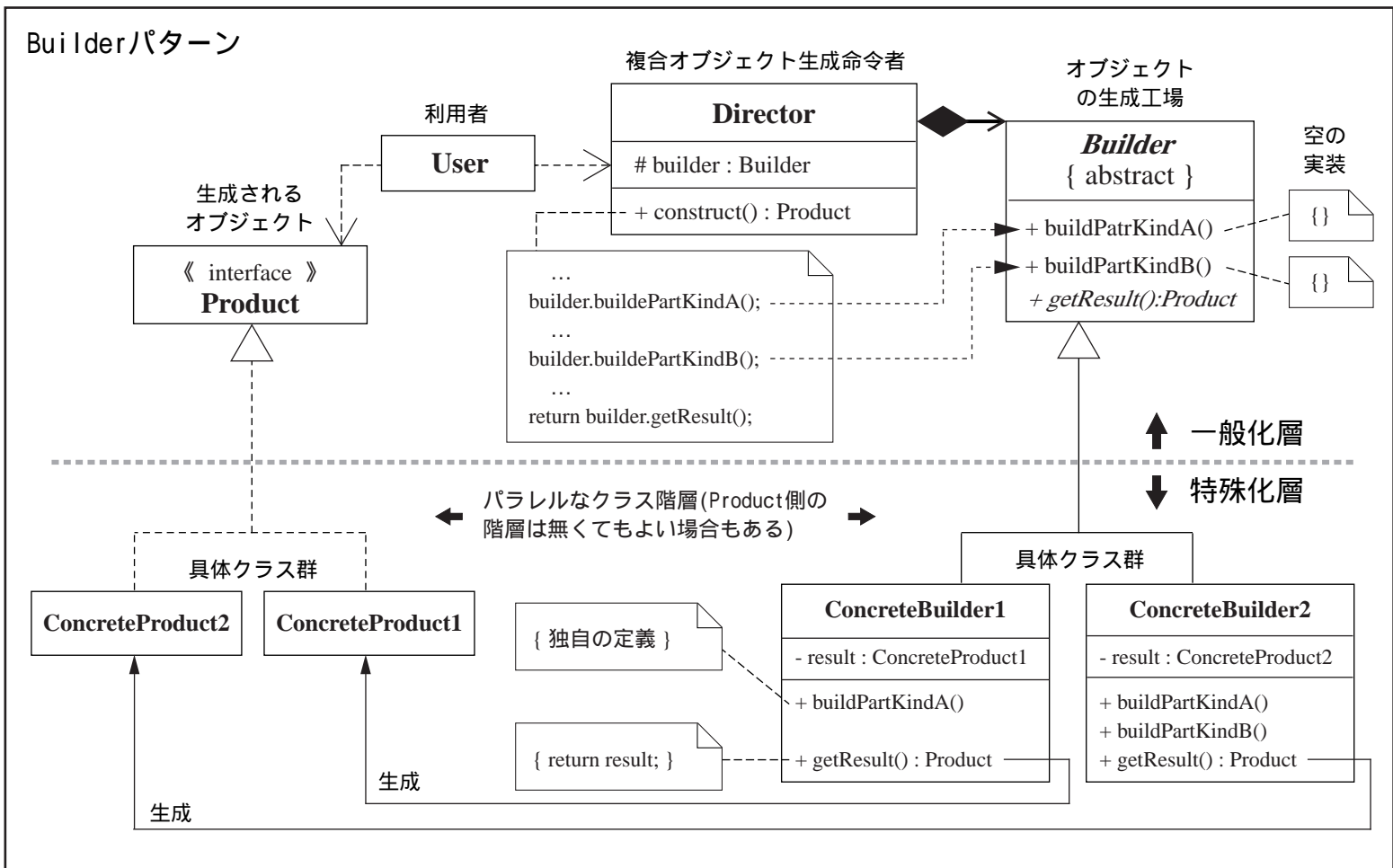
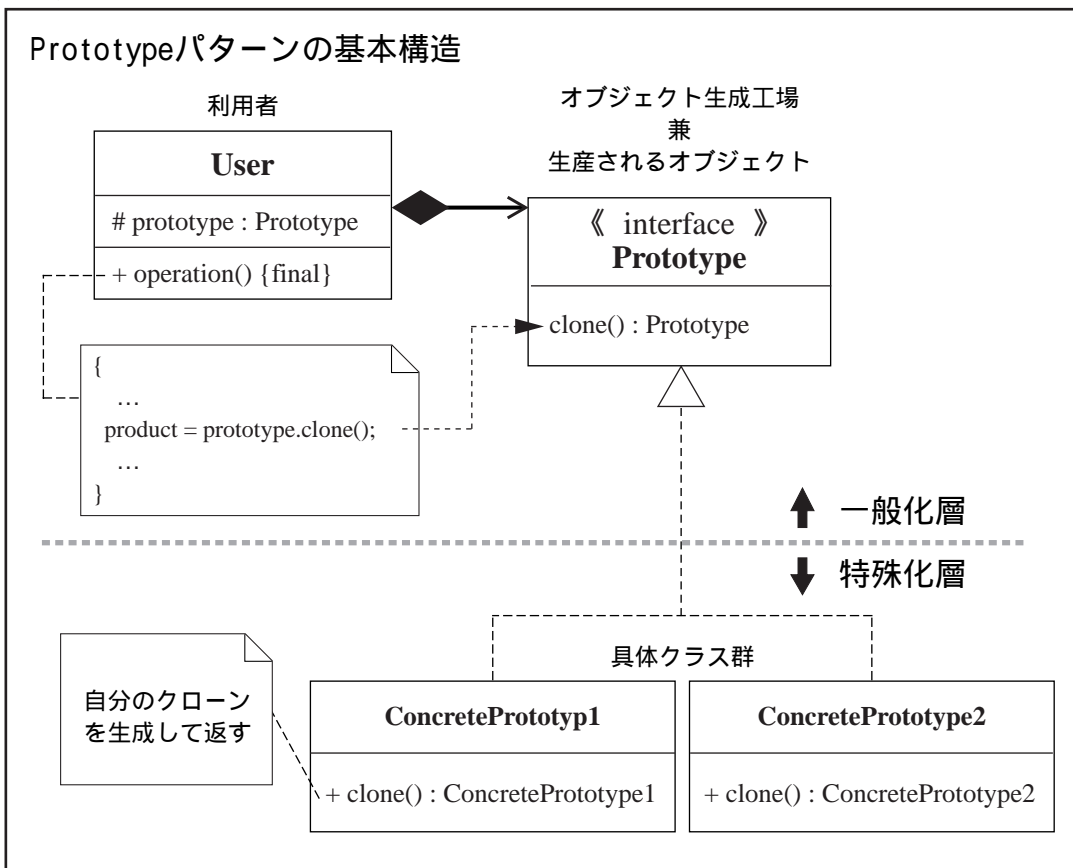


Fig.12[Prototypeパターンの基本構造]



インタフェースを変更する媒介物

Adaptorパターン

Bridgeパターン

Iteratorパターン

Fig.13[Adaptorパターン(多重継承版)]

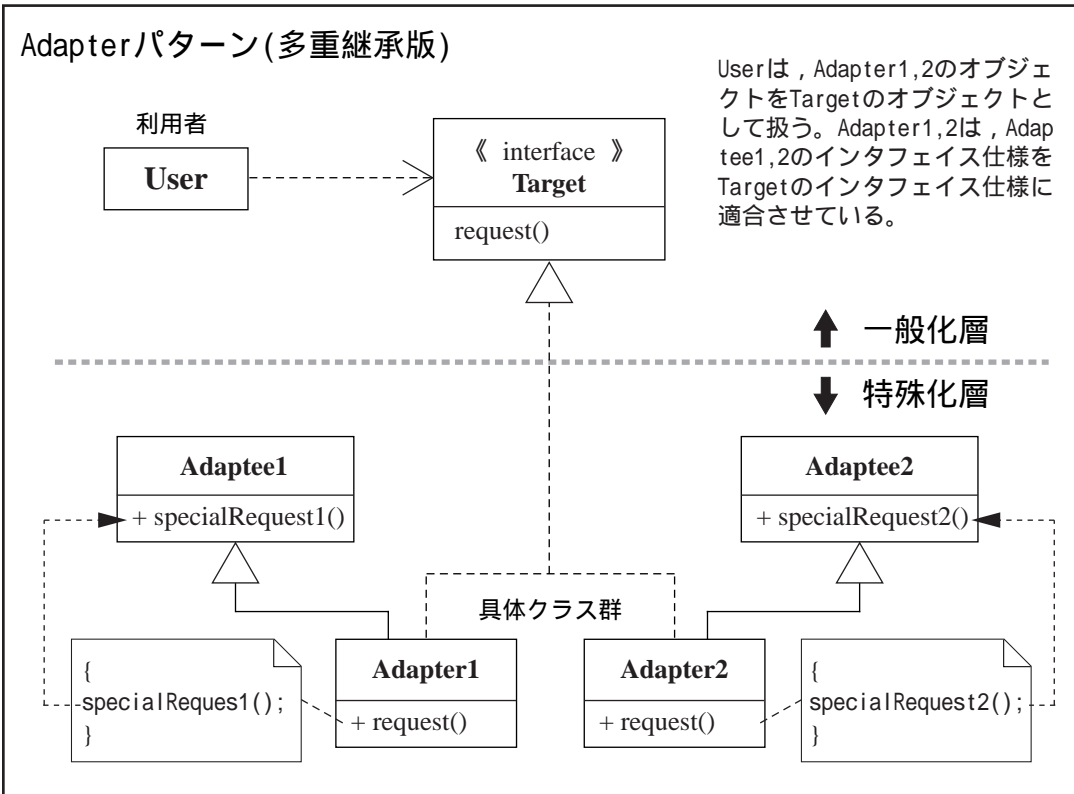


Fig.14[Adaptorパターン(オブジェクトコンポジション版)]

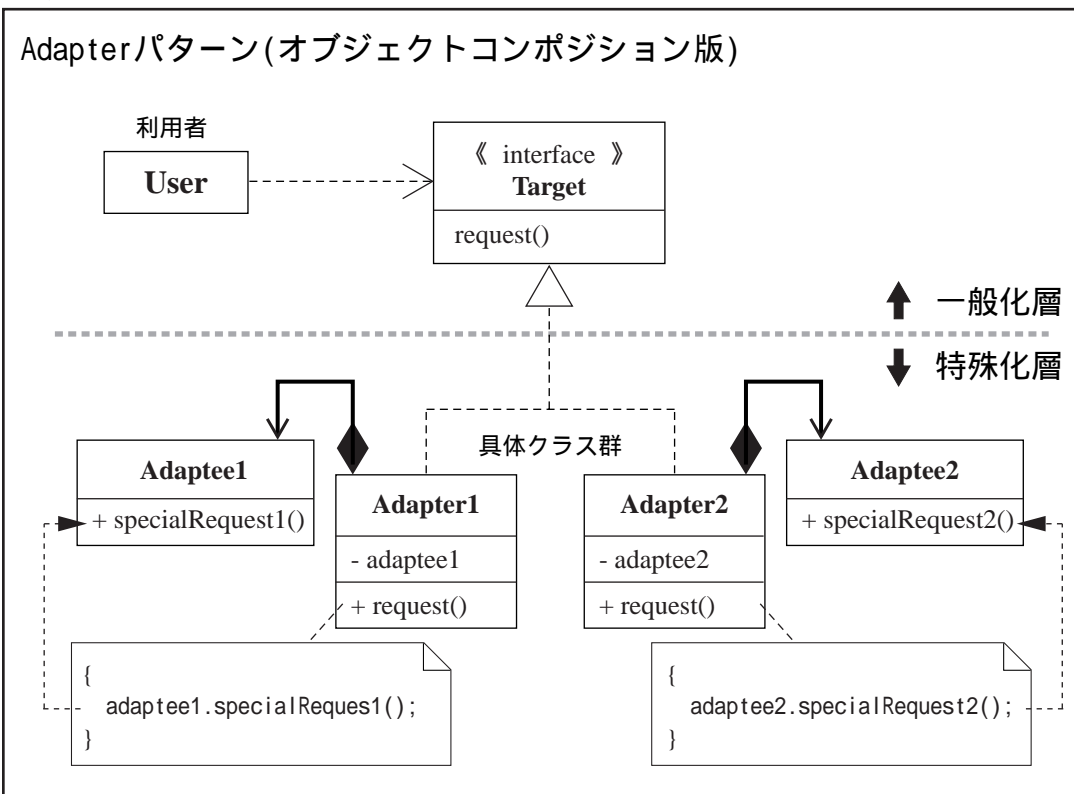


Fig.15[Bridgeパターン]

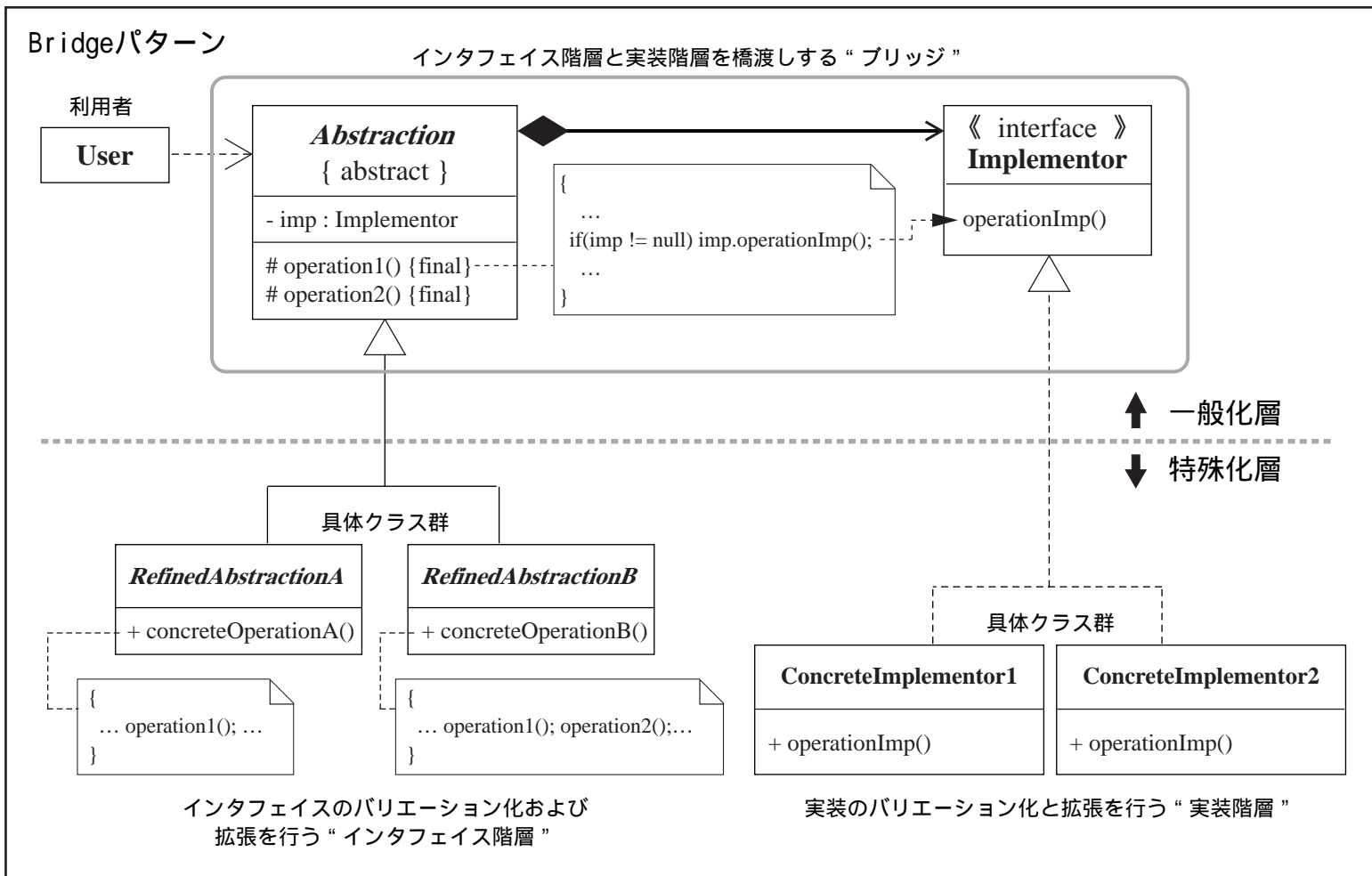
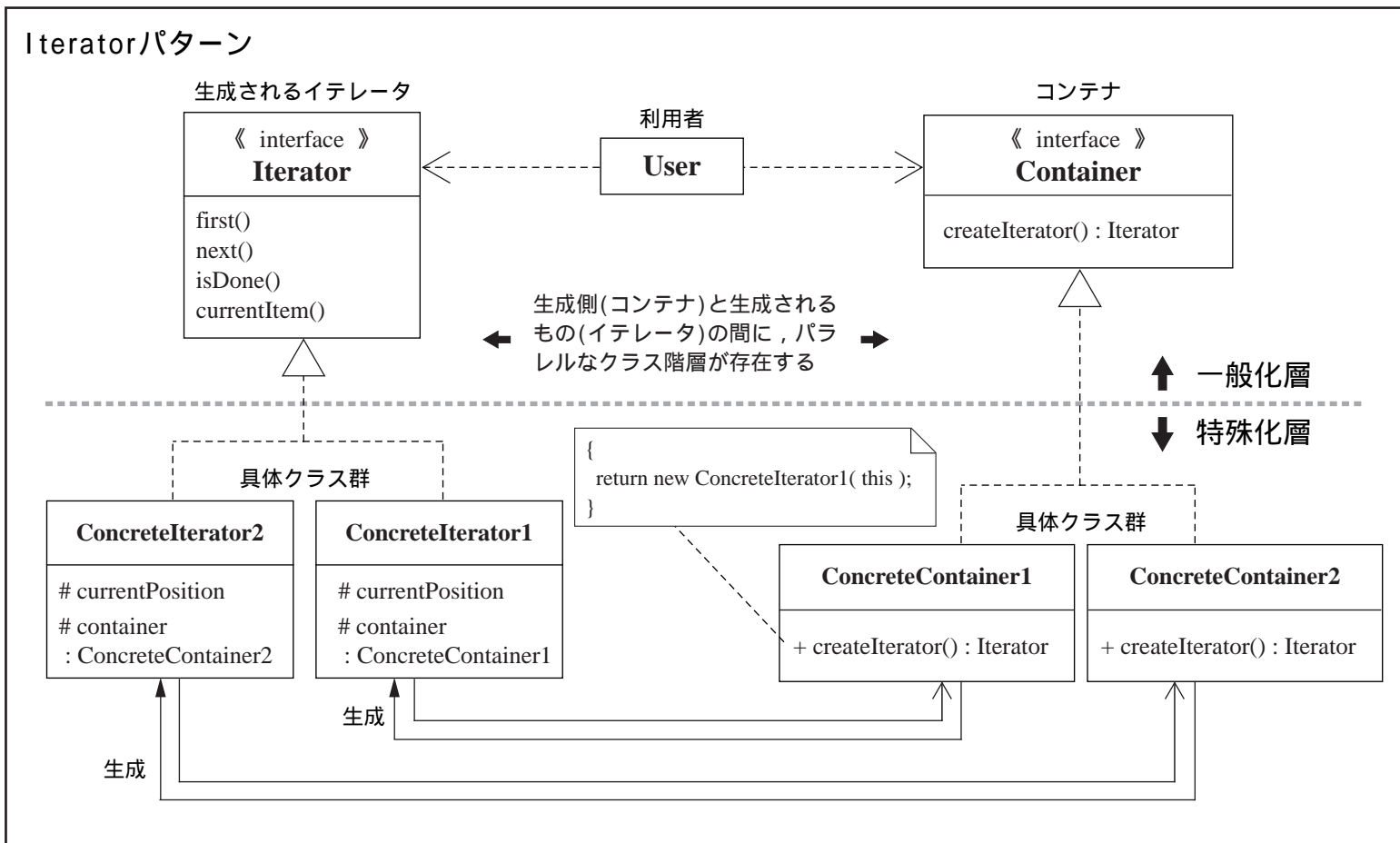


Fig.16[Iteratorパターン]



状態に関するパターン

Flyweightパターン

Stateパターン

Observerパターン

Mementoパターン

Fig.17[Flyweightパターン]

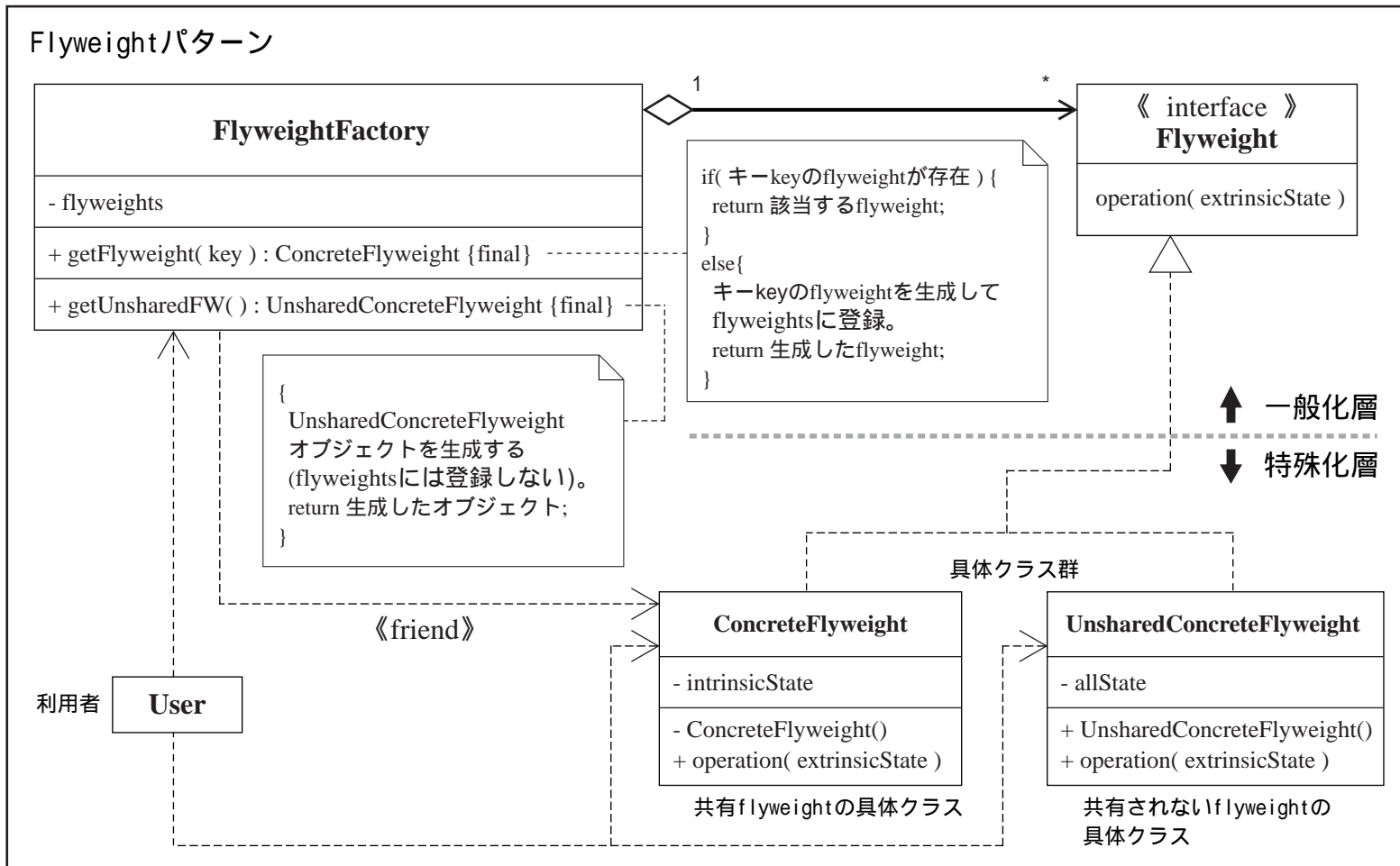


Fig.18[Stateパターン]

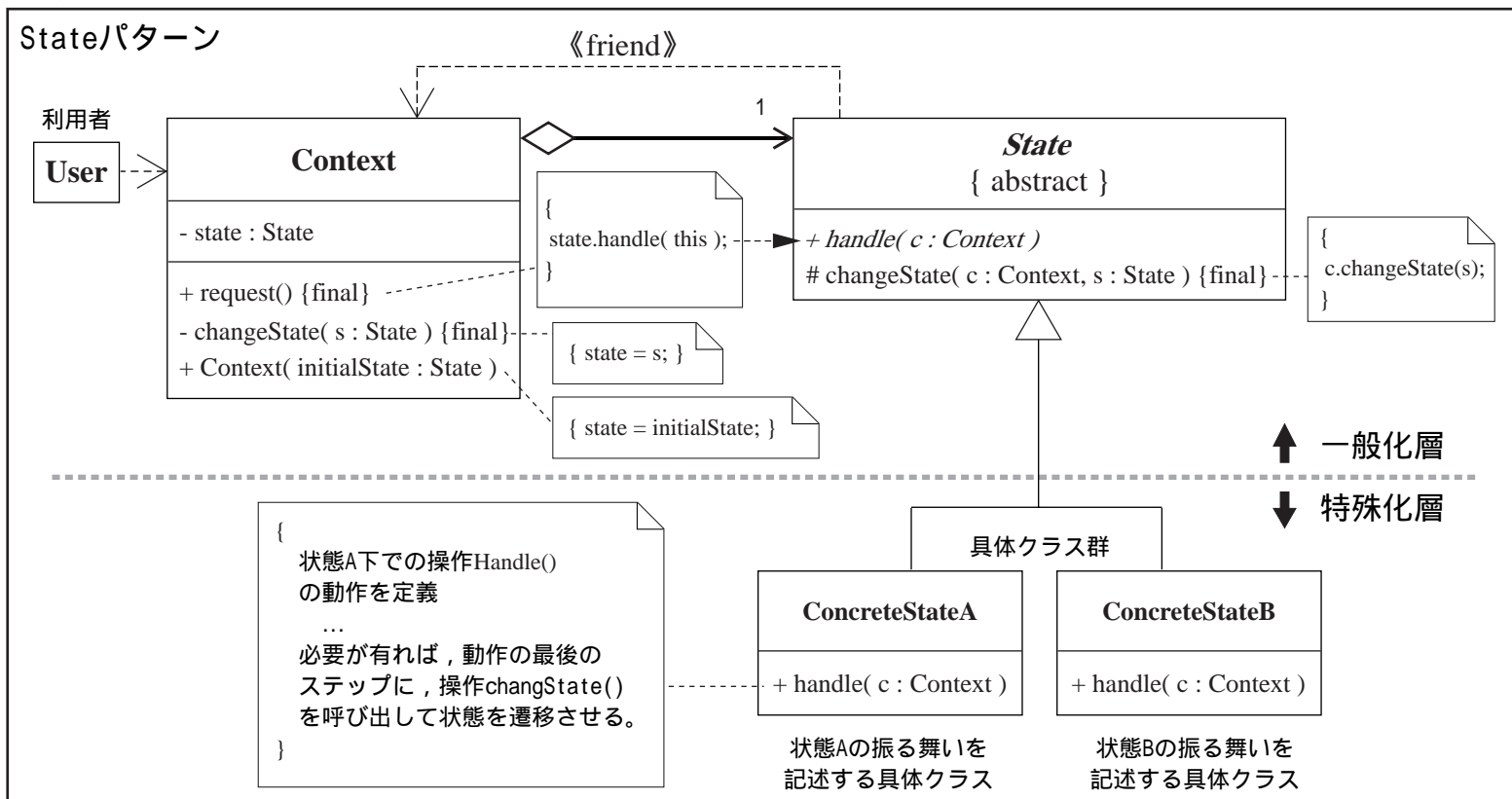


Fig.19[Observerパターン]

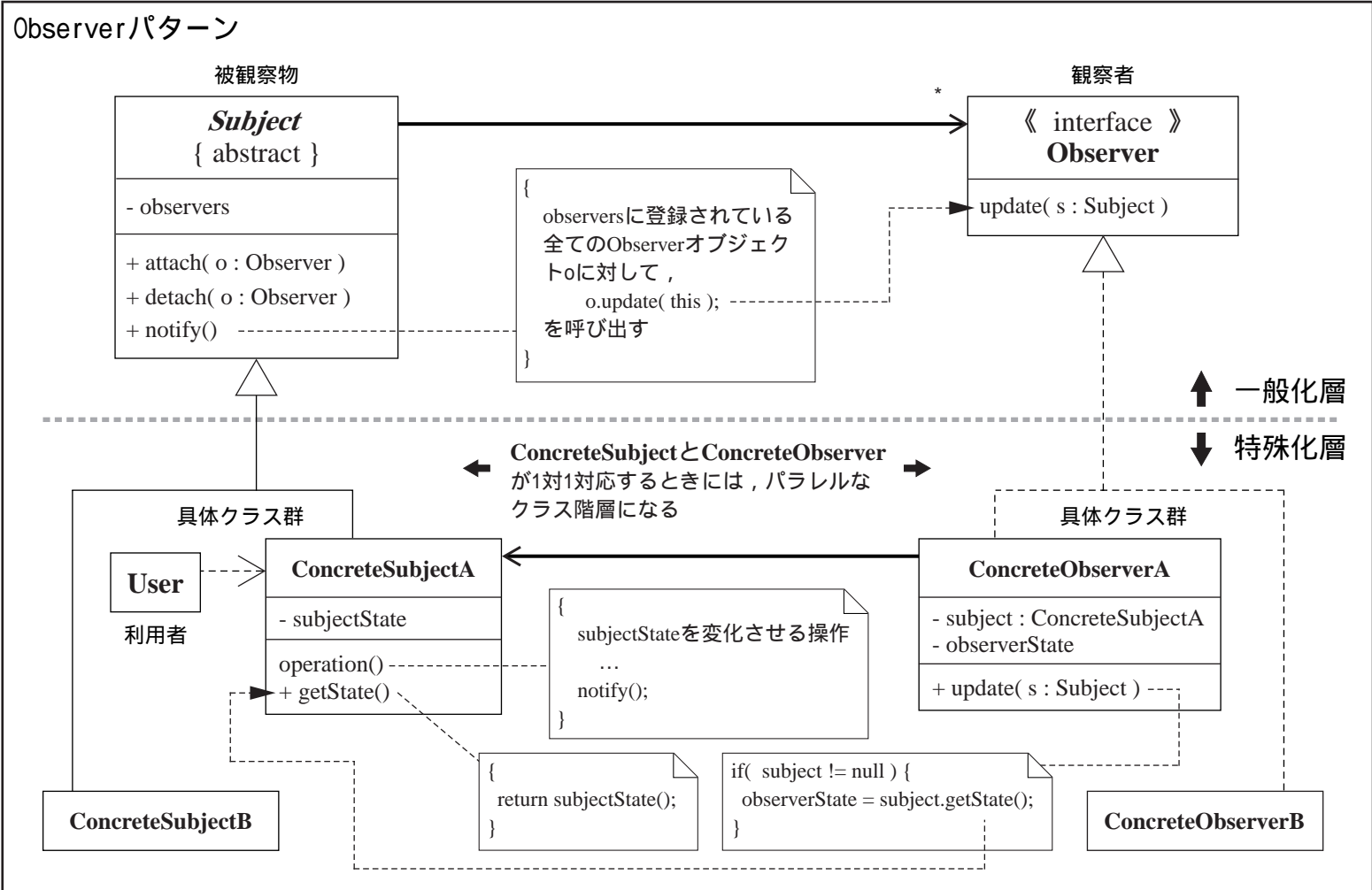
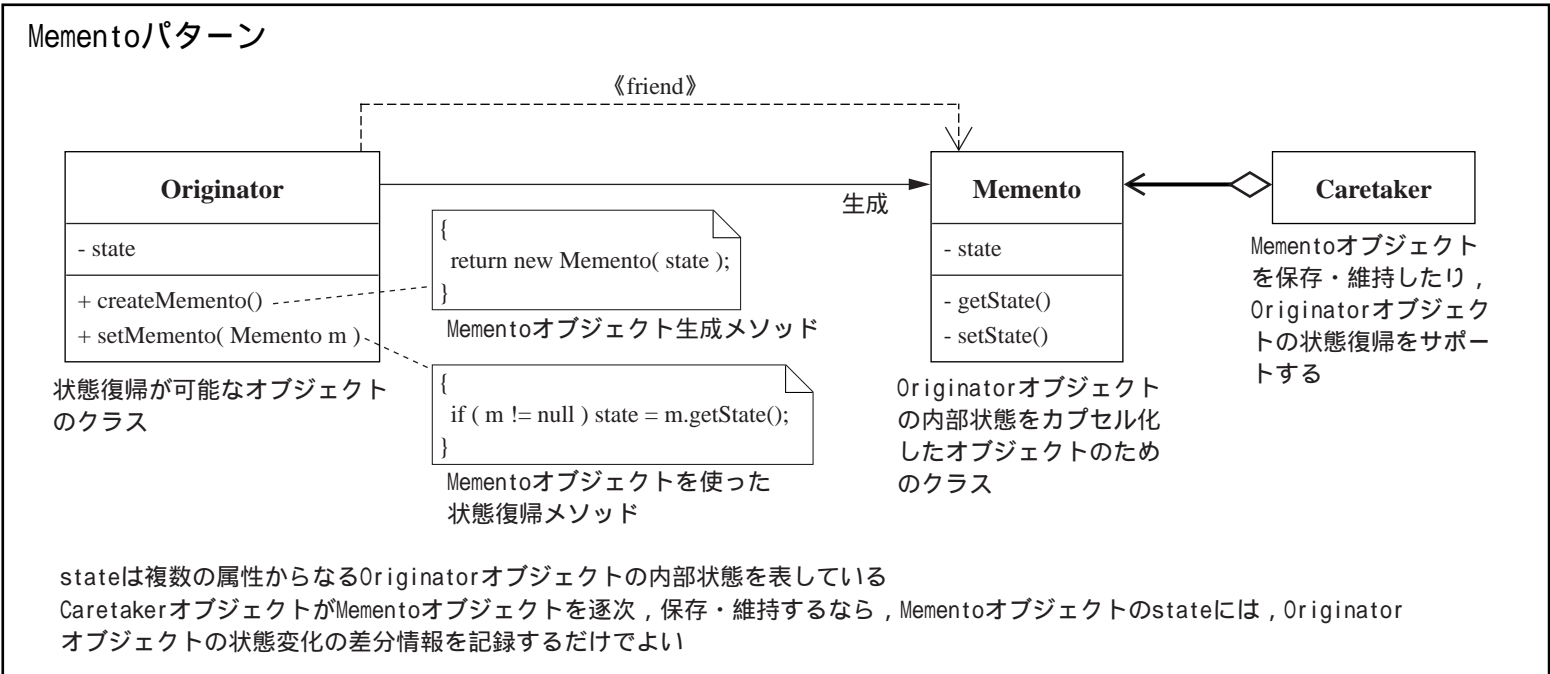


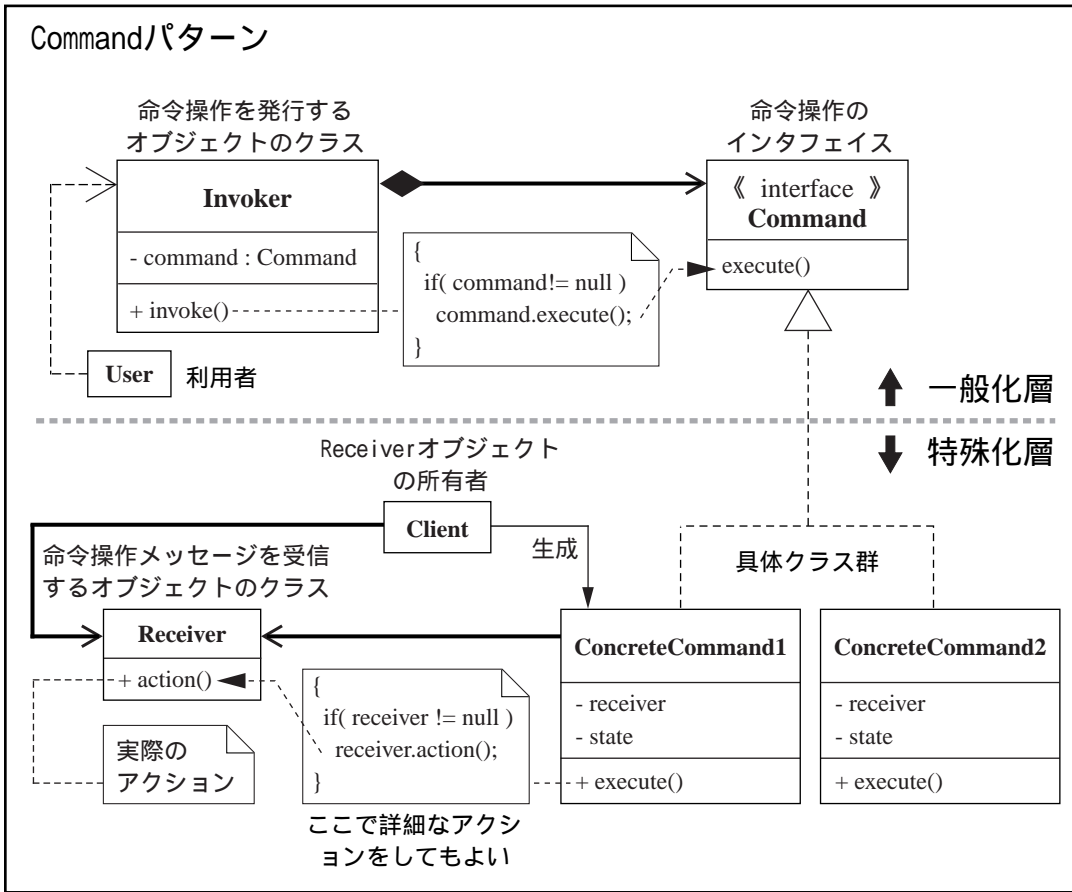
Fig.20[Mementoパターン]



命令のオブジェクト化

Commandパターン

Fig.21[Commandパターン]



関係を減らすパターン

Mediatorパターン Facadeパターン

この他, Adaptorパターンなども関係を減らすために使用できる

Fig.22[Mediatorパターン]

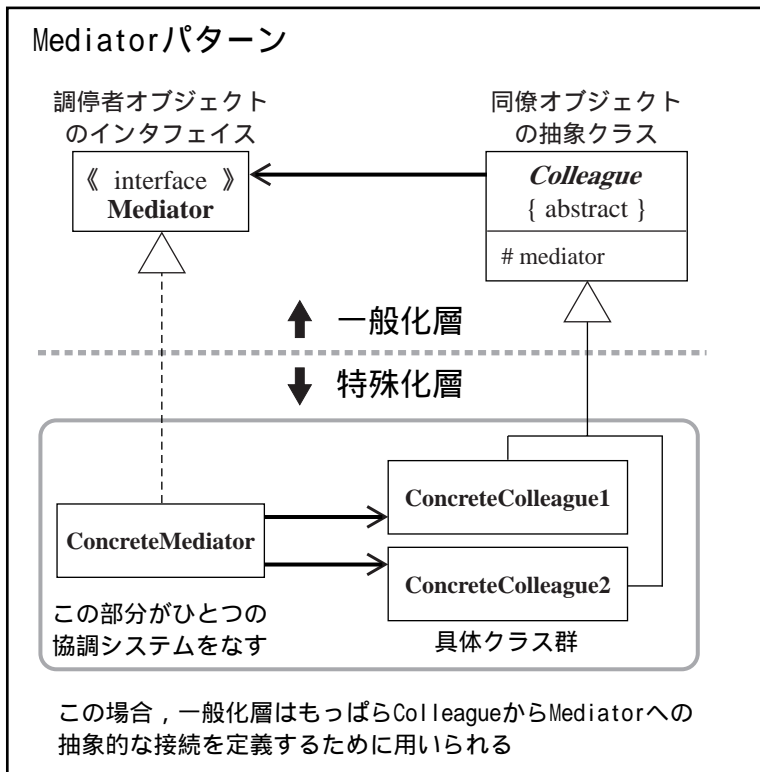
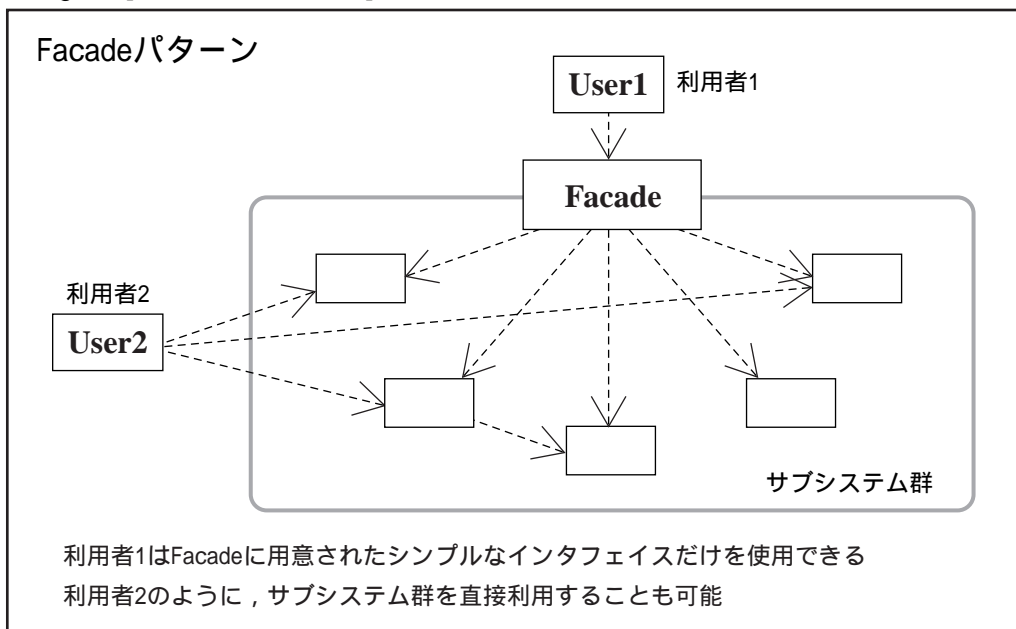


Fig.23[Facadeパターン]



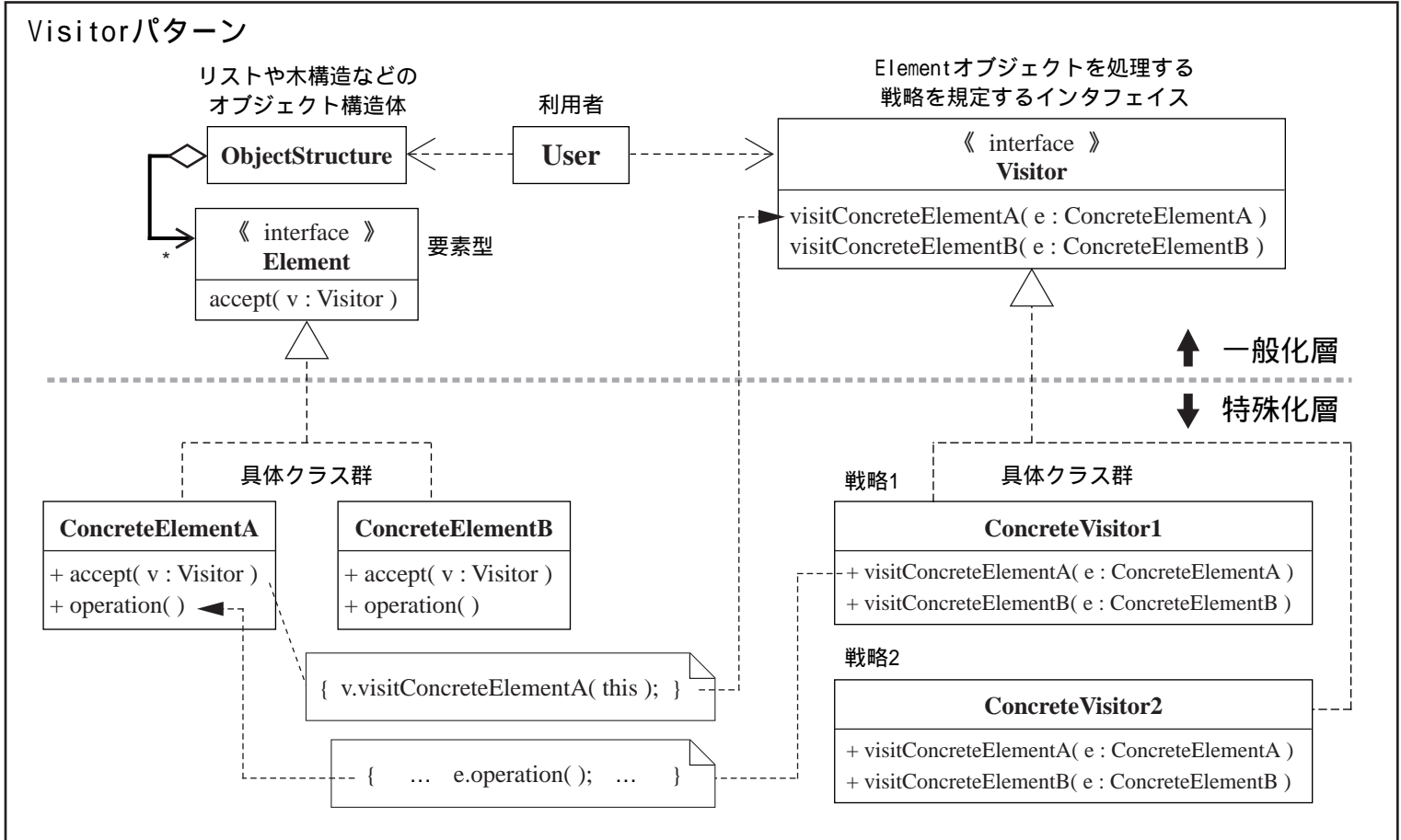
コンテナ要素への操作

Visitorパターン

Iteratorパターン

Iteratorパターンに関しては、「インタフェースを変更する媒介物」の項を参照

Fig.24[Visitorパターン]



Visitorパターンは、コンテナ要素への機能追加を目的として用いられることが多いが、主たるクラスへ複雑な機能を追加するために使用することもできる