

ソフトウェア設計法及び演習 ソフトウェア工学概論及び演習

関澤 俊弦
日本大学 工学部

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - ・ データ属性, メソッド
 - ・ クラス, インスタンス
 - ・ カプセル化と情報隠蔽
 - オブジェクト間の関連

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - ・ データ属性, メソッド
 - ・ クラス, インスタンス
 - ・ カプセル化と情報隠蔽
 - オブジェクト間の関連

レビュー

■ レビュー

- 仕様書や設計書, プログラムなどを, **開発者とは別の人**が内容を検討し, 結果をフィードバックする工程
- レビュー項目
 - 仕様や要求を満たしているか
 - 誤りや不具合の有無
 - 冗長性の有無



思い込みによる検討漏れを防ぐなど, 「開発者とは別の人」がレビューを実施することが重要

レビュー

■ 注意点

- 客観的に評価する(私情を挟まない)
- 肯定的な評価を心がける
- 指摘事項・コメントは具体的に書く

レビュー後の作業

■ レビューの清書

- 紙媒体のレビューをWordで清書する

■ 設計演習1の修正

- レビューで指摘された項目を検討する
- 検討結果に従って、設計演習1を修正する

■ 修正方法

- 修正前後のファイルを分ける
- 加筆・修正履歴を付ける

修正した設計演習1の提出

■ 提出

- 期限: 2016年6月5日(日) 23:59
- 方法: ポータルサイトの課題管理

■ 提出物:

- 下記ファイルをZIPで1ファイルとする

1. レビュー(3枚)(.pdf)
 - ポータルサイトのレビュー用紙(Word)を用いて清書したのち, 3枚のレビューを1つのPDFとすること
2. 設計1を修正したレポート(.pdf)
 - 修正点を明らかにすること
 - ファイルフォーマットはPDFとする

(再掲)成績評価

■ 成績評価

設計演習2回と期末試験で評価する

- 設計演習(2回) ... 60%
- 期末試験 ... 40%

■ 判定

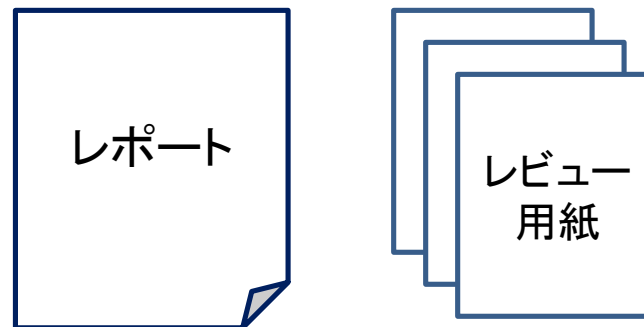
- 総計60点以上で単位認定

設計演習1とレビューの配点

■ レポート・レビューの配点に関して

$$\text{レポートの点} + \alpha_1 \pm \alpha_2$$

- レポートの点: レポートそのものの点数
- α_1 : レポートの加点
- α_2 : レビューの加点/減点
 - レビューの点はレビューアに付く



レビュー方法

1. レポート提出者(設計者)は, レビュー用紙を3枚受け取る
2. レビュー1 (20分)
 1. 設計者は, **共同作業者以外の人**にレビューを依頼し, レポートとレビュー用紙1枚を渡す
 2. レビューアはレビューを行ない, 評価を記載したレビュー用紙とレポートを設計者に返す
3. レビュー2, レビュー3を行なう
 - 同じレビューアは不可
4. 設計者は, レビュー3枚の指摘事項を検討し, 必要に応じて設計を修正する

レビュー: 1回目

■ 時間: 20分間

hh:mm – hh:mm

■ 手順

1. 設計者は、**共同作業者以外の人**にレビューを依頼し、レポートとレビュー用紙1枚を渡す
2. レビューアはレビューを行ない、評価を記載したレビュー用紙とレポートを設計者に返す

レビュー: 2回目

■ 時間: 20分間

hh:mm – hh:mm

■ 手順

1. 設計者は、**共同作業者以外の人**にレビューを依頼し、レポートとレビュー用紙1枚を渡す
2. レビューアはレビューを行ない、評価を記載したレビュー用紙とレポートを設計者に返す

レビュー: 3回目

■ 時間: 20分間

hh:mm – hh:mm

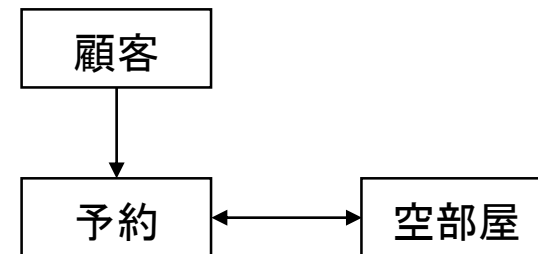
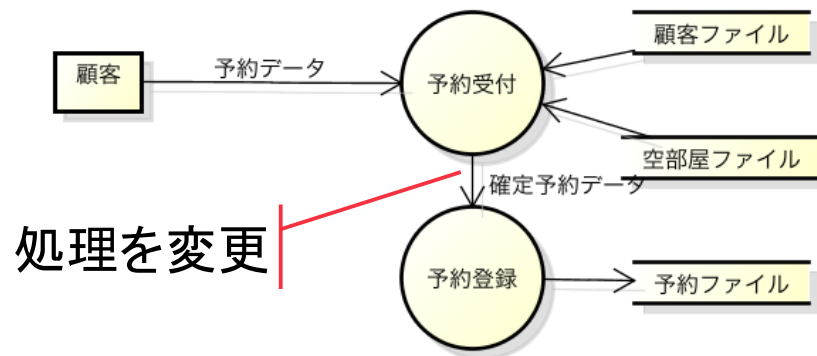
■ 手順

1. 設計者は、**共同作業者以外の人**にレビューを依頼し、レポートとレビュー用紙1枚を渡す
2. レビューアはレビューを行ない、評価を記載したレビュー用紙とレポートを設計者に返す

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - ・ データ属性, メソッド
 - ・ クラス, インスタンス
 - ・ カプセル化と情報隠蔽
 - オブジェクト間の関連

従来の設計方法の問題点

- 処理やデータ(構造)に変化があった場合, 設計に影響を及ぼす
- 問題点
 - システム側からの視点での設計
 - ・ データ構造や処理



処理の変更に応じて,
数々の修正が必要

オブジェクト指向開発

■ 概要

- システムの記述を「オブジェクト」と呼ばれる構成要素に分割してシステムを記述する開発手法
- “データ”と“操作(手続き)”を一体とする
 - データ(属性)とその処理をモジュールとしてまとめる(コンポーネント)
 - 機能中心ではない



オブジェクト指向 (OO: Object Oriented)では,「オブジェクト」はあくまでも「もの」の捉え方の概念であり,プログラムとは必ずしも対応しません.

オブジェクト指向開発

■ 特徴

- システムの静的・動的側面の記述
- 人間の視点からの分析・設計
(現実に即したシステム)
- シームレス開発
- 拡張性, 再利用性

オブジェクト

■ オブジェクト(object)とは

 p.79

□ 対象を(抽象的に)表わす実体

- ・ エンティティに近い

□ 次のメンバ(≡要素)をもつ

- ・ 属性(データ属性)
- ・ 操作(メソッド, 振舞いと呼ぶこともある)

オブジェクト
データ属性1 データ属性2 データ属性3
メソッド1 メソッド2

エンティティ
データ属性1 データ属性2 データ属性3



オブジェクトの詳細は後述

オブジェクト指向の効果

■ データと操作の一体化

- データの操作方法を一体化して捉えられる
- オブジェクトの独立性を高める
(部品(コンポーネント)化を促進する)
 - 再利用性, 拡張性
 - コンポーネントの追加で機能拡張を行なう

システムの静的・動的側面

■ 静的な側面

- データの流れやデータ構造など
- 業務モデル(DFDなど), ER図などで表現

■ 動的な側面

- 動作タイミングやイベント割込みなど
- シーケンス図やステートチャートで表現



シーケンス図, ステートチャートは後述(6.3節)

人間の視点からの分析・設計

■ 機能中心の設計

- 仕組みを「機能」として抽象化
- ユーザの視点とは異なる
 - ・ システムの最終形が分かりづらい

■ オブジェクト指向開発

- 実世界の形態をオブジェクトとして捉える
 - ・ オブジェクト間の関連性も業務の仕組み(ビジネスルール)により決定する

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - データ属性, メソッド
 - クラス, インスタンス
 - カプセル化と情報隠蔽
 - オブジェクト間の関連

オブジェクト

■ オブジェクト(object)とは



p.83

□ 対象を(抽象的に)表わす実体(エンティティ)

• 実体は抽象的なもの(概念)でもよい

– 銀行口座システムの「ATM」,「口座」など

■ オブジェクトは次のメンバ(≡要素)をもつ

□ 属性


□ 操作(メソッド, 振舞いと呼ぶこともある)



「オブジェクト」はあくまでも「もの」の捉え方の概念であり、プログラムとは必ずしも対応しません。

データ属性, メソッド

■ データ属性

 p.83, 84

□ オブジェクトの性質や設定を表わす情報

■ メソッド

□ オブジェクトのデータ(属性)に対する操作

オブジェクト
データ属性1 データ属性2 データ属性3
メソッド1 メソッド2

ATM
支店番号 ATM番号 :
カード要求 暗証番号要求 払戻要求 :



ATMの内部構造を知る必要なく, ATMを操作できる

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - データ属性, メソッド
 - クラス, インスタンス
 - カプセル化と情報隠蔽
 - オブジェクト間の関連

クラス, インスタンス

■ クラス

□ オブジェクトを定義したもの

- 属性と操作を規定する
- プログラミングではデータ型に相当する

■ インスタンス

□ オブジェクトのデータ属性に具体的な値を持たせて, 具体化したもの(実体)

- 通常, 1つのオブジェクトは複数のインスタンスを持つ



「オブジェクト」は, インスタンスを意味することもあります.



注意: このページの説明は教科書と若干異なります

例: クラス, インスタンス

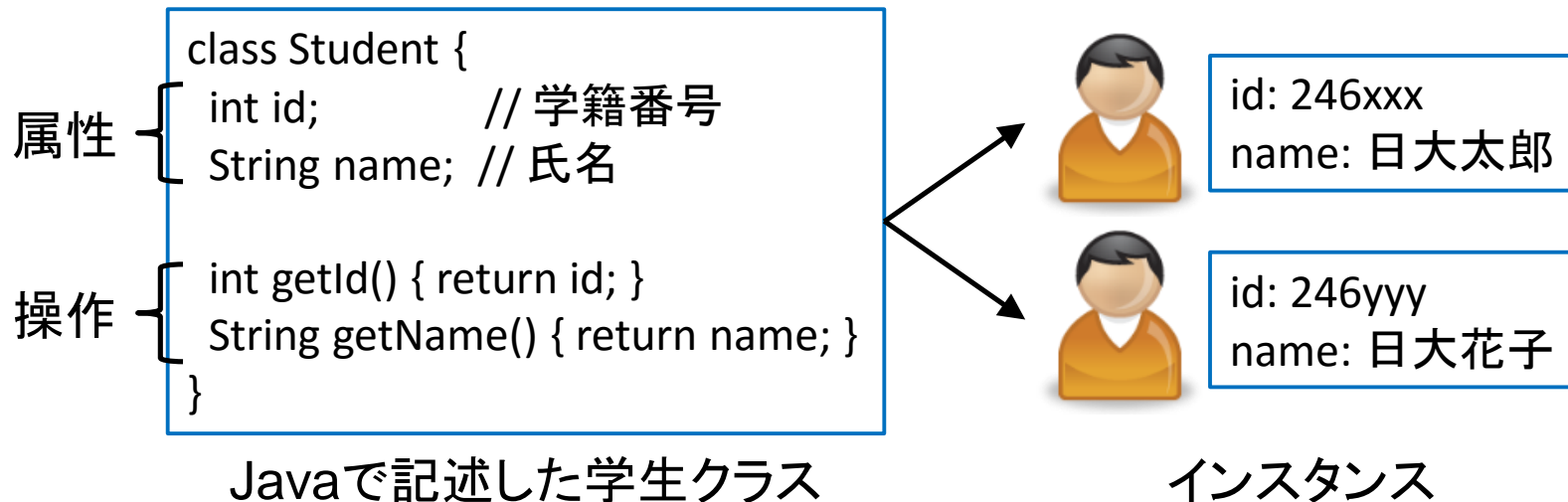
■ 学生クラス (学生オブジェクト)

- 「学生」という一般的な「もの」を定義する

学生
id
name :

■ インスタンス

- 「学生クラス」から, 「日大太郎」さんのように具体的な学生として実体化したもの



- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - データ属性, メソッド
 - クラス, インスタンス
 - カプセル化と情報隠蔽
 - オブジェクト間の関連

カプセル化と情報隠蔽

■ カプセル化

- オブジェクトの中に、属性とそれらの操作をまとめること
 - ・ メンバ: オブジェクトの属性と操作

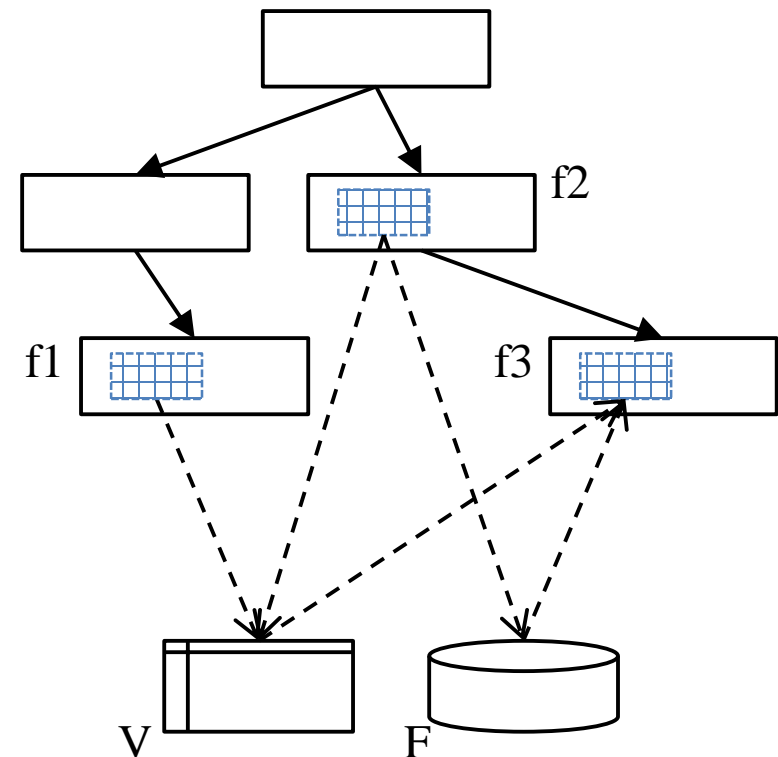
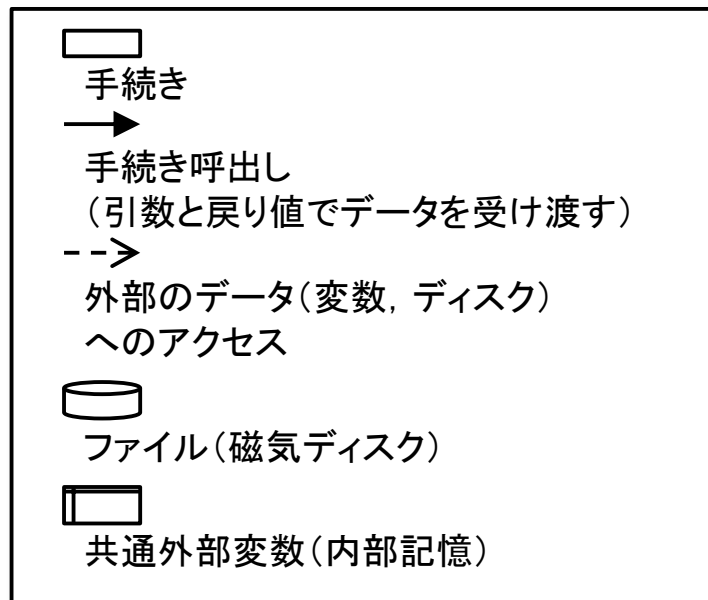
■ 情報隠蔽

- カプセル化とメンバの非公開設定により、(オブジェクトの)外部から属性へのアクセスを制限すること

例: カプセル化と情報隠蔽(1)

■ 手続きによるモジュール化を考える

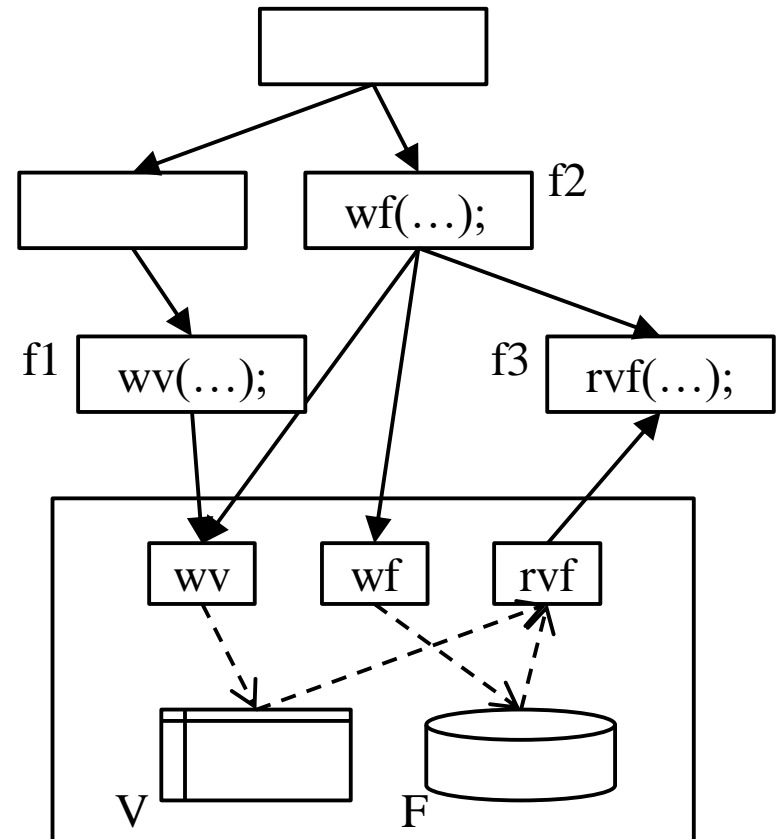
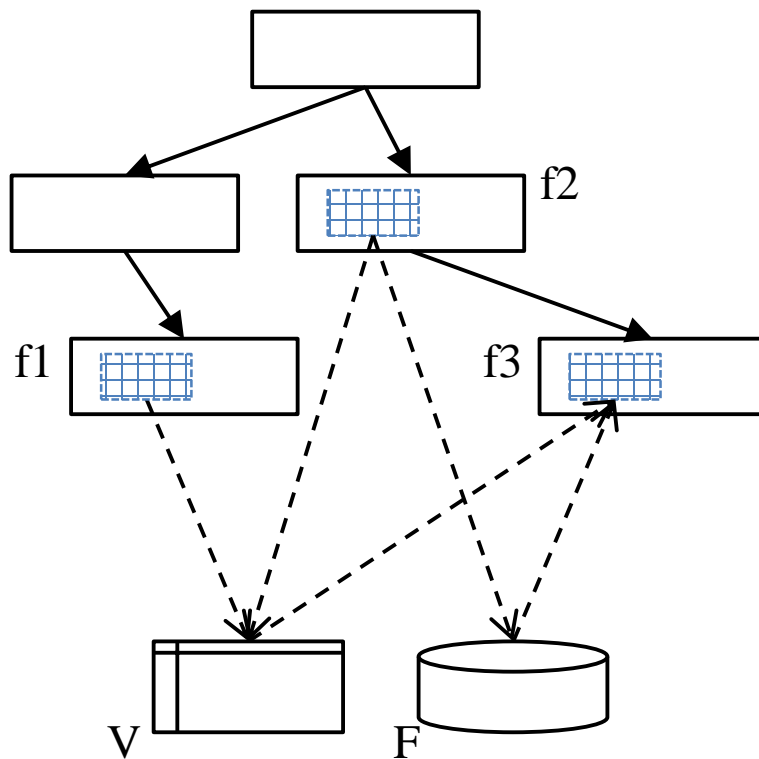
- f1: V に書き込み
- f2: V, F に書き込み
- f3: V, F から読み込み



例: カプセル化と情報隠蔽(2)

■ カプセル化による情報隠蔽

□ wv , wf , rvf により, V , F を隠蔽する

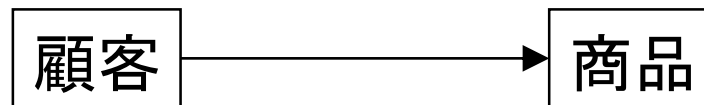


- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - データ属性, メソッド
 - クラス, インスタンス
 - カプセル化と情報隠蔽
 - オブジェクト間の関連

オブジェクト間の関連

■ オブジェクト間の関連性

- 1対1, 1対多, 多対多の関連性が存在する
 - ER図と同じ



上記の関連性の表記は特定の記述法に従ったものではありません. UMLにおける表記法は後述

クラスの階層化と継承

クラスは階層構造をとる

■ 継承 (inheritance)

- 上位クラスの属性や操作を引き継ぐこと
 - ・ 属性や操作を追加し, 新たなクラスを作る
 - ・ 上位クラスを「スーパークラス」, 新たなクラスを「サブクラス」と呼ぶ

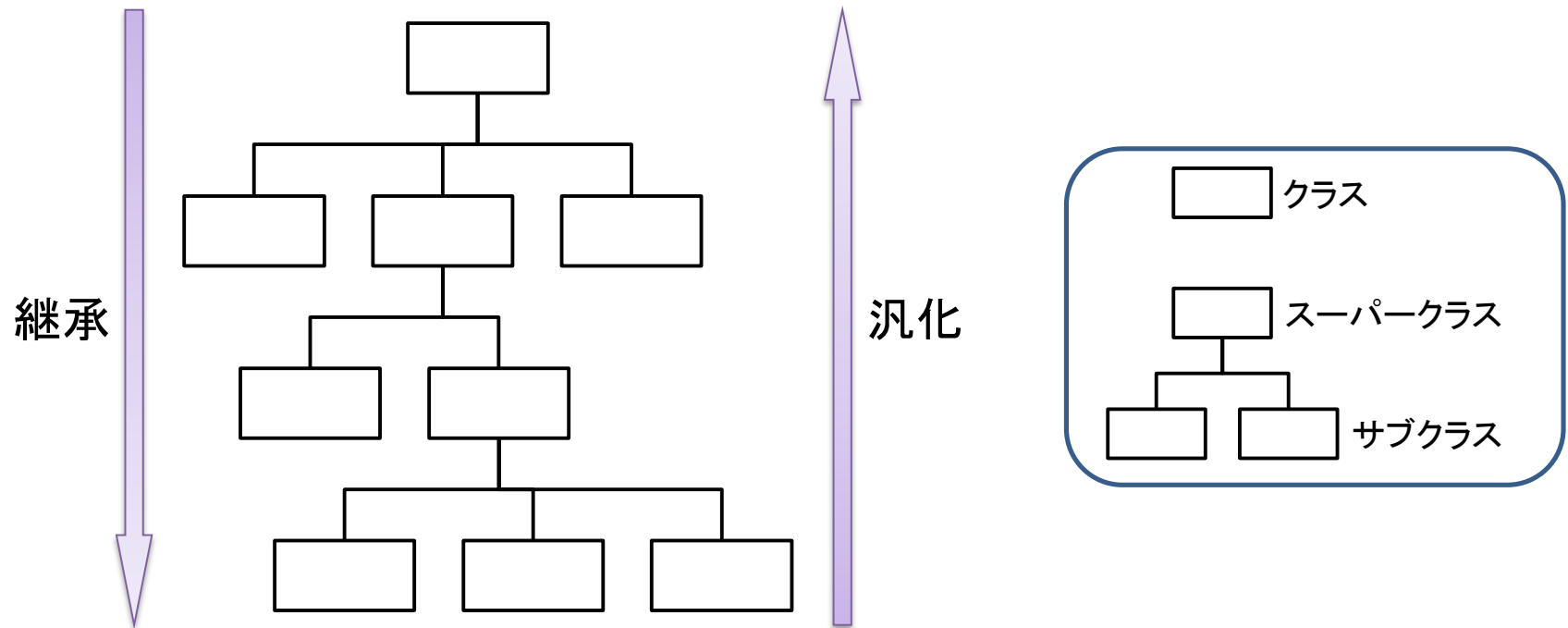
■ 汎化 (generalization)

- 共通する属性や操作を抽出し一般化すること
 - ・ 継承の逆

継承の階層

■ 継承は階層構造をなす

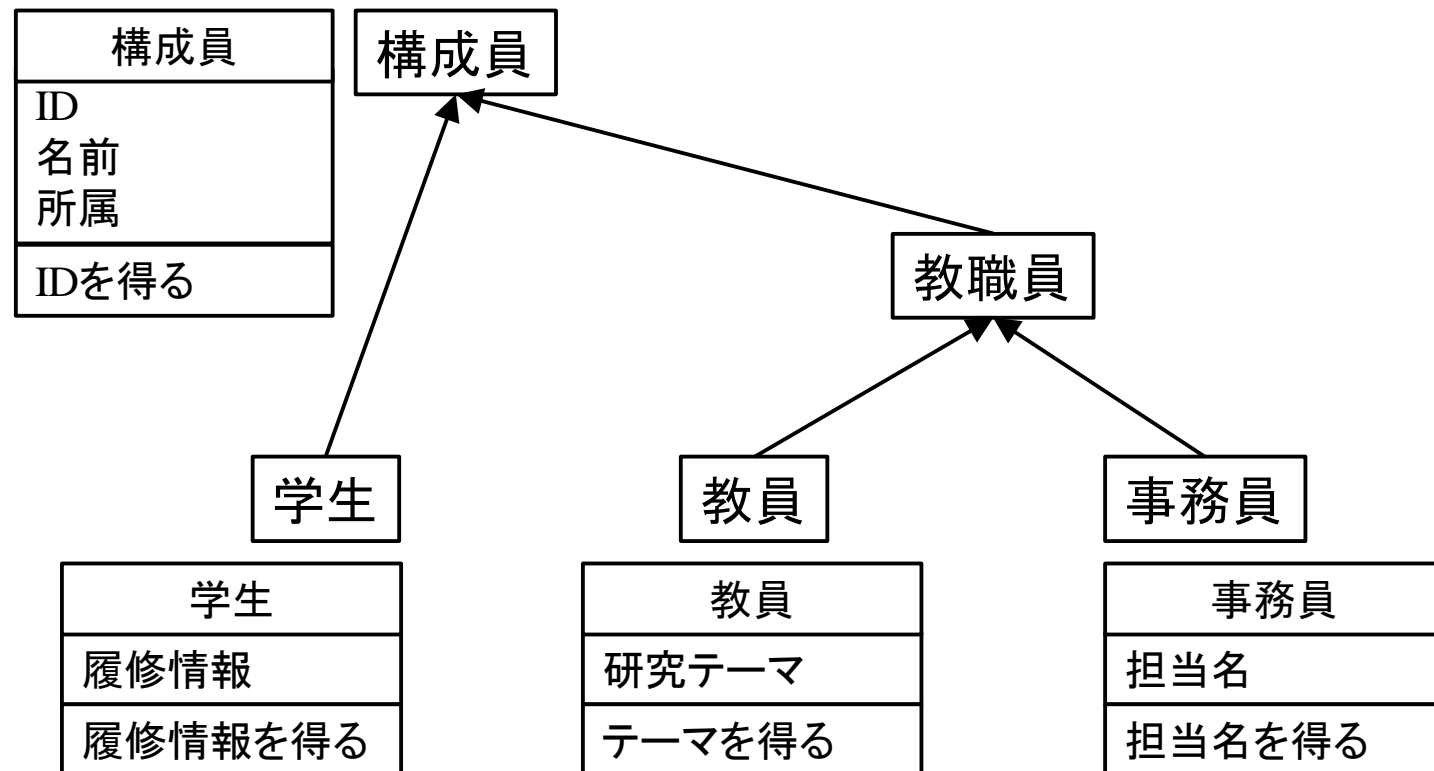
- 上位のクラスほど抽象度が高い
(下位のクラスほど具体性が高い)



例: 大学の構成員

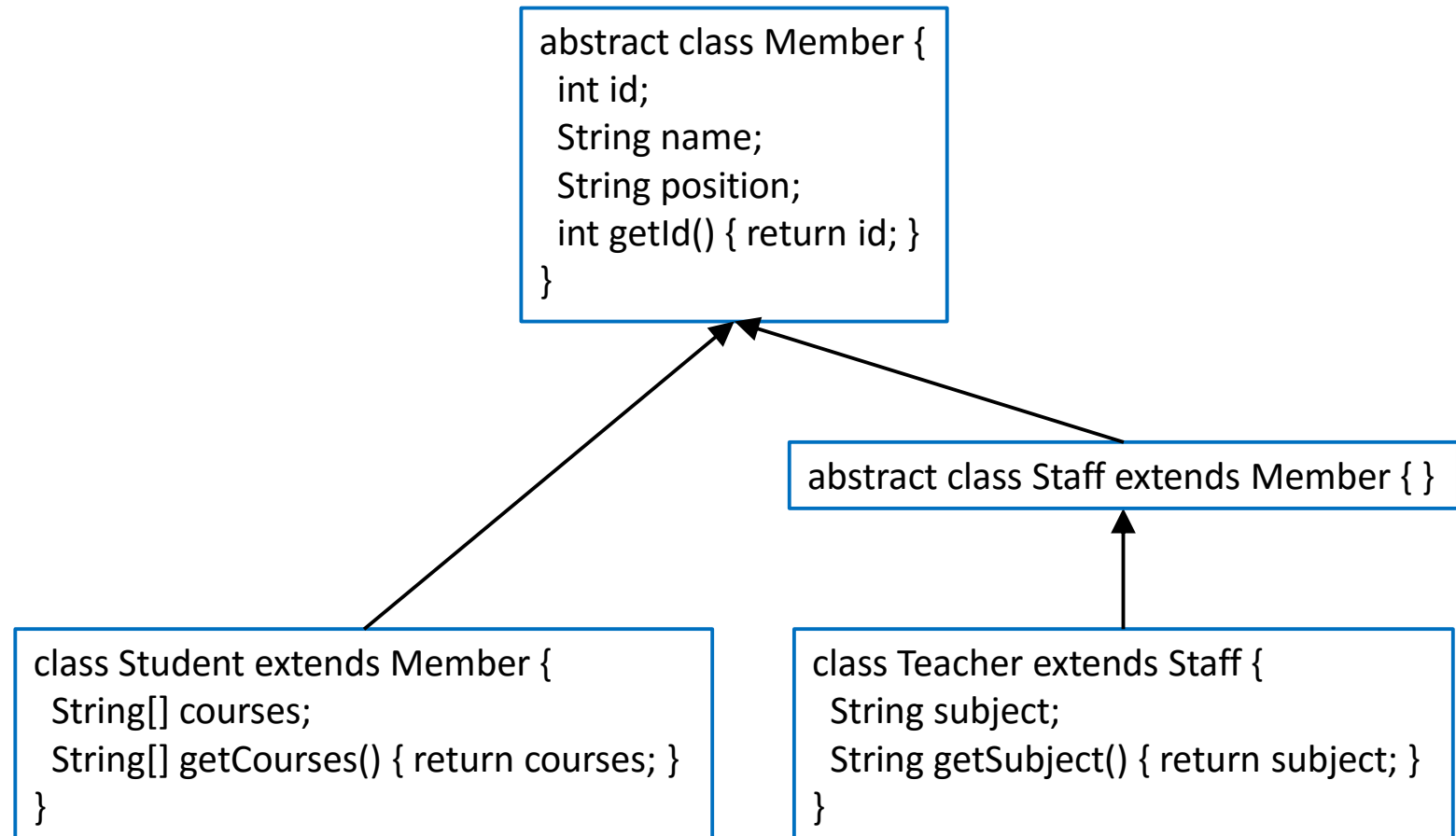
■ 「大学の構成員」のクラスを考える

□ 構成員: 学生, 教員, 事務員, ...



例: 大学の構成員

■ (参考)Javaでの記述例



クラスの階層化

■ クラスの階層化のタイプ

□ INSTANCE-OF

- すべてのインスタンスは共通のデータ属性を持つ

□ IS-A

- 複数のクラス間で共通する特性を抜き出し、一般化したクラスを定義する(汎化).
 - 汎化によりスーパー/サブクラスの関係が生じる

□ PART-OF

- 複数のクラスの集約により、別のクラスを構成するクラスを定義する(集約化)

まとめ

- 設計演習1のレビュー
- オブジェクト指向
 - オブジェクト指向開発
 - オブジェクト
 - ・ データ属性, メソッド
 - ・ クラス, インスタンス
 - ・ カプセル化と情報隠蔽
 - オブジェクト間の関連