ああオブジェクトにあらましかば
ーオブジェクト指向プログラミングー

岐阜経済大学　経営学部 経営情報学科　井戸　伸彦

来歴：

0．0版　2003年9月29日

1．0版　2005年11月28日：全般見直し

スライドの構成

はじめに

- 直感的な説明を第一とし、正確さに欠ける記述を敢えて行っています。
- 本スライドは、オブジェクト指向とはどのようなものであるかを理解することを目的としています。設計するための方法については、別スライドにて勉強します。
- 本スライドでは、オブジェクト指向言語の例として、Javaを想定しています。オブジェクト指向であることと、Javaの仕様であることとの区別を、明示的に書いていない場合もあります。
- ツールとして、Judeを用います。
- スライド中、「Javaオブジェクト指向スライド」と示しているのは、次のスライドを指します。
  - ドゥビドゥバ、ジャバ　ー直感Ｊａｖａのオブジェクトー
    (http://www.gifu-keizai.ac.jp/~ido/doc/java/java_class.pdf)

課題に関して

■本スライドを勉強していく中で、次のような課題を実施します。

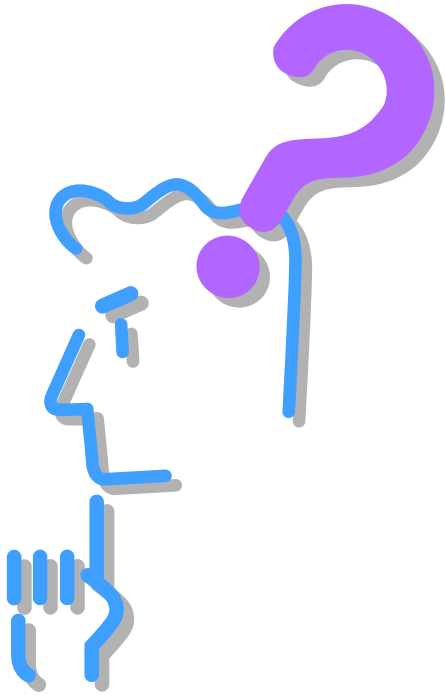
- 本スライド中の基本演習中の課題は、全員が実施します。
- 「Javaオブジェクト指向スライド」を学んでいない人は、そのスライド中の課題を実施します。
- 「Javaオブジェクト指向スライド」をすでに学んでいる人は、本スライド中の発展演習中の課題も実施します。

(1．1) 何を学ぶのか？

■通常、オブジェクトは“もの”という日本語に訳されます。単純な言い方をすれば、“もの”を中心に考えていこうという訳です。

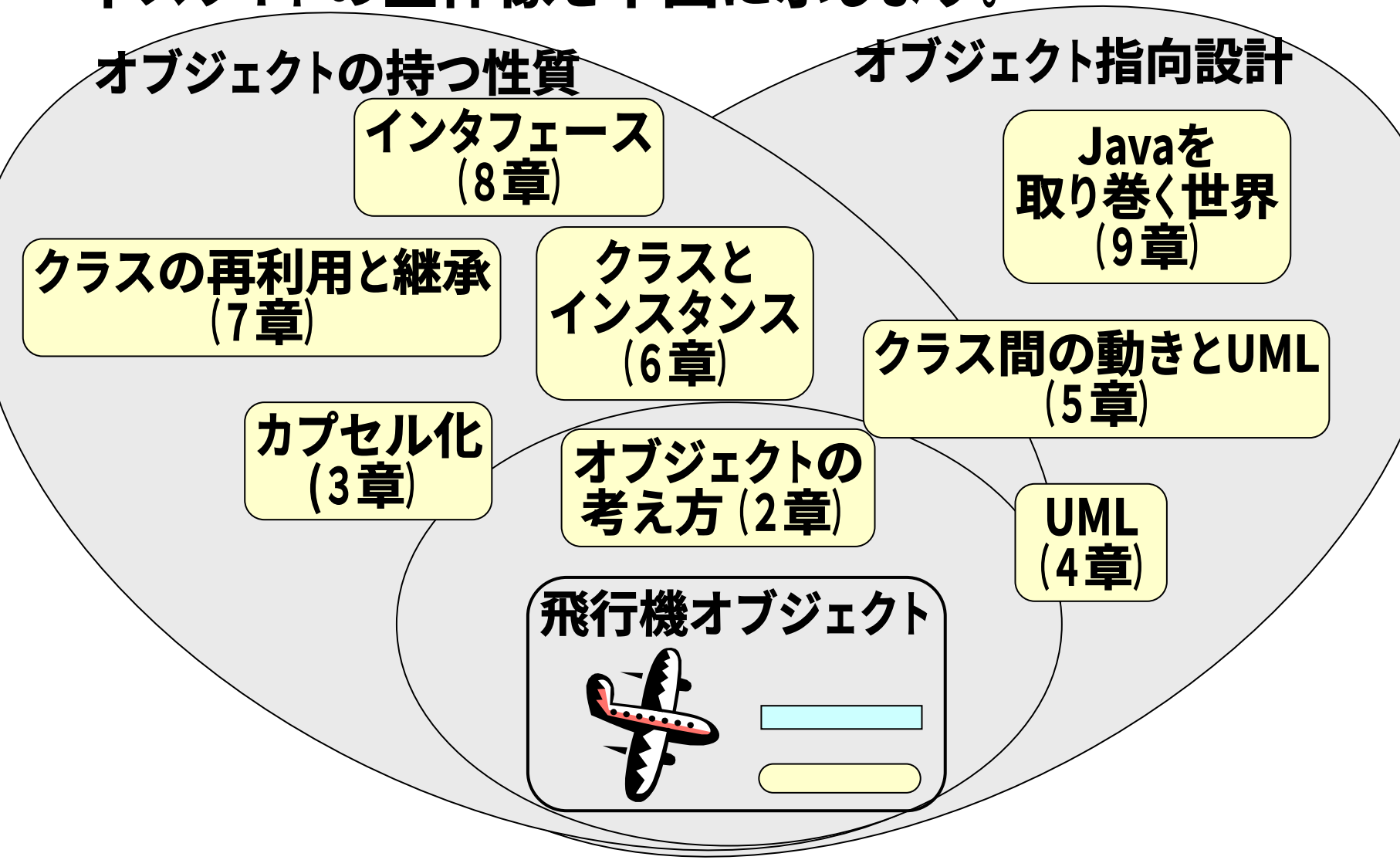
■このような言い方は、漠然としているように思えるかもしれませんが、実際、オブジェクト指向には、なにか厳密で難しい理論があると言うわけではありません。オブジェクト指向とは、“こんなふうに考えていこう”という流儀なのです。

■オブジェクト指向を学ぶことは、考え方の流儀（パラダイムと呼ぶことがあります）を学ぶことになります。



(1．2) 全体像

■本スライドの全体像を下図に示します。



(2) オブジェクトの考え方

(2．1)
オブジェクトとは？

とにかくオブジェクトを見てみよう！

(2．2)
モデリング

何のためにオブジェクトを考えるの？

(2．3)
オブジェクトとプログラム

プログラムの中でオブジェクトは何なの？

(2．4)
なぜオブジェクトか？

どうしてオブジェクトがいいの？

飛行機オブジェクト

(2．5)
Javaでのオブジェクト

Javaでのオブジェクトを少し見てみる。

(2．1) オブジェクトとは？

■“オブジェクト”(もの) とは何かという問いに対して、その定義を与えることは簡単ではありません。実際、何をオブジェクトと捉えるかについては個人により差異が生じ、その結果としてプログラムを作成した場合に違うものが出来ることは、当たり前に起こることです (理論ではなく、考え方の流儀ですから)。

■以下のスライドでは、G. Boochによる次のような考え方 (定義) に従って、例により理解していくことにします。

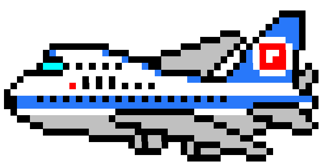
「オブジェクトは以下の3つの性質を持つ」

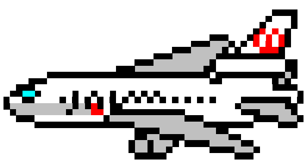
- 状態
- 振る舞い
- 識別性

(2.1.1) 飛行機の例

■飛行機をオブジェクトとして考えます。

- 状態:
  - ◆停止中、飛行中、上昇中、下降中
- 振る舞い
  - ◆離陸する、着陸する、加速する、減速する
- 識別性
  - ◆全日空所有のB747の3号機は、日本航空所有のDC-10の2号機とは、別のものである(識別できる)。





(2．1．2) 学生の例

■学生をオブジェクトとして考えて見ます。

- 状態
  - ◆睡眠中、勉強中、通学中、遊び中、休憩中
- 振る舞い
  - ◆食べる、笑う、叫ぶ、踊る、
- 識別性
  - ◆学生証番号５００１０００のA君と、５００２０００のBさんとは別の学生である(識別できる)。

(2．2．1) “休講”はオブジェクトか？

■オブジェクトとして、“コップ”を考えて見ます。

- コップの“状態”は何でしょうか？
  “空”、“水で満たされている”とかが
  考えられるかも知れません。
- “振る舞い”は何でしょうか？
  “割れる”、“傾く”とかが考えられるかも知れません。

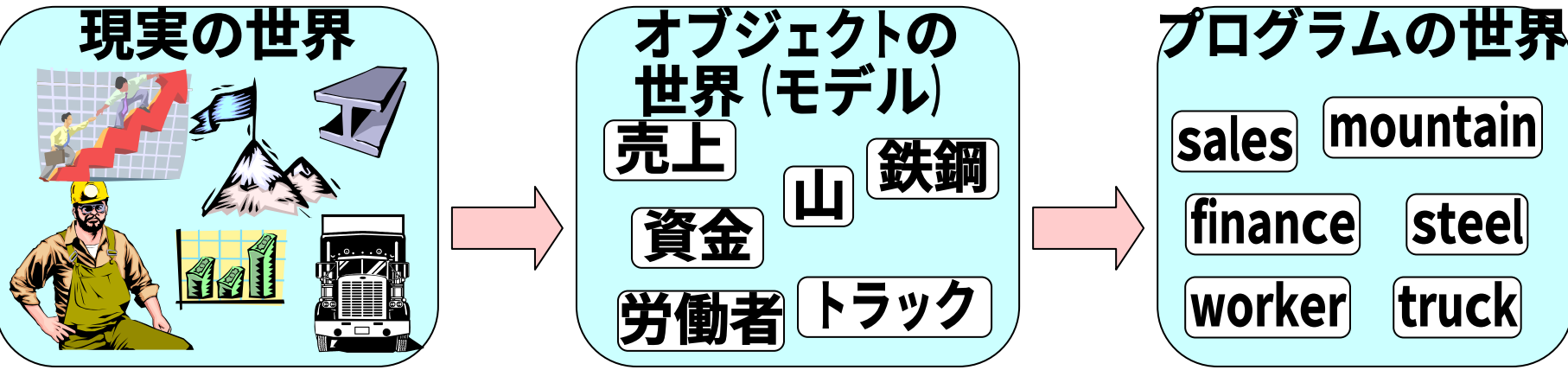
■“コップ”の例は、前の“飛行機”や“学生”よりは考えにくいかも知れません。コップをオブジェクトとして捉える時、正解はどうなるのでしょうか？

■“休講”をオブジェクトとして考えるとどうなるのでしょうか？

■オブジェクトになりやすいものとなりにくいものがあるのでしょうか？

(2．2．2) システムの目的とオブジェクト

- ■何を目的とするかを決めないと、何をオブジェクトとするかを考えることは、あまり意味がありません。
- ■オブジェクトは現実の世界から抽出され、後述するようにそれがソフトウェアの単位となります。
- ■“現実⇒オブジェクト⇒ソフトウェア”と考えていく中で、システムの目的は常に意識しなければなりません。
- ■次の2つのスライドで、例を見てみます。



（２．２．３）野球部学生の例

■現実の世界の、“野球部の学生”について考えて見ます。現実では同じ“もの”（＝野球部の学生）が、目的により、別のオブジェクトとして考えることが出来ます。

現実の世界

（野球部の学生）

→ **オブジェクトの世界（モデル）**

■状態：打席中、守備中、
■振る舞い：ヒット、捕球、
■識別性：背番号５、１０、

→ **プログラム**

試合記録プログラム

目的：野球の試合の記録

→ **オブジェクトの世界（モデル）**

■状態：講義中、休学中、
■振る舞い：レポート提出、
■識別性：学籍番号5001000

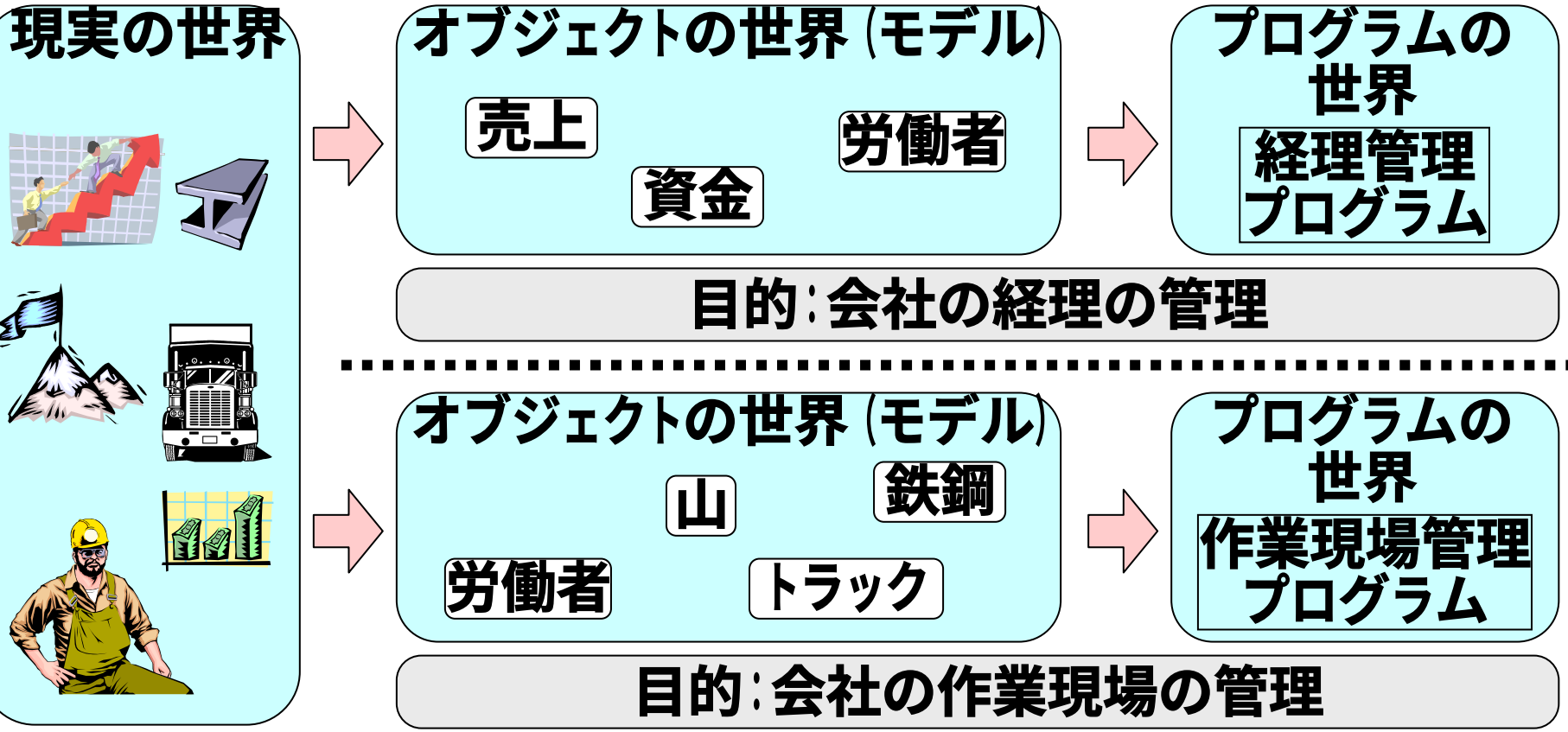
→ **プログラム**

学籍管理プログラム

目的：学生の学籍の記録

(2．2．4) 会社の例

■何をオブジェクトとして現実の世界からつむぎ出すかも、システムの目的により、変わってきます。

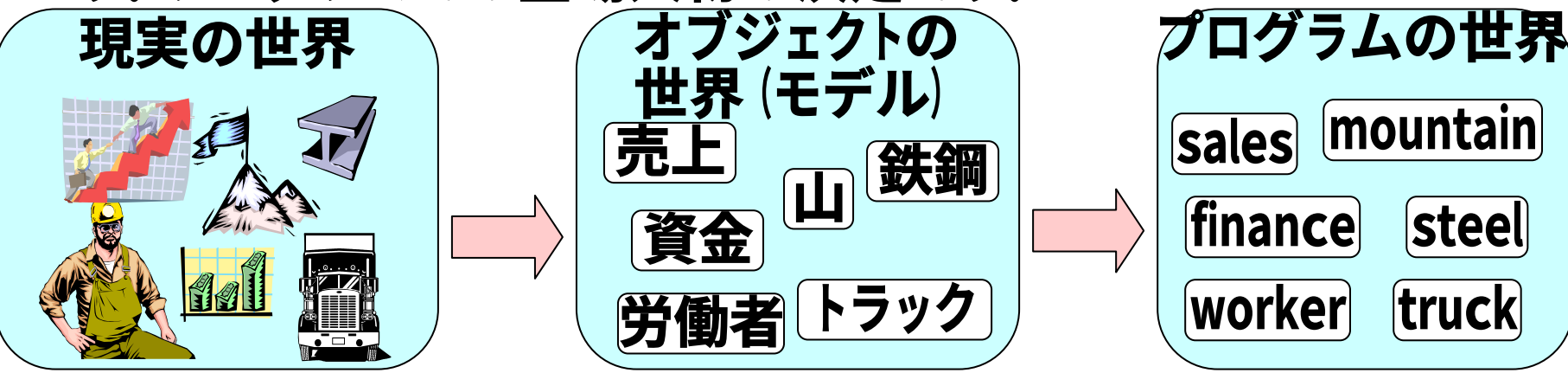


(2．2．5) モデリング

■モデルとは、現実の世界の実体に対して、目的に応じた限られた解釈を与えたものを言います。

■現実の世界の実体は複雑で、そのすべての記述することは困難ですが、目的に必要な性質のみを取り上げることで、簡潔にモデルとして記述します。

■我々の目的はシステム開発です。システムの目的に沿った性質を持つオブジェクトを抽出することがモデリングというわけです。プログラムでの登場人物の決定です。



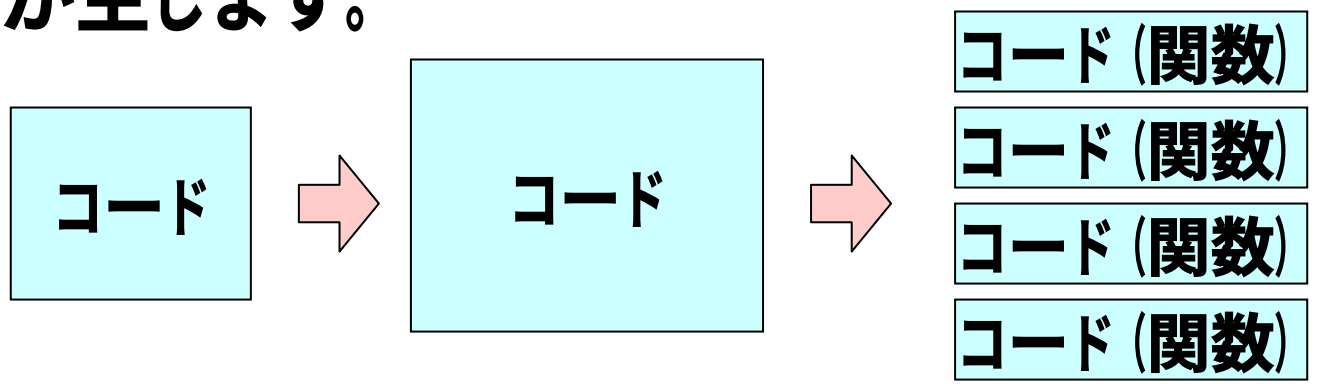
システム (ソフトウェア) の目的

(2．3) プログラムの中のオブジェクト

■プログラムの中でオブジェクトとは何かについて見ていきます。

■ソフトウェアは、コード (if文,while文,代入文,etc.) とデータ (文字変数、) から成り立っています。

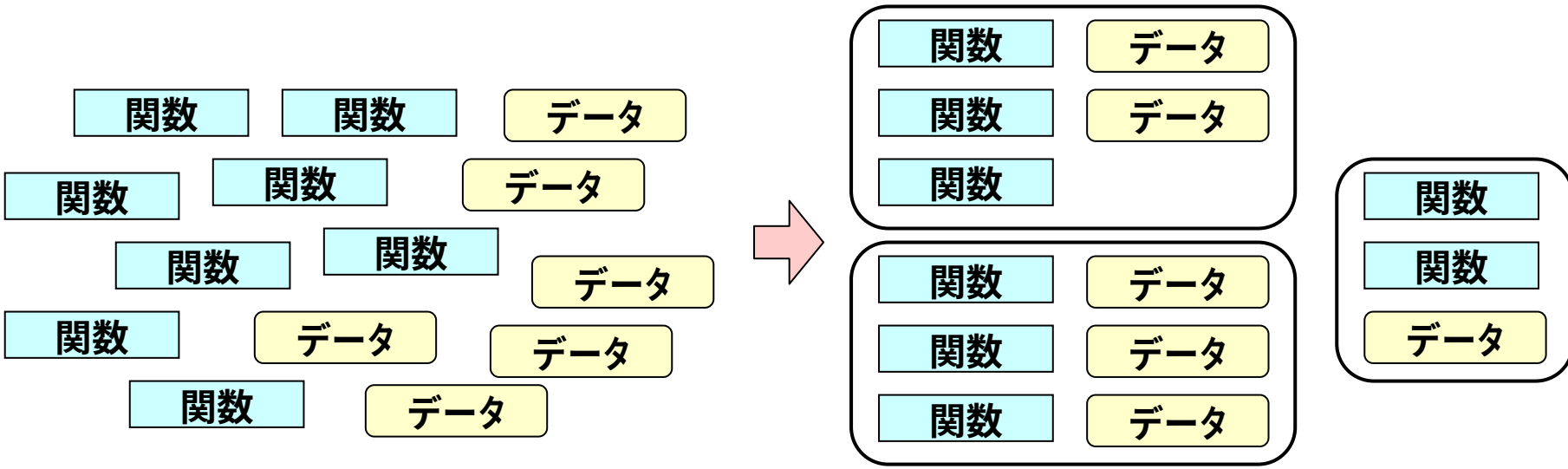
■小さなプログラムではコードを分割する必要を感じることはありませんが、大きなプログラムでは、関数 (サブルーチン、プロシージャ、ファンクション) に分割する必要が生じます。



(2．3．1) プログラムのまとまり

■関数 (サブルーチン、プロシージャ、ファンクション) の数が増え、データの数も増えていくと、整理がつき難くなります。

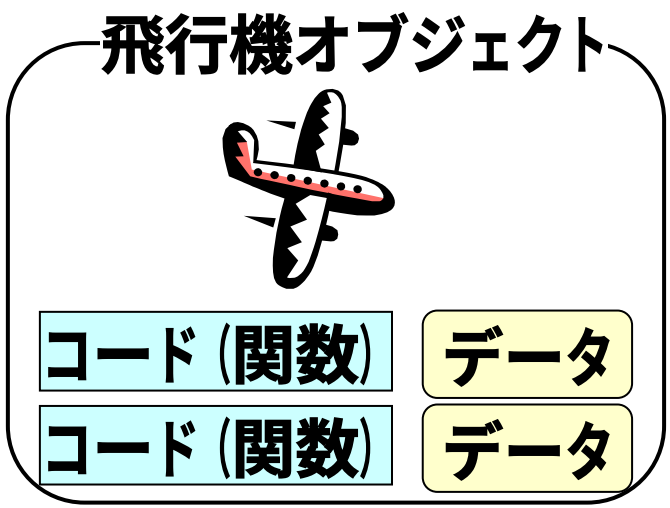
■複数の関数とデータとをひとまとめにしておくことで整理をつけることは、オブジェクト指向以前のソフトウェアでも実施されていました。



(2．3．2) どのようにまとめるか？

■プログラム (コードとデータ) をどのようにまとめるか、どのように整理をつけるかは、ソフトウェアを設計する上で本質的な問題です。これがうまくいっていないと、ソフトウェアは開発･保守はうまくいきません。

■オブジェクト指向プログラムでは、このまとまりをオブジェクトとします。すなわち、前にみたオブジェクトの例であれば、“飛行機オブジェクト”とか、“学生オブジェクト”を作る訳です。



(2．3．3) 状態、振舞い

■先ほどみた「状態、振る舞い、識別性」によるオブジェクトと、ソフトウェアの中のオブジェクトとは、次のように対応します。

“もの”(飛行機) と認識されるものをプログラムのまとまり (単位) であるオブジェクトに対応させる。

飛行機オブジェクト

コード (関数)	データ
コード (関数)	データ

※ 識別性については、後で説明します。

“もの”(飛行機) の振る舞いを記述する

■振る舞い：・離陸する、着陸する、上昇する、下降する、

“もの”(飛行機) の状態を覚えておく

■状態：・停止中、飛行中、上昇中、下降中

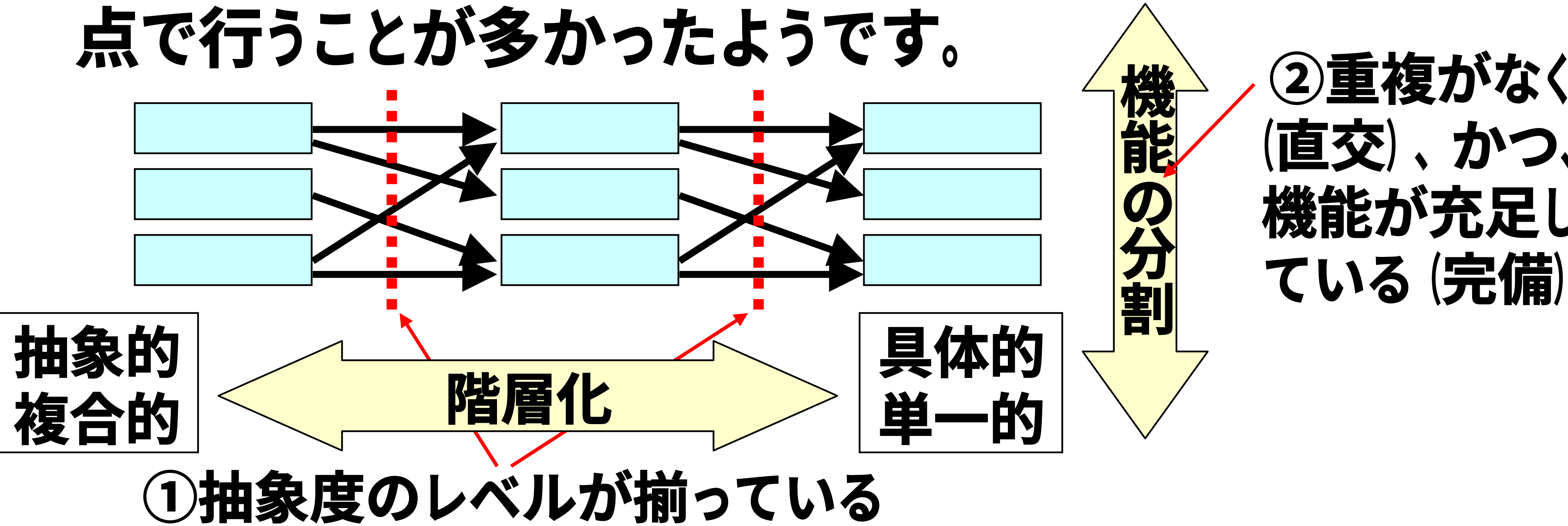
(2．4) なぜオブジェクトか？

■ここまで (2) 項には、“プログラムの単位としてのオブジェクト”について記してきました。なぜ、プログラム単位をオブジェクトとするかについて、考えていきます (スライド(1.2) に記したオブジェクト指向のさまざまな性質・機能も関係するのですが、ここではそれらには触れません) 。

■オブジェクト指向以前のソフトウェアでは、どのような観点でプログラムの単位を決めていたのでしょうか？これについては、「構造化プログラミング」のスライドにて既に勉強しました。

(2．4．1) 階層化･機能分割

■階層化･機能分割では、コードの分割を次のような観点で行うことが多かったようです。



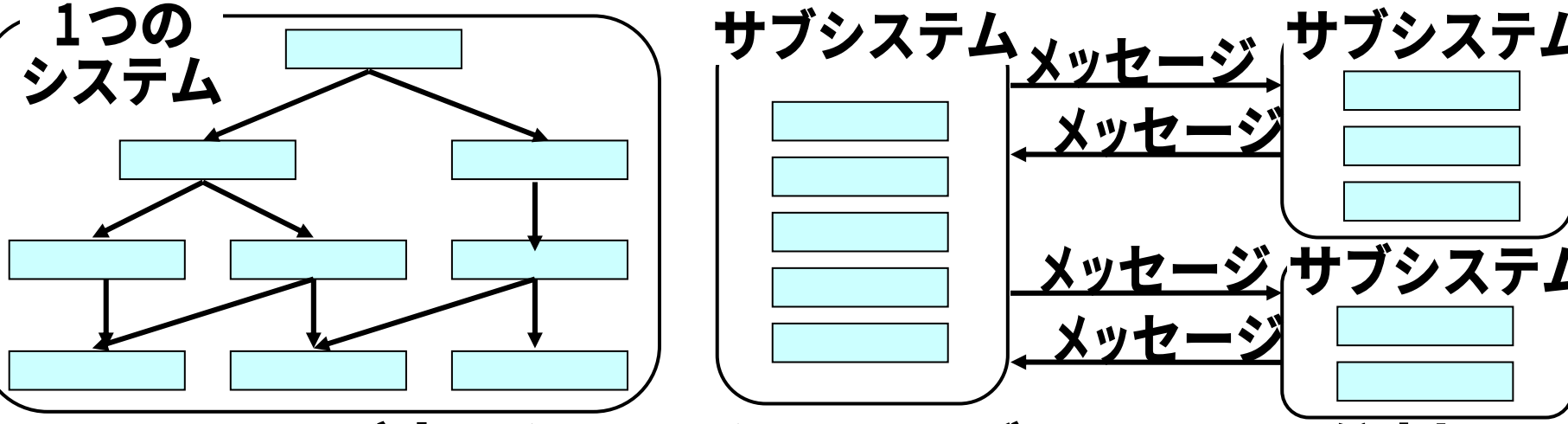
■上記のような階層化･機能分割の考え方それ自体に不備があるというわけではありません。

■以下のスライド(2.4.2)(2.4.3)では、オブジェクト指向と階層化･機能分割との対比を見ていきます。

(2．4．2) サブシステム化

■階層化されたシステムでは、全体が大きなピラミッド型のプログラム構成になる場合が多いようです。

■オブジェクト指向のシステムは、相互に作用する独立したサブシステムとして構成する場合が多いようです。



■システムが大きくなってくると、サブシステムに分割して考えたほうが有利になってきます。

■サブシステム間 (オブジェクト間) の作用を、メッセージと呼ぶことがあります。

(2．4．3) 自然な設計

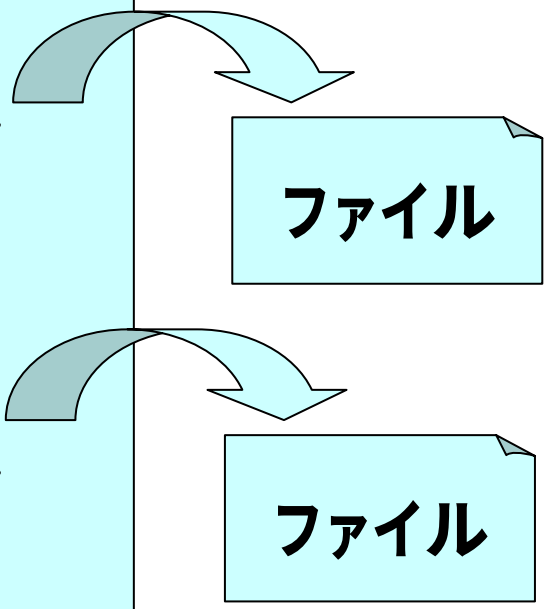
■機能分割においては、なるべく自然な機能を定義するとしても、多くの人に共通な感覚にまとめることはなかなか困難です。

■オブジェクトを単位にすると、多くの人にとって自然な単位とすることが容易になります。

(2．5) JAVAでのオブジェクトの記述

■Javaでは、オブジェクト(クラス)を次のように記述します。

```
class Hikouki{
# ここに“飛行機オブジェクト”の
# コードとデータが入る
}
class Gakusei{
# ここに“学生オブジェクト”の
# コードとデータが入る
}
```



■1つのファイルに1つのオブジェクト(クラス)を入れておくことが普通です。ファイル名は、オブジェクトの名前と同じにします。

■Javaオブジェクト指向スライド(2)

(2．6) 基本演習：ゲームプログラムの起動

■“ブラックジャック”のプログラムを、井戸のネットワークドライブからダウンロードして、動かしてみます。

- 井戸のネットワークドライブから、“マイドキュメント”の下の“javawork”のフォルダに、次のフォルダをコピーし、コンパイルしてください。
  - ◆“blackjack”
- [start]-[すべてのプログラム]-[アクセサリ]-[コマンドプロンプト]を選択してください。
- 次のようなイメージで動作します。

```
> cd javawork              ← javaworkのフォルダに移動
> java Window    ← Window.classのプログラムを起動
```

- eclipse上ではインポートして実行します。

(2．6．1) 基本演習：ゲームの説明

■ブラックジャックの簡単なルールを説明します。

- ジョーカーを除いた52枚のトランプを使います。
- 持ち札が21に最も近いプレーヤーが勝ちになります。22点以上になると、バーストとなり、その時点で負けになります。
- 点数を計算するとき、絵札 (J、Q、K) は10点です。
- エース (A) は、1点と11点とのいずれの数え方も出来ます。
- コンピュータがディーラー (画面右) となって、ユーザ (画面左) と1対1で勝負します。
- [new]ボタン：ゲーム開始です。
- [hit]ボタン：カードを要求します。
- [stand]ボタン：カードを要求せず、現在の持ち札で勝負します。
- [shuffle]ボタン：カードをリフレッシュ (52枚のカードから開始) します。
- [exit]ボタン：プログラムを終了させます。

(2．6．2) 基本演習 (課題2－1、2－2)

■ (課題2－1) このゲームの中のオブジェクトを考えて見てください。

- 必ずしも、答えが一通りになる訳ではありません。
- みんなで話し合ってみましょう。

■ (課題2－2) コピーしたフォルダの中のファイルに、どのようなオブジェクトをもつプログラムがあるかを見てください。

- コピーしてきた“src.windows”のフォルダを開き、さらに、その中の“blackjack”のフォルダを開いてください。
- このなかの、拡張子が“.java”であるファイル (****.java) を、ダブルクリックにより開いてください。
- ファイルの中身については、スライド (2.5) で見た、“class”のみに注目すれば結構です。

(2．6．3) 発展演習 (課題2－3)

(この課題は、Javaオブジェクト指向スライドを勉強したことのある方に向けたものです。)

■課題2－3

- 配布したblackjackのプログラムでは、標準パッケージを利用していません。次のクラスにおいて、ArrayListを利用することにより、プログラムを簡単にしてください。
  - ◆Player.java
  - ◆CardDeck.java