オブジェクト指向開発論

2020年5月14日

海谷 治彦

目次

- 講義の目標
- 単位認定について
- 参考書
- 授業の進め方
- なぜプログラミングは難しいか
- Java, UML, Eclipse
- 受講上の注意

講義の目標

- ソフトウェアを分析・設計してから，プログラムを開発するような人に受講生がなること.
  - いきなり，エディタ(もしくはIDE)でコードを書くのは今後はNG.
- 特にオブジェクト指向設計ができること.

- Javaでオブジェクト指向プログラムが書けること.
- UMLでオブジェクト指向設計ができること.

単位認定について

- 演習(プログラミングやモデリング)の結果で判断します.
  - 大体, 4回前後.
- 中間試験や期末試験は,
  - 行わない予定です.

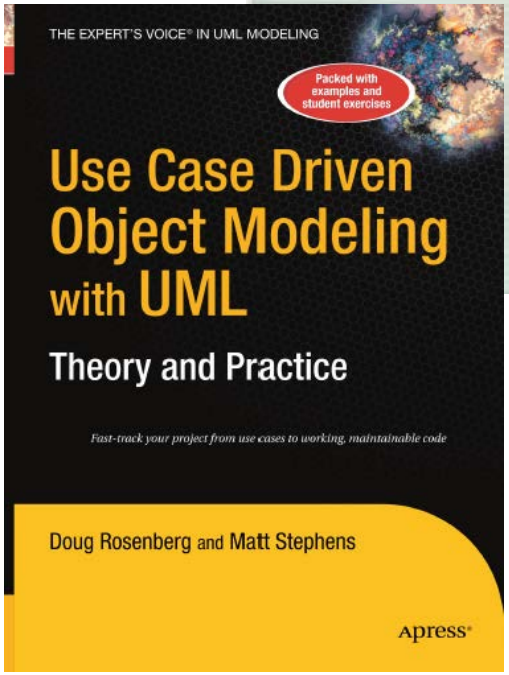
- Javaは適宜補足してゆきます.
  - 並行してJavaの授業とってください.

参考書

- 古い本ですが，とてもしっかりした考え方を平易に伝えています．
- Javaの文法は古いので，授業中に適宜補います．
- 上級者と思っている人にも学ぶべき所のある本です．
- 題名が残念．

参考書 その2

- ICONIXという手法の解説本です.
- まぁ, この分野ではポピュラーではないかと思います.
- この講義は, ほぼ本書の話.



授業の進め方

- 基本的にシラバスにそって講義やります.
- 演習をやってもらいます.
  - プログラム，設計，たまには感想や考察.
  - 提出先は teams の予定.
- 特に発表会とかは考えてませんが，やりたい人がいれば名乗り出てください.
- オフィスアワー 水曜 PM8-10 試しに夜に
  - 出張等でいない場合もあり
- スケジュール: 以下から確認してください.

http://www.sci.kanagawa-u.ac.jp/info/kaiya/oo/

ソフトウェア工学

- 授業の題目は「オブジェクト指向開発論」ですが，一般(他大学等)には「ソフトウェア工学」と呼ばれる内容となります．
- ソフトウェア工学は，ソフトウェアを開発する手順を系統化，自動化することを目的としてます．
  - ソフトウェアの軍事利用が発端だったようです．
- 系統化するための理論としてオブジェクト指向が一番メジャーなので，そのような授業名となりました．

ソフトウェア工学

ってか，工学って何？

言葉の定義 by 広辞苑

- 工学:

  基礎科学を工業生産に応用して生産力を向上させるための応用的科学技術の総称。
- 工業:

  原料や粗製品を加工して有用なものとする産業。
- 生産:

  自然物に人力を加えて、人にとって有用な財を作り出し、もしくは獲得すること。
- 産業:

  生産を営む仕事、すなわち自然物に人力を加えて、その使用価値を創造し、また、これを増大するため、その形態を変更し、もしくはこれを移転する経済的行為。

言葉の定義 by 広辞苑

- 工学:
  基礎科学を工業生産[illegible]用[illegible]向上させるための応用的[illegible]
- 工業:
  原料や[illegible]。
- 生産
  自然[illegible]り出し、もしく[illegible]
- 産業
  生産を[illegible]を加えて、その使用価値を[illegible]増大するため、その形態を変更し、もしくはこれを[illegible]る経済的行為。

工業・工学の根幹は主観的

↓

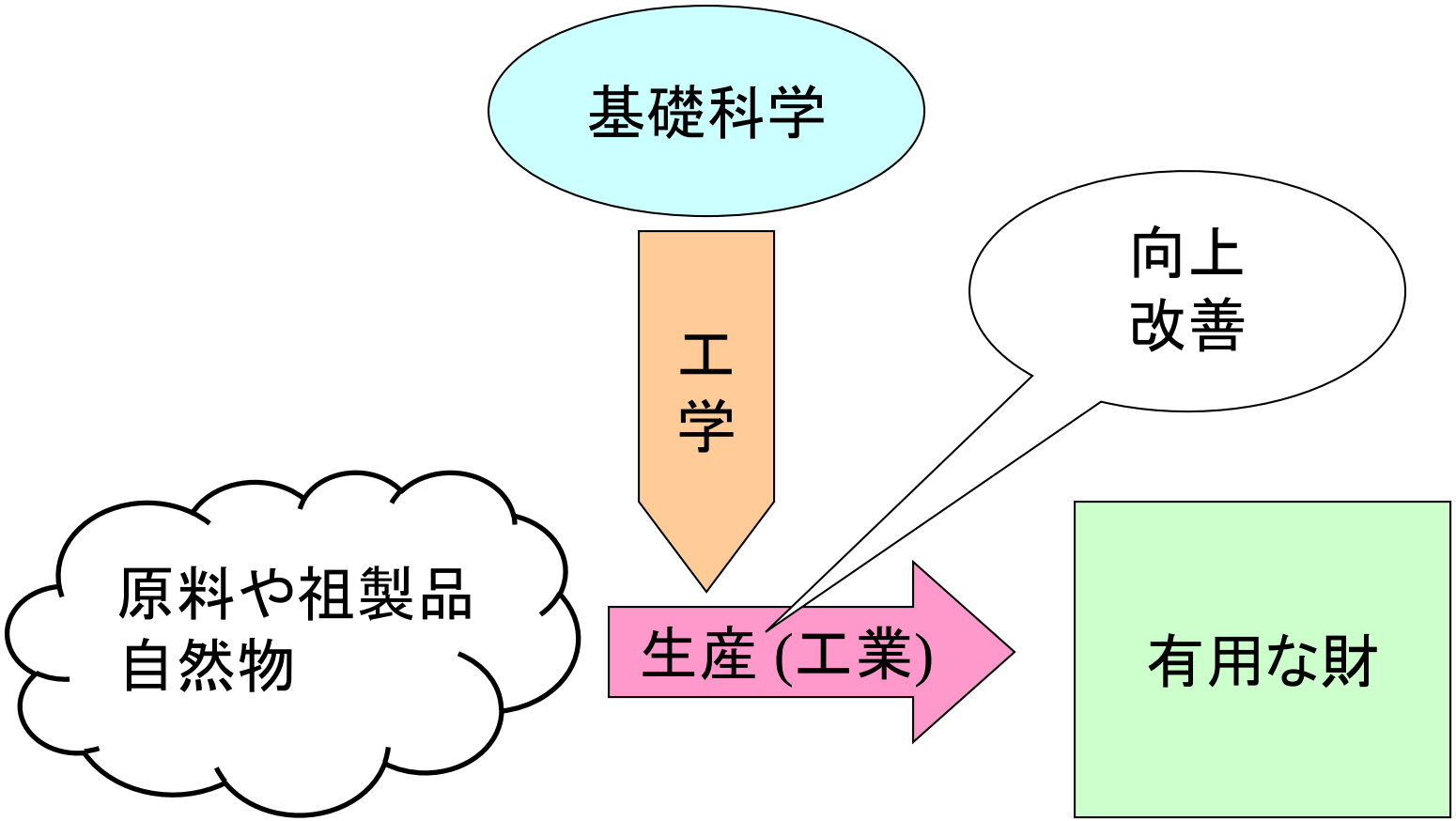
ヒトから見た
有用性，価値に
依存している．

科学

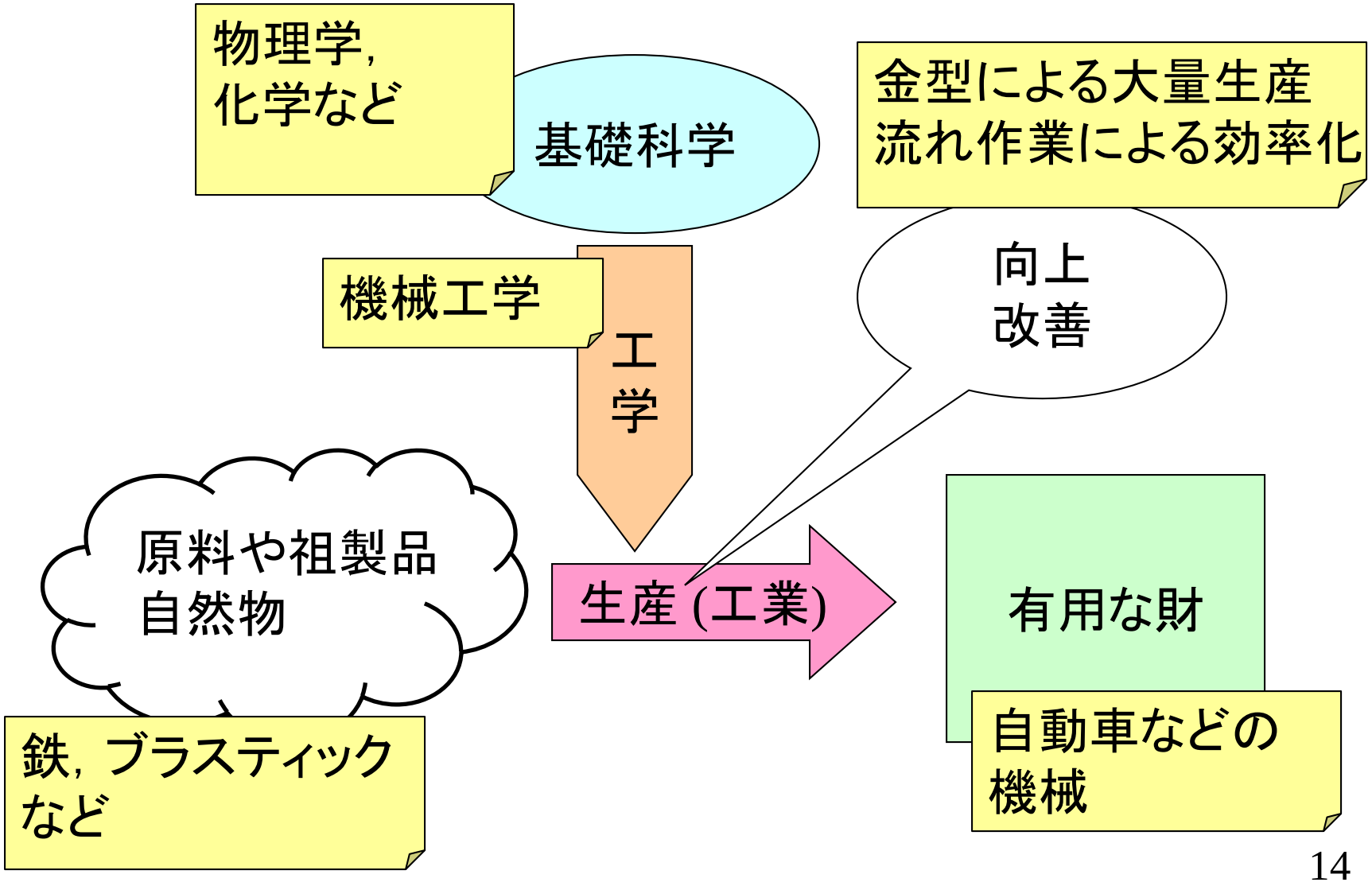
- 観察や実験など経験的手続きにより実証されたデータを論理的・数理的処理によって一般化した法則的・体系的知識。
- また、個別の専門分野に分かれた学問の総称。
- 物理学・化学・生物学などの自然科学が科学の典型であるとされるが、同様の方法によって研究される社会学・経済学・法学などの社会科学、心理学・言語学などの人間科学もある。
- 狭義では自然科学と同義。

[広辞苑 第七版]

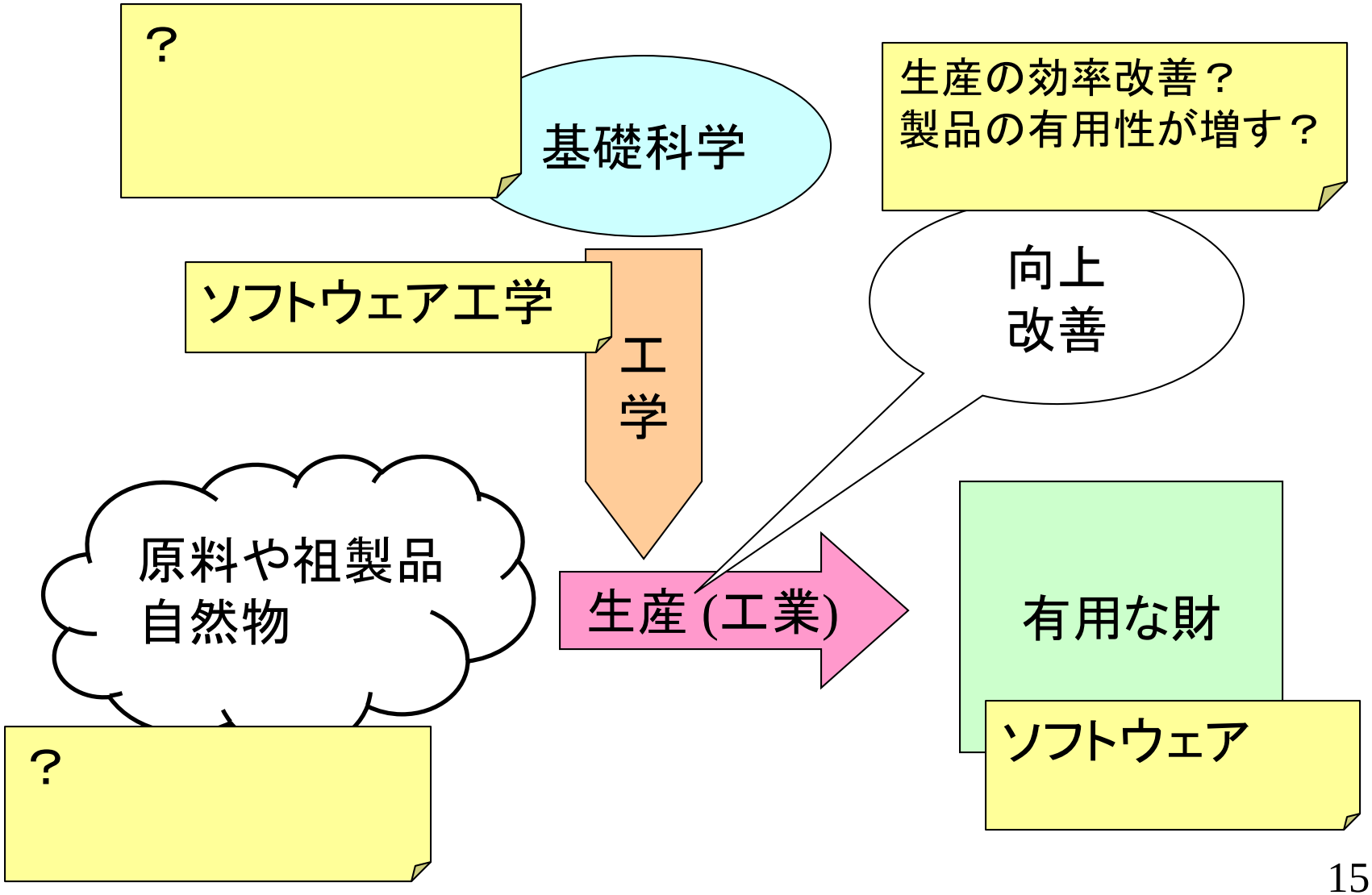
工学



工学と機械工学


物理学，
化学など
基礎科学
金型による大量生産
流れ作業による効率化
機械工学
工学
向上
改善
原料や祖製品
自然物
生産 (工業)
有用な財
鉄，ブラスティック
など
自動車などの
機械

工学とソフトウェア開発



プログラミングは何故難しい？

その結論

- なぜプログラミングは難しいか？

以下のどこかでつまづいている！

1. コンピュータに行わせたいことを理解
2. 理解したことを説明できるレベルまで整理
3. コンピュータにわかる言葉に翻訳

プログラミングとは何か？

- コンピュータにやらせたいことの手順を，コンピュータのわかる言葉で書く.
- 参考書 15～16ページの掛け算の例

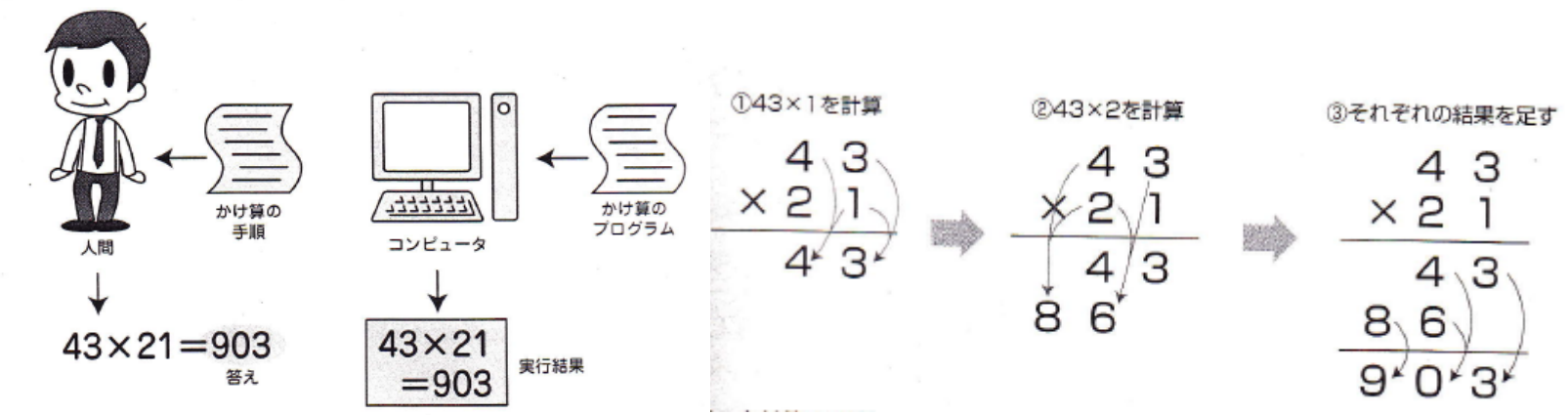
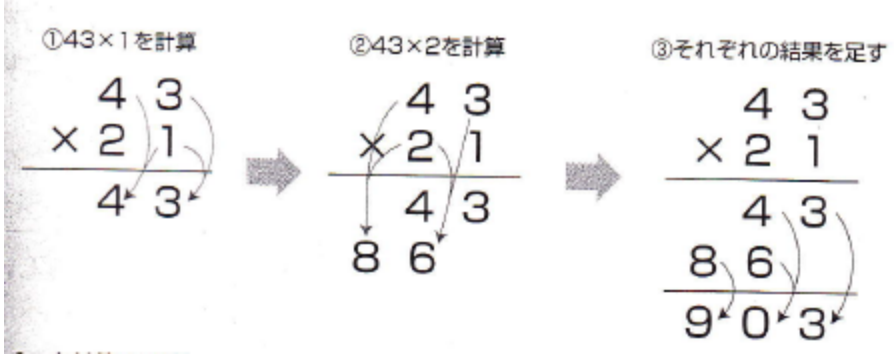


図1-2　かけ算とプログラムの対比

理解の失敗例

- もし，以下にあるような掛け算の手順を追い，意味が分からなければ，掛け算のプログラムはできない！



- もし，銀行業務が理解できなければ，その業務支援ソフトウェアは作れない.

何故，考案ではなく理解か？

- コンピュータにやらせたいことの多くは，現実世界の業務や手順の一部である.
  - 放射性物質の飛散予測の計算の一部（全部）
  - 銀行業務の一部
- そのような業務や手順は，その道の専門家が考案する.
  - 原子力専門家，物理学専門家，気象学専門家
  - 銀行員
- 我々，コンピュータ技術者は，これらを理解し，計算機で(どれだけ)肩代わり可能か判断する.
  - 判断のためには理解が必須！

整理の失敗例

- 個々の掛け算の計算手順はわかるが，それを一般化（整理）して，他人に説明できない．
  - 例えば，算数や数学にあるようにNやxみたいなパラメータを使って，計算手順を一般化できないとか．
- 銀行業務も個々の決済等の業務を一般化した手順として書けなければ，ソフトウェアを作れない．

整理のポイント

現実業務と計算機が可能なことのバランスが重要

- 現状の計算機(プログラム言語)で実現不可能な整理をしてもシステムはできない.
  - 微分方程式で飛散予測を整理できても，それをコンピュータで直接実行はできない.
- 業務と大きく剥離した形で整理しても，そもそも整理されているか確認しようが無くなる.
  - プログラムを直接見せられても分からない銀行員も多いだろう.
  - 結果として，計算機にやらせたいことが整理されているか，確認しようが無くなる.

説明相手は誰？

基本的に相手は二種類を想定する.

- 計算機
  - 計算機にやらせることを想定しているのだから，計算機に説明することを想定しないと.
  - 曖昧さや，直観は通じない.
- そもそもの業務専門家
  - コンピュータ技術者の理解が合っているかは，往々にして，専門家じゃないとわからない.
  - 専門家は人間だから，基本，長大なプログラムやアルゴリズム記述の意味はわからない.

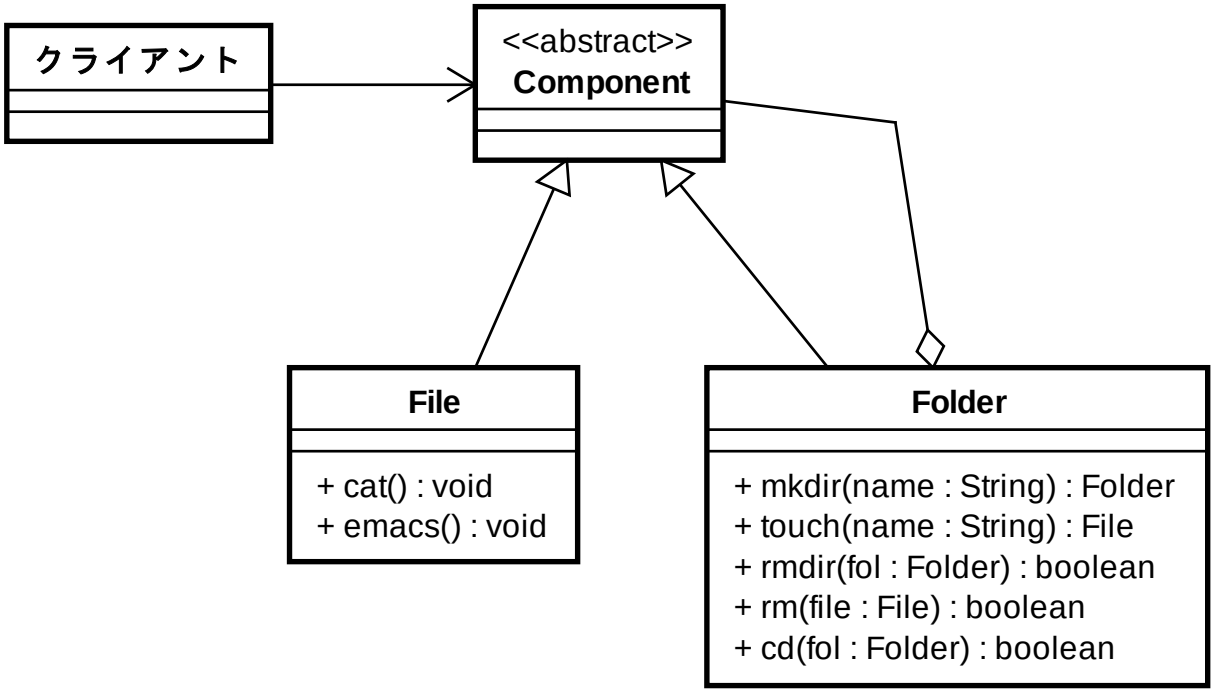
整理に創造は必要か？

- 現実的には，どう整理するかについて，新しい整理法や整理様式を新規に創造する必要は無い.
- 理解したことを，既存の様式を真似て，整理する技術をまずは覚え使えるようにすること.
- 例
  - PCのフォルダや組織階層等，階層的な構造(木構造)を整理するにはよく知られた様式が既に存在する.

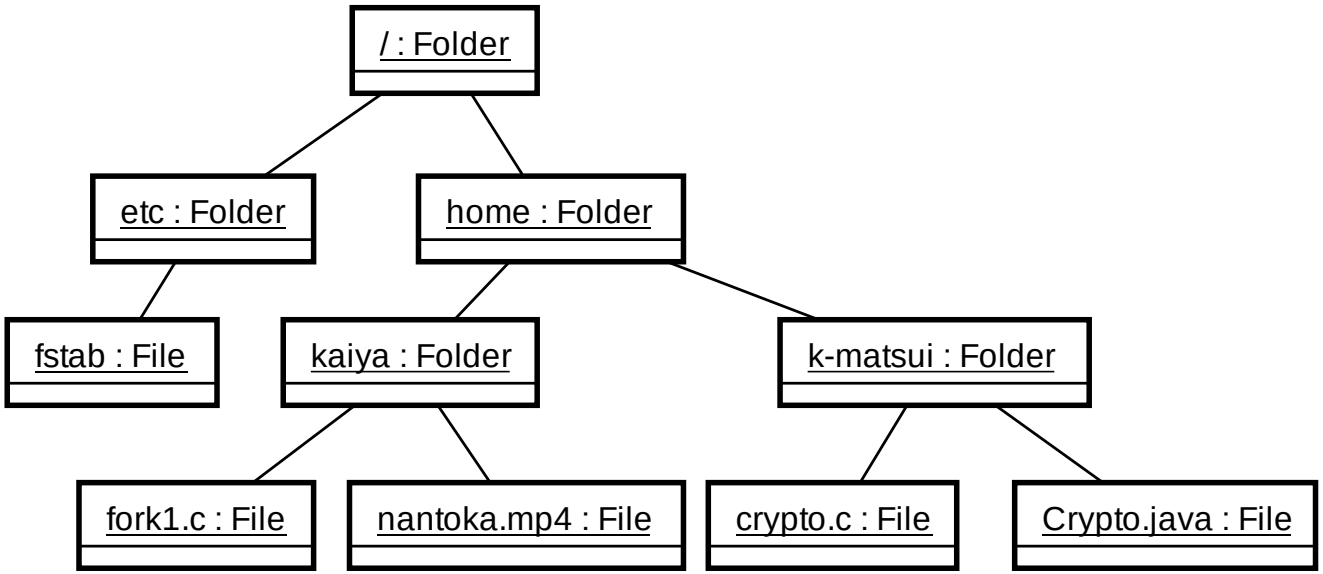
    (コンポジットパターン)
  - この様式を超える整理法を創造するのは多分無理.
- 既存の整理法を数多く知った上で，より良い整理法を創造することを最終的には目指してほしい.

コンポジットパターンの例

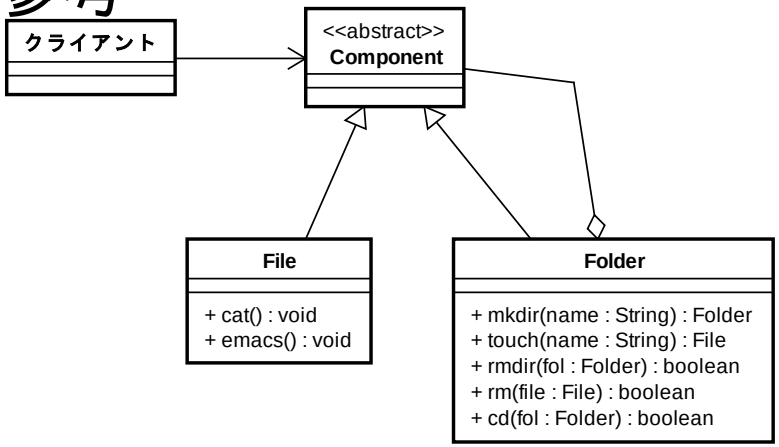
- UNIX風のコマンドをメソッドとして実装.
- 実際，UNIXでは，ファイルとフォルダの作成，削除のコマンドが異なるため，それも準拠.



インスタンスの例


/ : Folder
etc : Folder
home : Folder
fstab : File
kaiya : Folder
k-matsui : Folder
fork1.c : File
nantoka.mp4 : File
crypto.c : File
Crypto.java : File

参考


クライアント
<<abstract>>
Component
File
+ cat() : void
+ emacs() : void
Folder
+ mkdir(name : String) : Folder
+ touch(name : String) : File
+ rmdir(fol : Folder) : boolean
+ rm(file : File) : boolean
+ cd(fol : Folder) : boolean

翻訳の失敗例

- ずばり，プログラム言語を知らない．
  - この辺を補う図書は腐るほど出版されている．
  - 所謂，プログラム言語の授業は結構，この辺のみが重視されている．
- 言語を知っていても，一般化した手順の記述との対応が分からない．
  - 配列，リスト，スタック，木等は知っていても，それが「整理されたコンピュータにやらせたいこと」の何に対応するか分からない．
- 日本語を知っていても，コミュ力が無い人が居るのと同じ．

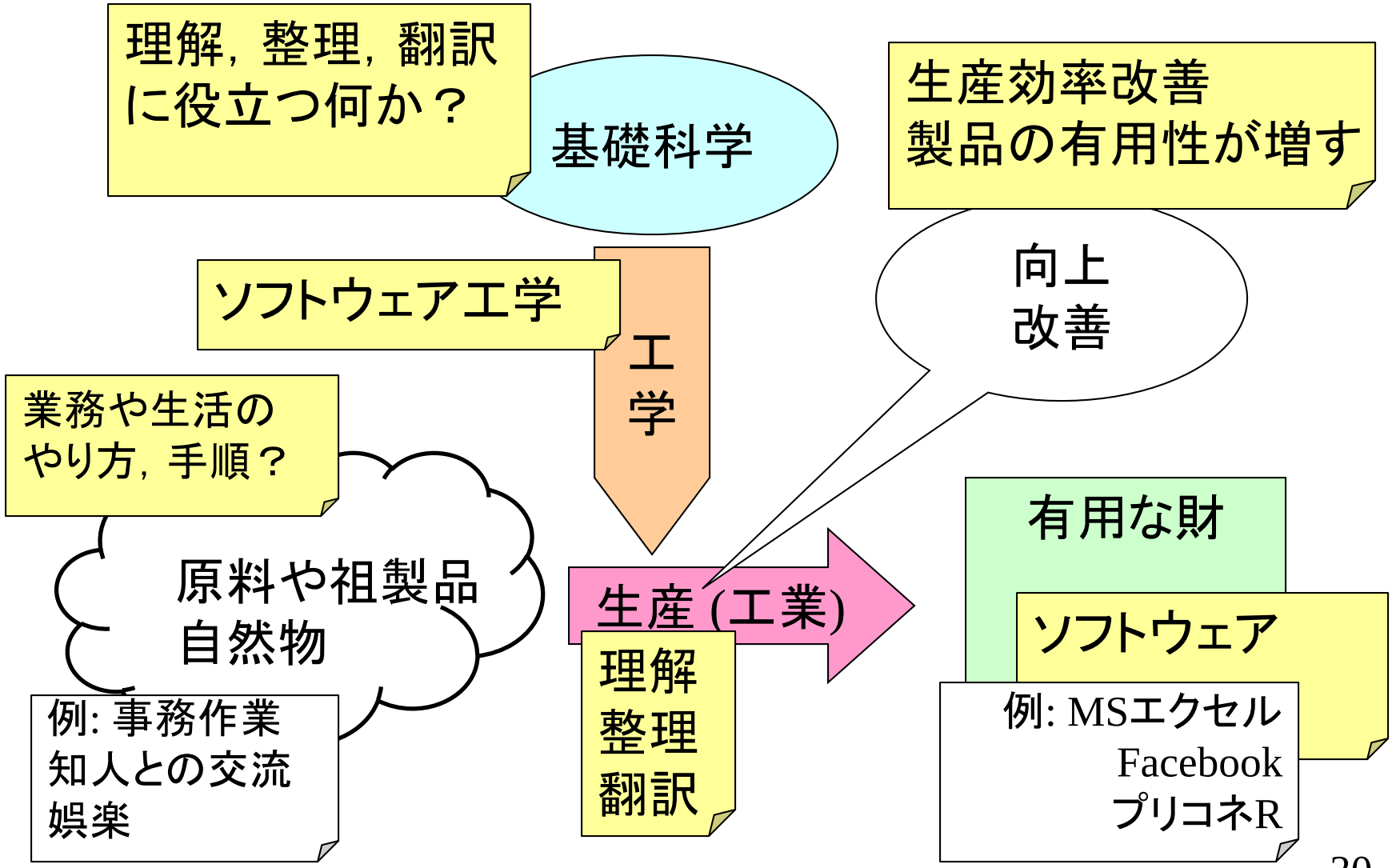
本当に難しいのは理解と整理

- 理解
  - 株取り引きの業務が理解できてますか？
  - ゲーム内での3D表示の人間の描画法を理解できてますか？
  - SPEEDI (スピーディ)の放射性物質の飛散予測法を理解できますか？
  - 迷惑メールとそれ以外との違いが理解できますか？
- 整理
  - 上記を個々の事例ではなく，一般化して整理できますか？（整理する既存法を知ってますか？）
  - 整理したことを人間と計算機の双方に都合よく説明できますか？

計算機屋は下請けか？

- 大筋で YES
  - 対象分野の一部もしくは全部を肩代わりする下請け業務.
- しかし，計算機技術により業務・生活の方が変化することも最近は多い.
  - 技術主導で世の中のあり方・やり方が変わる.
  - 例
    - 検索技術の発達による情報整理法の変化.
      - 図書館等 VS Webやデスクトップ検索
    - 携帯端末の発達による待ち合わせ法の変化.
      - いまどき，きっちり場所と時間を指定しなくても集まれる.
    - SNSの発達により，グループ，伴侶，家族等の在り方の変化.

工学とソフトウェア開発



基礎科学としての情報科学授業

- 理解
  - 心の情報処理
- 整理
  - 数理論理学関係 (命題論理や述語論理)
  - オートマトン，アルゴリズム
- 翻訳
  - プログラミング系
  - OS, ネットワーク
  - 画像処理，暗号，データベース

二章へのイントロ

オブジェクト指向の利点

- 理解対象である現実世界の事柄(株取引やゲームソフト)を理解や整理するのに，従来のやり方よりはマシである.
  - 従来のやり方: データ構造を作り，それを関数等で処理するC言語的な方法.
- 従来手法よりは，作ったものを改造しやすい.

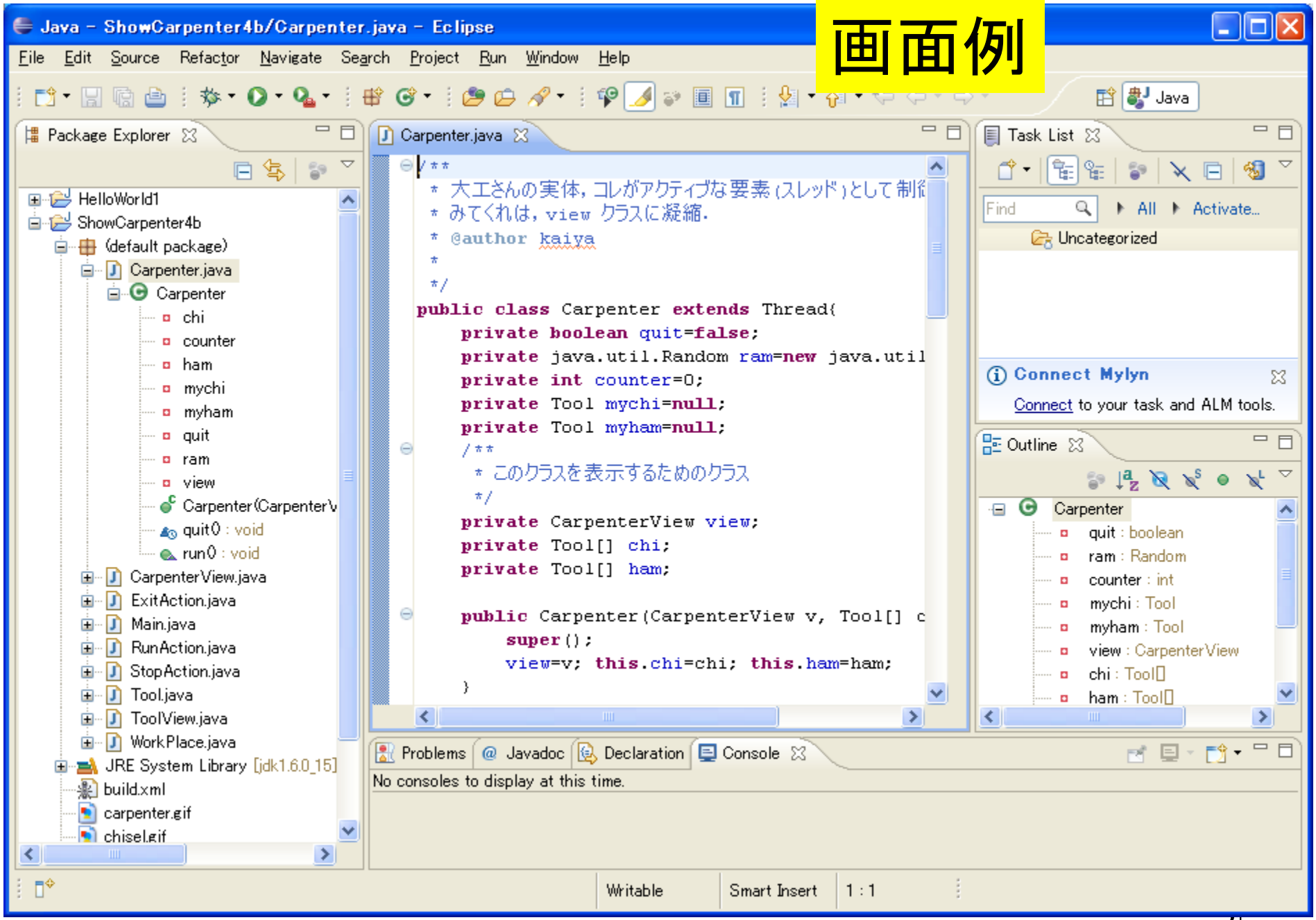
詳細は次回に.

Java と UML

- Java
  - C言語と同様，プログラミング言語です．
  - 代表的なオブジェクト指向言語です．
  - 発案した会社はオラクルに食べられました．
  - 今となっては古参な言語．
- UML
  - 前述のコンピュータにやらせることを整理する道具．
    - というか，整理した結果を書く道具か．
  - ソフトウェア(プログラム)を設計するための図の書き方．図式言語．
  - ソフトウェアの設計書としては今日一般的．
  - フローチャートとかの親戚と当面思ってください．

Eclipse えくりぷす

- プログラムを開発する専用のプログラムの一種.
- Integrated Development Environment (IDE) 統合開発環境と総称されるものの一つ.
- Eclipse はJava専用というわけではありません.
- マイクロソフトのVisual Studio やAndroid StudioもIDEの一種.
- 昨今のプログラム開発では，1,2年生の授業のように，テキストエディタ，コンパイラを直に使うことは稀で，通常，IDEを使う.
- どう使うか，どう便利かはおいおい解説していきます.

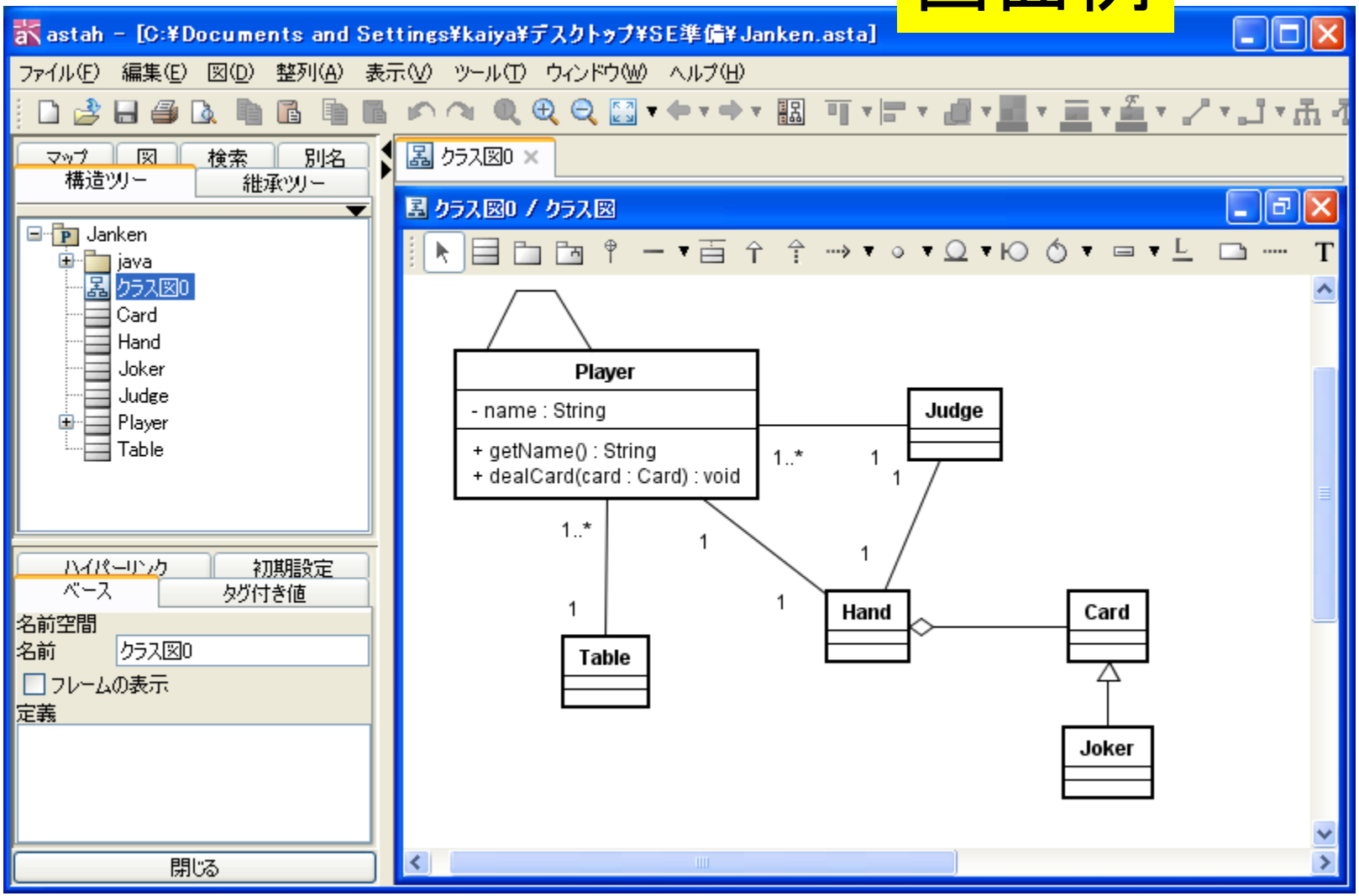

画面例
Java - ShowCarpenter4b/Carpenter.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explorer
HelloWorld1
ShowCarpenter4b
(default package)
Carpenter.java
Carpenter
chi
counter
ham
mychi
myham
quit
ram
view
Carpenter(CarpenterV
quit() : void
run() : void
CarpenterView.java
ExitAction.java
Main.java
RunAction.java
StopAction.java
Tool.java
ToolView.java
WorkPlace.java
JRE System Library [jdk1.6.0_15]
build.xml
carpenter.gif
chisel.gif
Carpenter.java
/**
* 大工さんの実体，コレがアクティブな要素（スレッド）として制御
* みてくれは，view クラスに凝縮．
* @author kaiya
*
*/
public class Carpenter extends Thread{
private boolean quit=false;
private java.util.Random ram=new java.util
private int counter=0;
private Tool mychi=null;
private Tool myham=null;
/**
* このクラスを表示するためのクラス
*/
private CarpenterView view;
private Tool[] chi;
private Tool[] ham;
public Carpenter(CarpenterView v, Tool[] c
super();
view=v; this.chi=chi; this.ham=ham;
}
Task List
Find All Activate...
Uncategorized
Connect Mylyn
Connect to your task and ALM tools.
Outline
Carpenter
quit : boolean
ram : Random
counter : int
mychi : Tool
myham : Tool
view : CarpenterView
chi : Tool[]
ham : Tool[]
Problems Javadoc Declaration Console
No consoles to display at this time.
Writable
Smart Insert
1 : 1

astah

- UMLを描くためのお絵かきツール
- どのようなソフトにするかの設計をする際に利用する.
- この分野では珍しく made in Japan!

- 実装に近い形でソフトを設計すれば，コードのひな型も生成してくれる.
- 概念的な設計にも利用できる，例えばビジネスモデリング等.

画面例


astah - [C:¥Documents and Settings¥kaiya¥デスクトップ¥SE準備¥Janken.asta]
ファイル(F) 編集(E) 図(D) 整列(A) 表示(V) ツール(T) ウィンドウ(W) ヘルプ(H)
マップ 図 検索 別名
構造ツリー 継承ツリー
Janken
java
クラス図0
Card
Hand
Joker
Judge
Player
Table
ハイパーリンク 初期設定
ベース タグ付き値
名前空間
名前 クラス図0
フレームの表示
定義
閉じる
クラス図0
クラス図0 / クラス図
Player
- name : String
+ getName() : String
+ dealCard(card : Card) : void
Judge
Hand
Card
Table
Joker
1..*
1
1..*
1
1
1
1

受講上の注意

- 受講登録を行い，teamsから授業ページ等の情報を得て，オブジェクト指向開発論のページを参照できるように必ずしておいてください.
- 自分のラップトップ(ノートパソコン)は必ず用意してください.

来週までの宿題

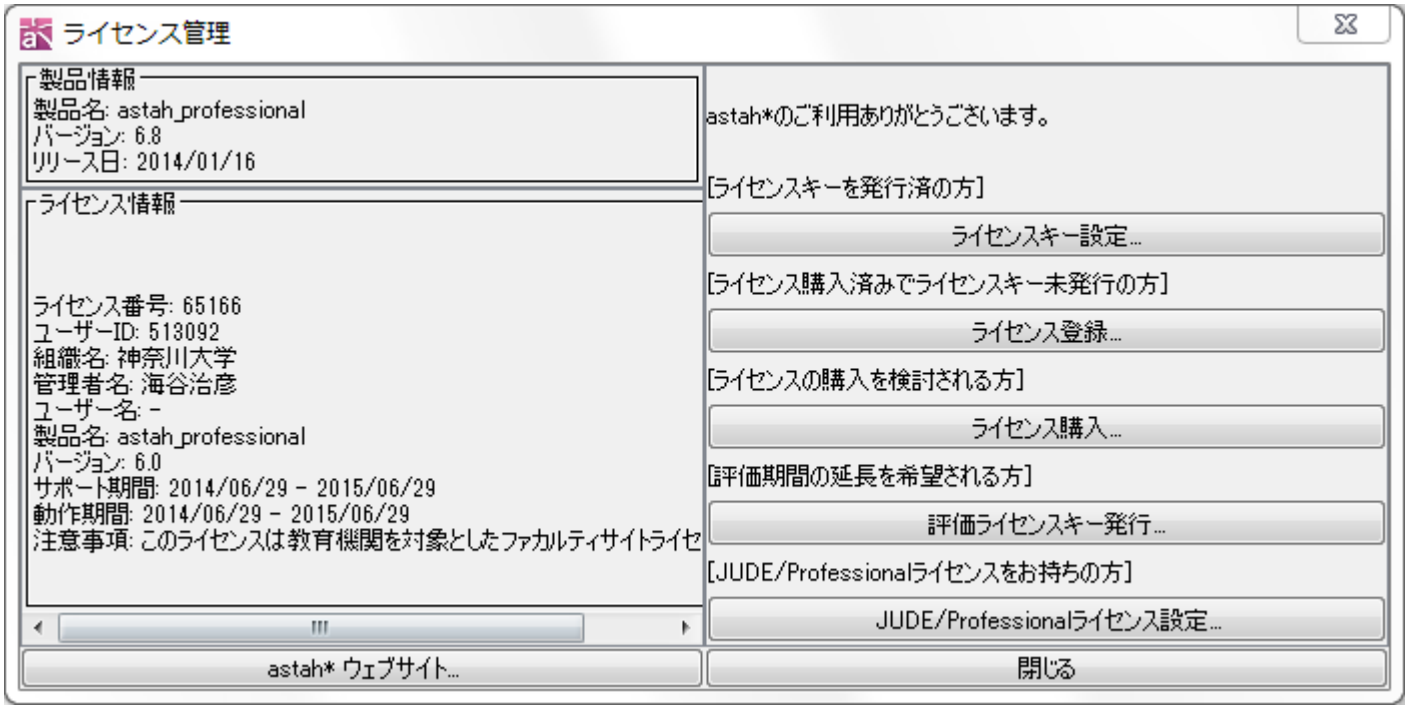
- astah モデリングツールをインストールしておいてください.
- 詳細な手順やデータ，URLは授業のページからリンクがありますが，次頁に概要を提示します.
- ライセンスファイル等はteamsにおいてあります.

手順

1. teamsからライセンスファイルをダウンロード (XML形式)
2. プログラムをダウンロード.
   - astah説明のページにリンクがあります.
3. プログラムを管理者権限でインストール.
4. 初回実行で, プログラムにライセンスファイルを与える. この際も管理者権限で実行が必要.
5. とりあえず, 図 > クラス図から, 適当に画をかいてみてください. (余力があれば)

ライセンスファイルの付与

- ツール > ライセンス設定 より，
- ライセンスキー設定ボタンを押して，ダウンロードしたファイルを選択.



管理者として実行

- メニューから，右クリックで以下のメニューを出す．