Composite Pattern :

扱う対象が木構造のような
再帰構造をしているときに有用

他にもこのような「定石」はない？

例題

航空会社のマイレージプログラム

飛行マイルをためる（会員の口座へ）
貯めたマイルに応じてサービス有
無料航空券の獲得など

	ノーマル (Normal)	シルバー (Silver)	ゴールド (Gold)
+ボーナスマイル	100	125	150
割引率	0%	3%	5%
会員資格	<25000	<50000	<75000

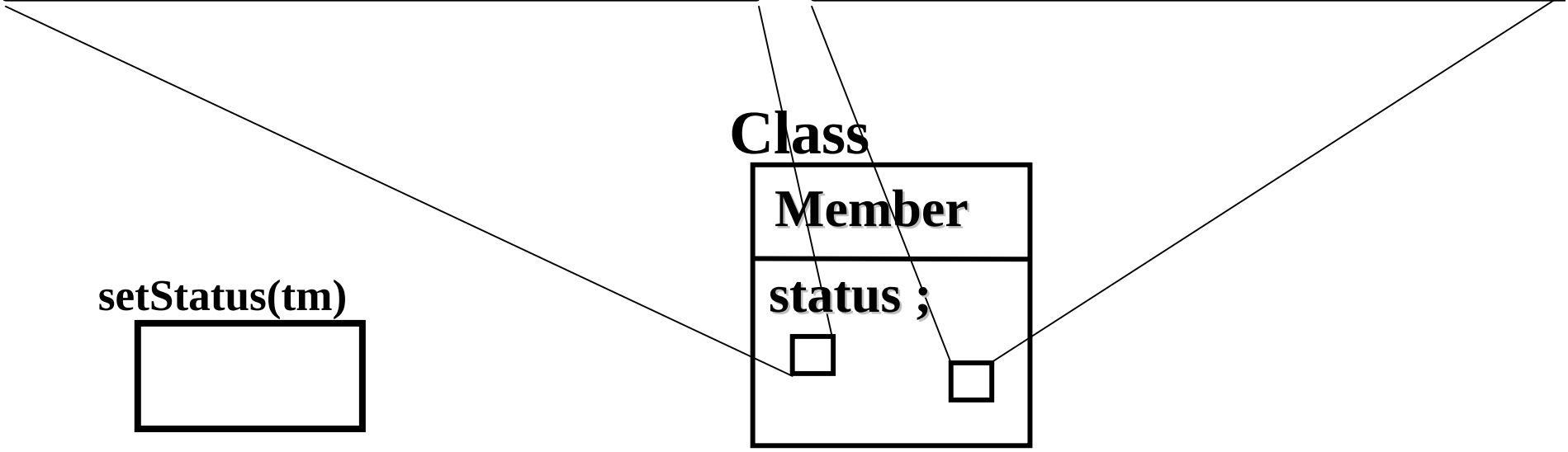
コーディング例

calculateMile(m)

```
if (status == "NORMAL"){
   return m ;
}else if (status == "SILVER"){
   return m*1.25 ;
}else return m*1.5 ;
```

getPrice(p)

```
if (status == "NORMAL"){
   return p ;
}else if (status == "SILVER"){
   return p*0.97 ;
}else return p*0.95 ;
```



変更(1)

会員のグレードを増やす
プラチナを追加

	ノーマル (Normal)	シルバー (Silver)	ゴールド (Gold)	プラチナ (Platinum)
+ボーナスマイル	100	125	150	200
割引率	0%	3%	5%	10%
会員資格	<25000	<50000	<75000	≧75000

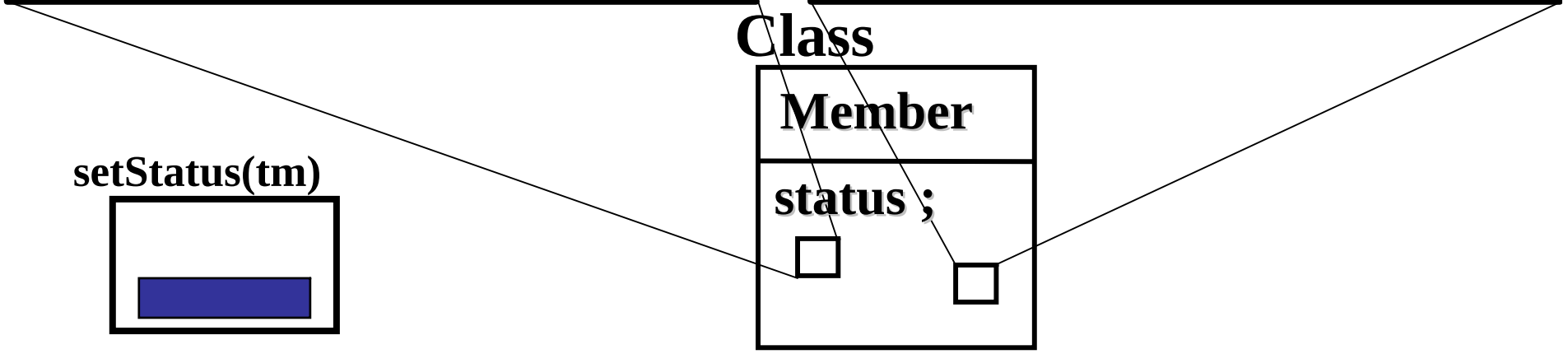
変更(2)

calculateMile(m)

```
if (status == "NORMAL"){
   return m ;
}else if (status == "SILVER"){
   return m*1.25 ;
```

getPrice(p)

```
if (status == "NORMAL"){
   return p ;
}else if (status == "SILVER"){
   return p*0.97 ;
```



問題点

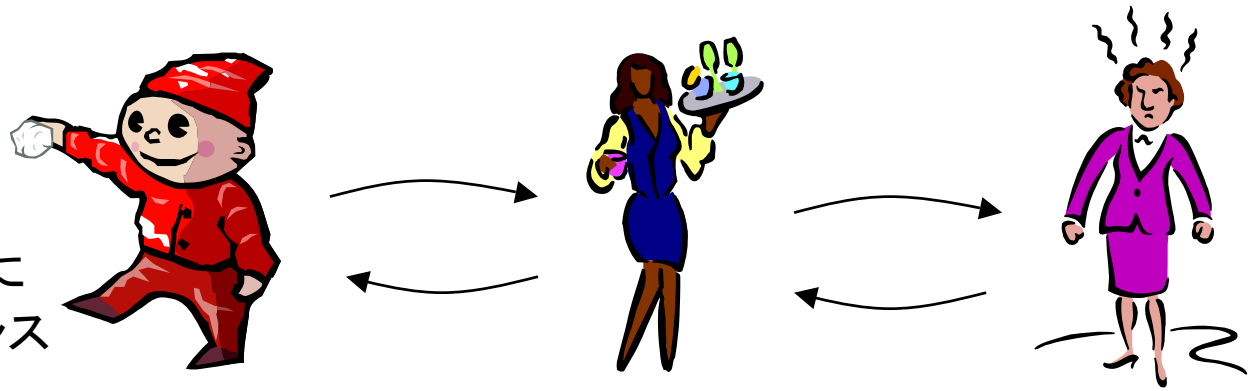
- 同じif文の構造（条件分岐）が複数のメソッドに出現
  - メンバーのstatusによって処理が異なるという構造
- ifに関連する変更や拡張が，複数のメソッドに及んでしまう.
  - メンバーのstatusの種類を増やした場合など

if文を１箇所にまとめられないか

ではどう考える？

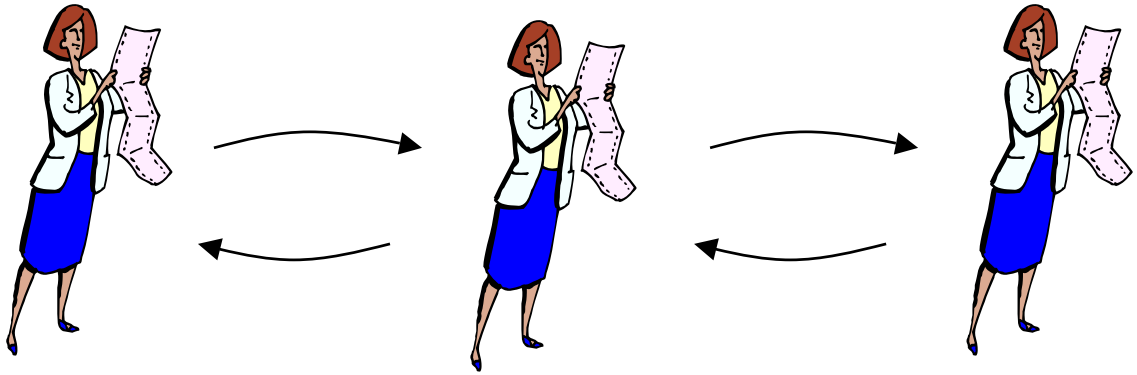
Normal, Silver, Gold, Platinumは何？　人間オブジェクトか？

Normal,
Silver,
Goldごとに
インスタンス
を生成

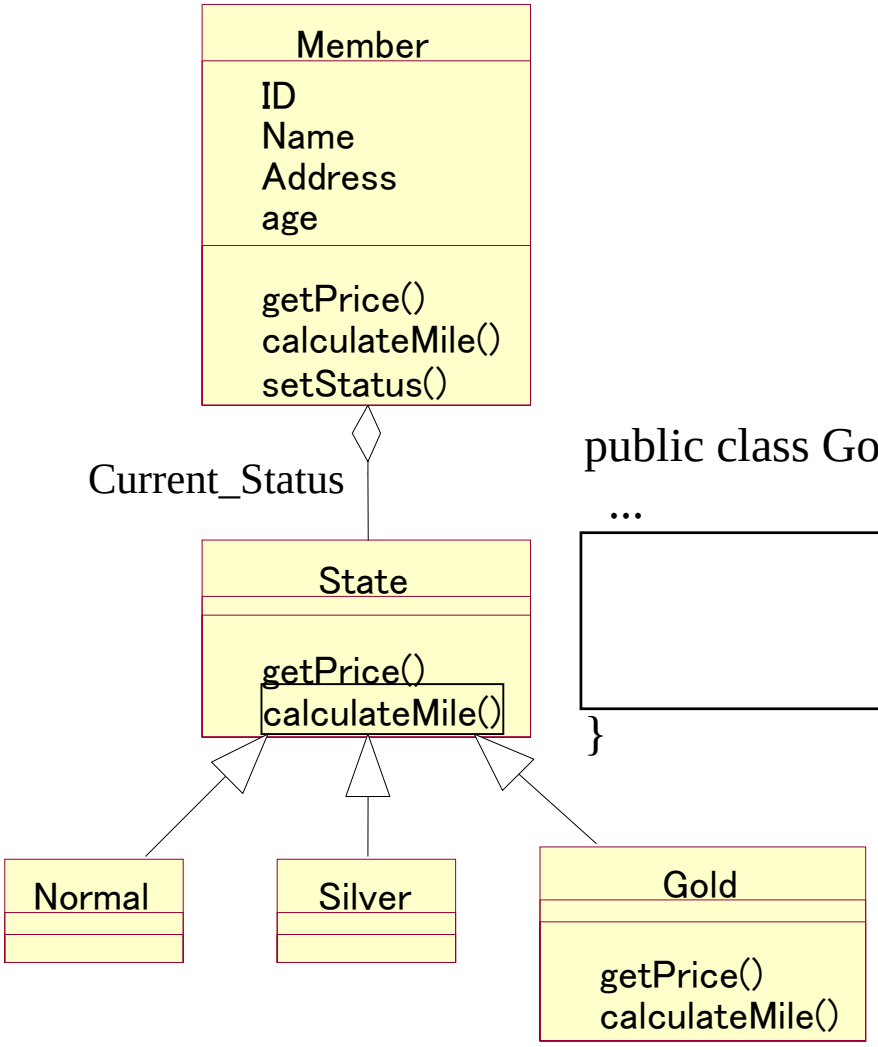


Normal　Silver　Gold

同じ人物で
「状態」が変化



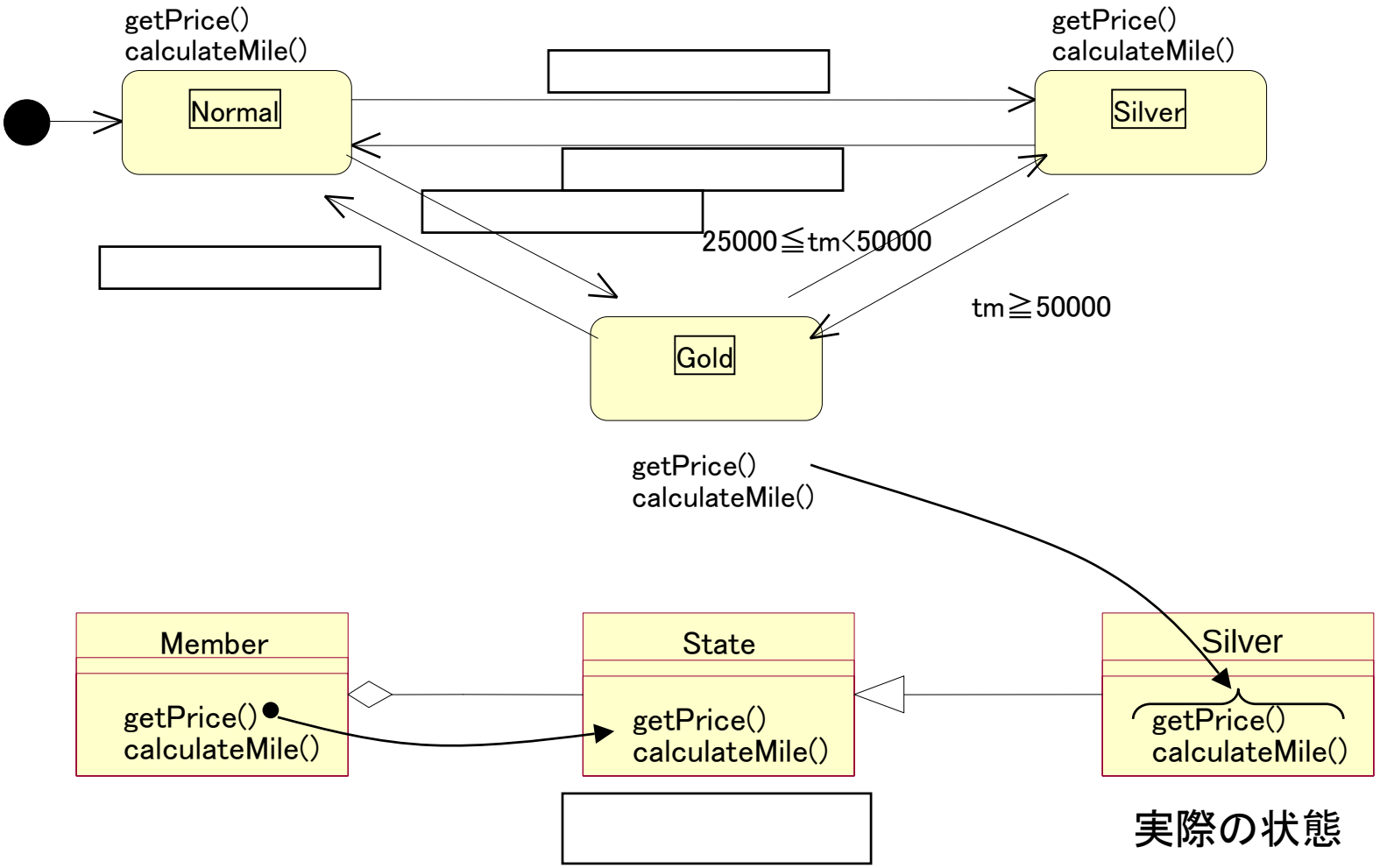
State Pattern (1)




```
public class Gold {
  ...
                                  }
```

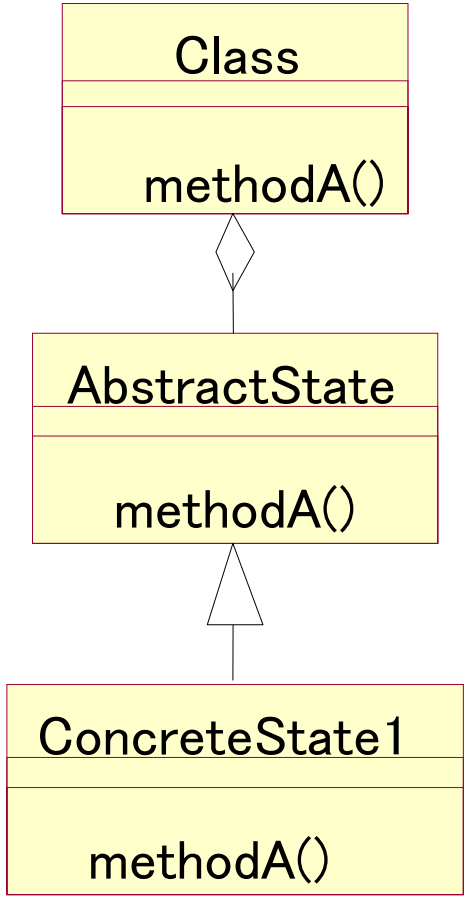
```
public class Member {
   state Current_Status ;
   ...

   public Member(int tm) {
      setStatus(tm) ;
    }

   public void setStatus(int tm) {
                                    }

   public int getPrice(int p) {
                                    }
    ...
}
```

State Pattern (2)

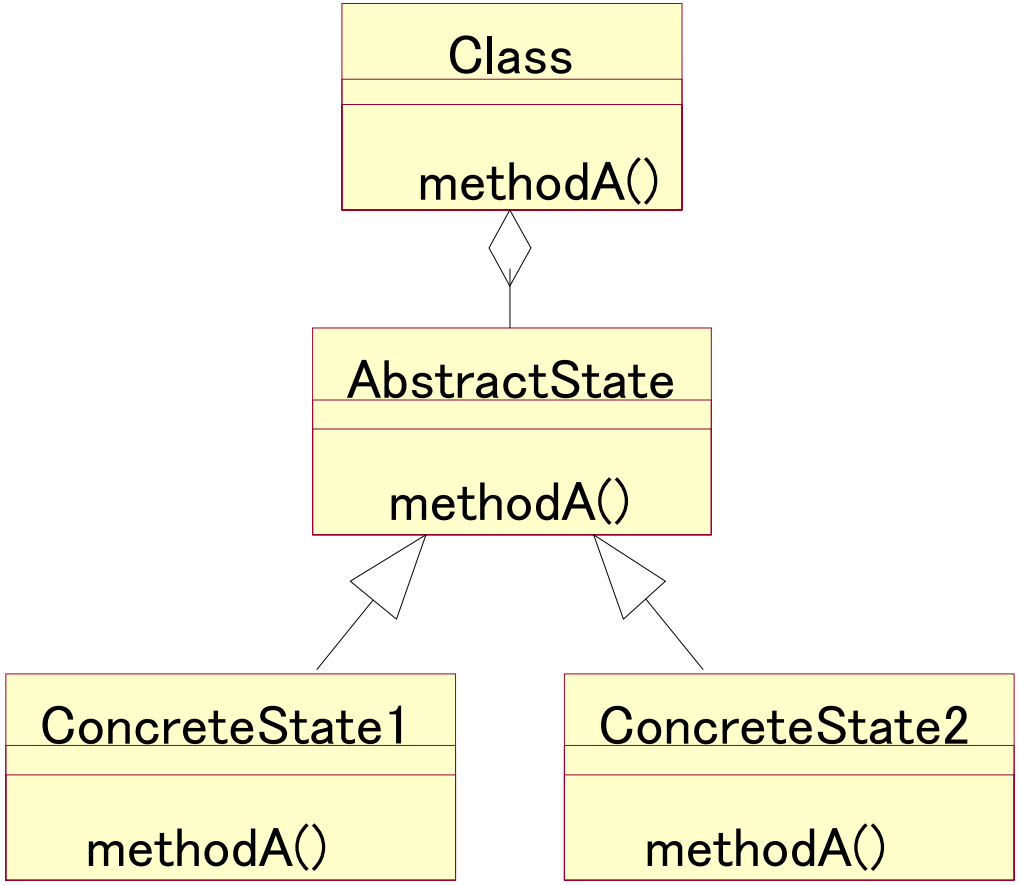


ポイント：状態によって同じ名前の処理でも内容が異なるとき

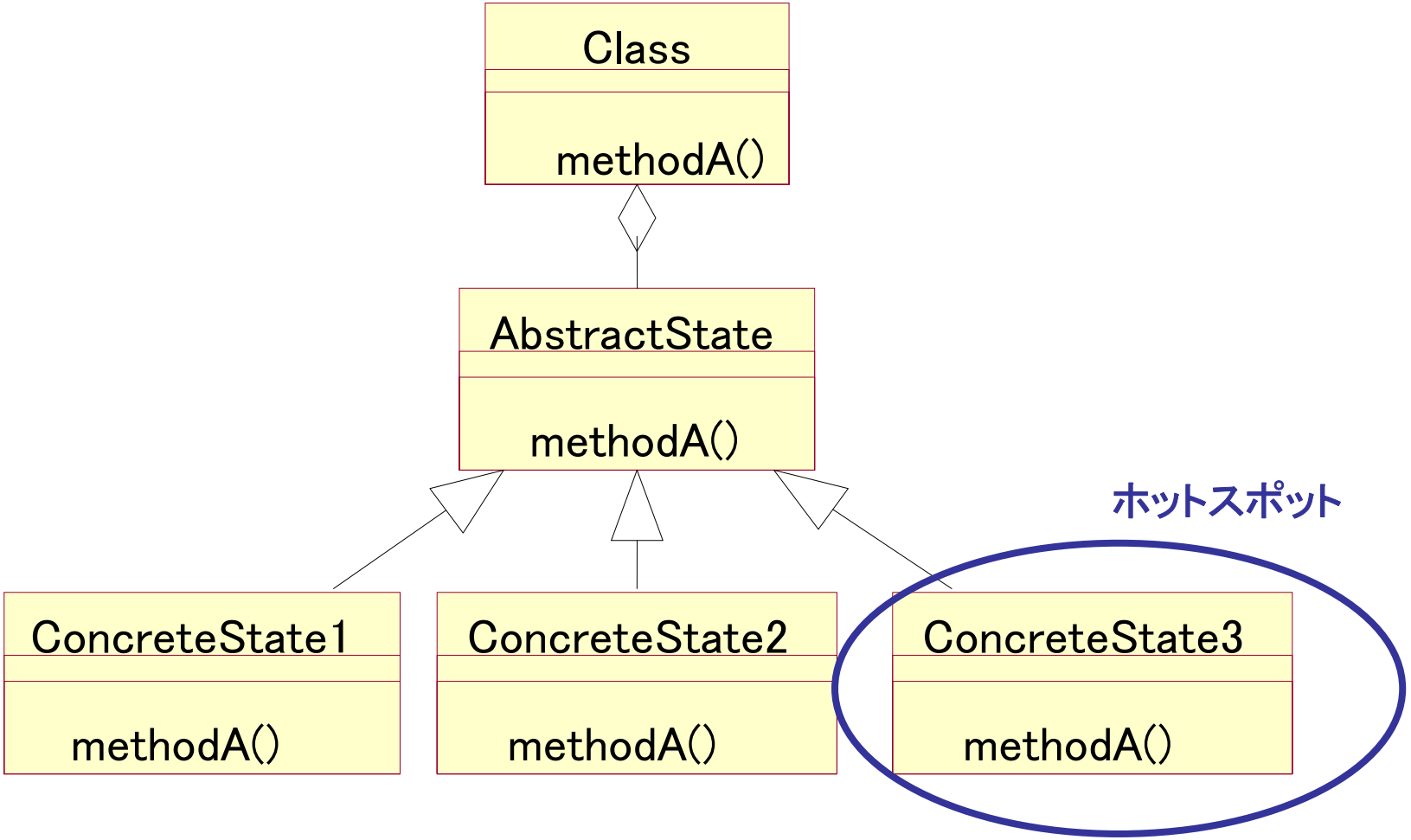
State Pattern (3)

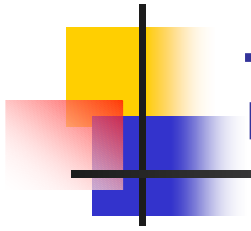


State Pattern (3)



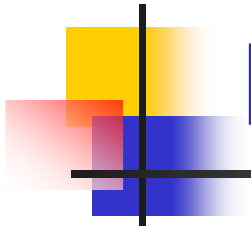
State Pattern (3)





設計の再利用

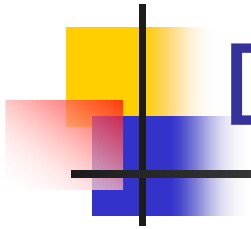
- ライブラリ
- オブジェクト指向：
  - 過去作られたクラスモジュールの再利用
  - 情報隠蔽
  - カスタマイズ：サブクラス化
  - 再利用の単位：クラスだけではなく、そのクラスがどう使われるかが重要
- パターン（モジュールの組み合わせ）の再利用へ



Design Pattern (1)

Design Pattern：設計の構造を表わしたもの、よく使われる構造をパターン化しカタログ化したもの

- ○○の場合は、××の構造にする
- 分類が重要



Design Pattern (2)

Design Patterns : Elements of Reusable OO Software by Gamma,E. et al (1995)

- 対象
  - クラス、オブジェクト、複合（再帰的な構造を持つオブジェクト）
- 特徴
  - 生成(Creational)：オブジェクトの生成
  - 構造(Structural)：クラスやオブジェクトの作成（継承関係やaggregation）
  - 振舞い(Behavioral)：クラスやオブジェクトのインタラクション、役割の分担

Design Pattern (3)

	生成	構造	振舞い
クラス	Factory Method	Adapter Bridge	Template Method
オブジェクト	Abstract Factory Prototype Solitaire	Adapter Bridge Flyweight Glue Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy
複合	Builder	Composite Wrapper	Interpreter Iterator Visitor