ソフトウェア工学実習
Software Engineering Practice
(第02回)

SEP02-001 オブジェクト指向とは
概念編

慶應義塾大学・理工学部・管理工学科
飯島 正
iijima@ae.keio.ac.jp

この授業の狙い

- 前提科目
  - ２年生秋学期のソフトウェア工学
    - （未履修者は頑張ってついてきてください．
      必要なら補習をします）
- この授業の目標
  - この授業の目標は，
    オブジェクト指向の概念を
    実感をもって理解し活用できるようになる
    ことです．

iijima@ae.keio.ac.jp

１．はじめに

- 1. はじめに
  - 目的：GUI (Graphical User Interface) を通して
    MVC (Model-View-Controler) の概念に基づいて
    オブジェクト指向を理解する
  - まずは，ごく簡単に，オブジェクトの概念をお話しします．
  - 次は，とにかく，GUI (Graphical User Interface) を
    作ってみましょう!!
  - 次回は，NetBeansという
    IDE (Integrated Development Environment: 統合開発環境)
    も使ってみます

iijima@ae.keio.ac.jp

【概念編】
オブジェクト指向の基本概念

オブジェクト

クラス

属性とメソッド

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

↓

エレベータのモデリング

iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

↓

エレベータのモデリング

iijima@ae.keio.ac.jp

オブジェクト指向とは…

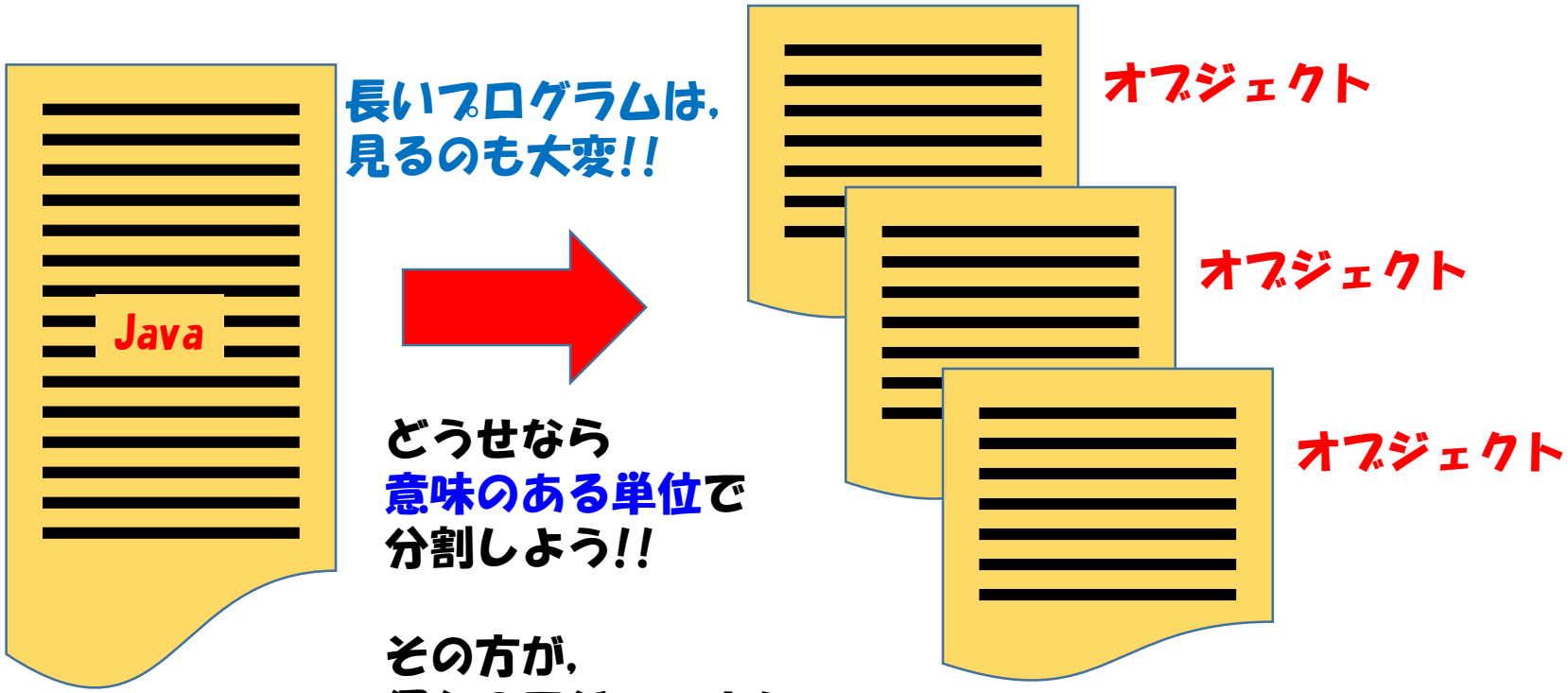
- **オブジェクト指向**とは…

- プログラムを
  オブジェクトの集まり
  で創る
  考え方（**プログラミングパラダイム**）
  です…

iijima@ae.keio.ac.jp

長いプログラムをモジュール単位に分割

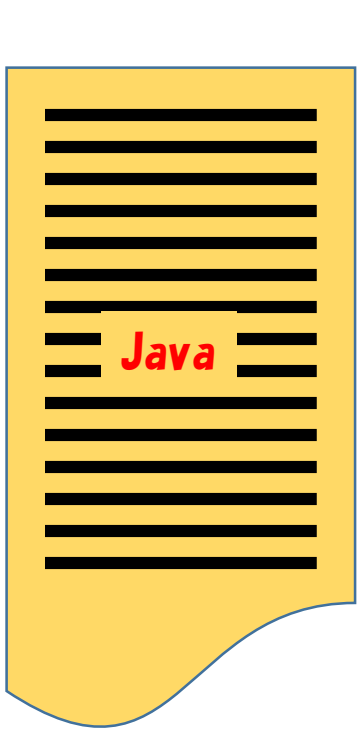
- Java（オブジェクト指向手続き型言語）では，
  モジュール単位に，オブジェクト
  という考え方を採用する



iijima@ae.keio.ac.jp

長いプログラムをモジュール単位に分割

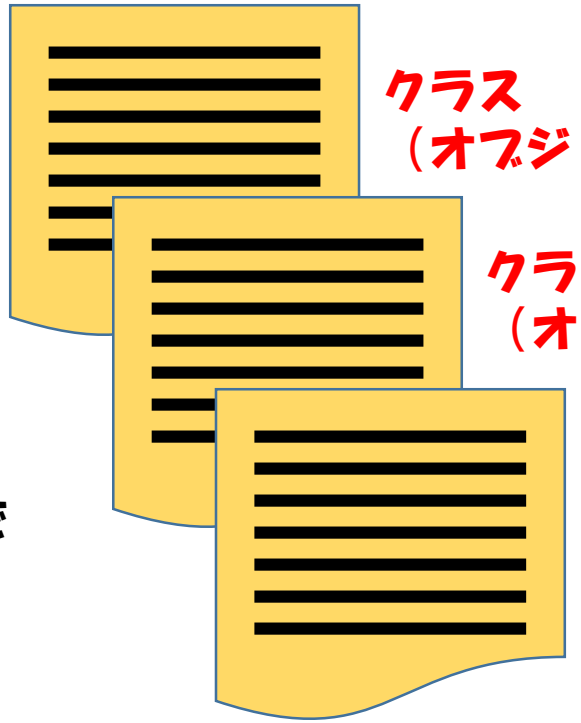
- Java（オブジェクト指向手続き型言語）では，
  モジュール単位に，オブジェクト（の定義であるクラス）
  という考え方を採用する



長いプログラムは，
見るのも大変!!

どうせなら
意味のある単位で
分割しよう!!

その方が，
個々のモジュールを
単独で理解しやすい



クラス
（オブジェクトの定義）

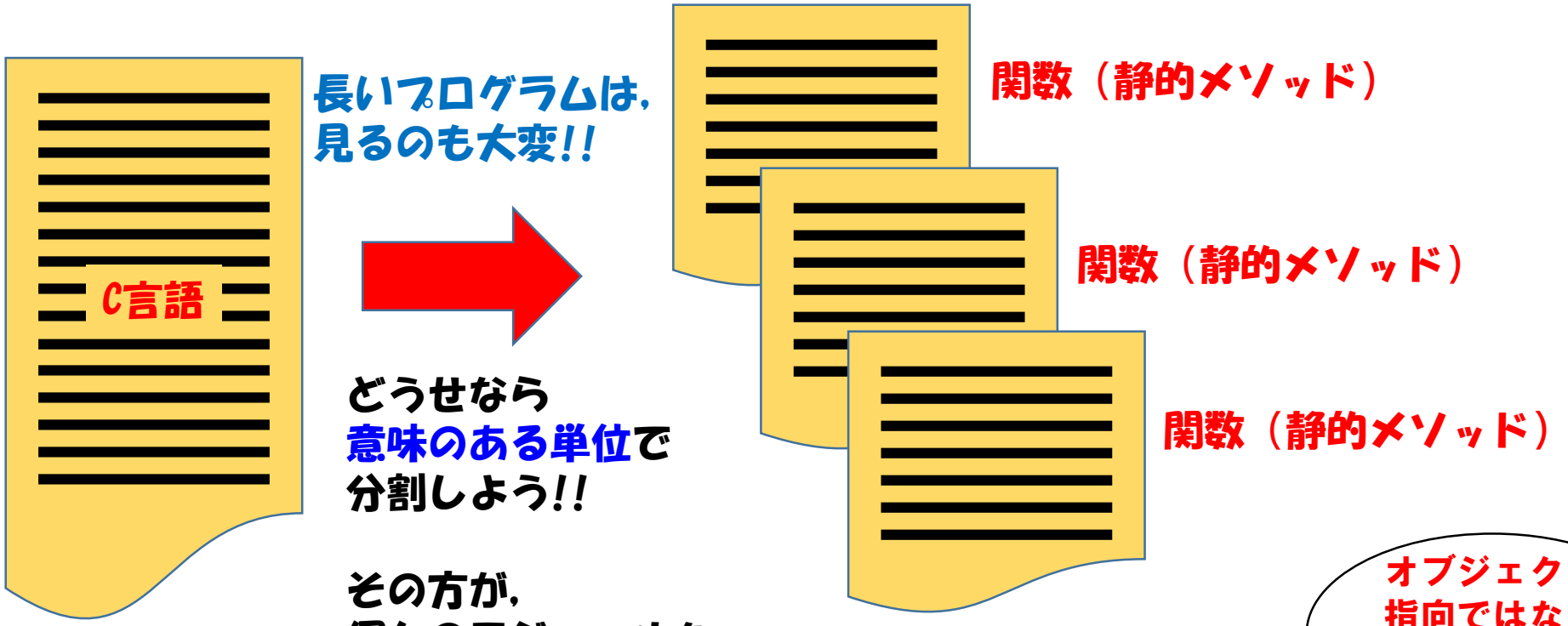
クラス
（オブジェクトの定義）

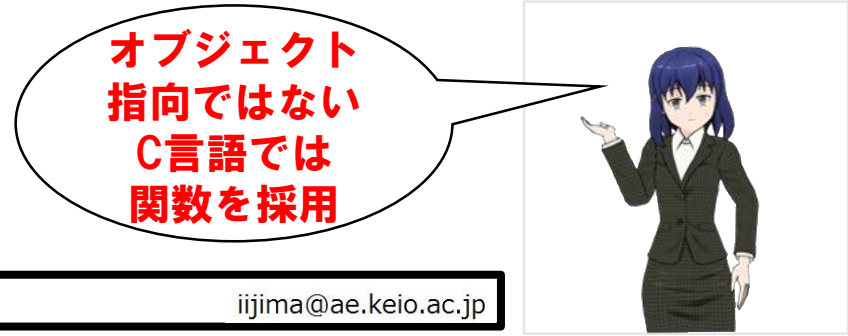
クラス
（オブジェクトの定義）

正確には，
オブジェクト
の定義である
クラスを採用

長いプログラムをモジュール単位に分割

- オブジェクト指向ではない手続き型言語のC言語では，
  モジュール単位に，関数（静的メソッドに相当）
  を採用している．

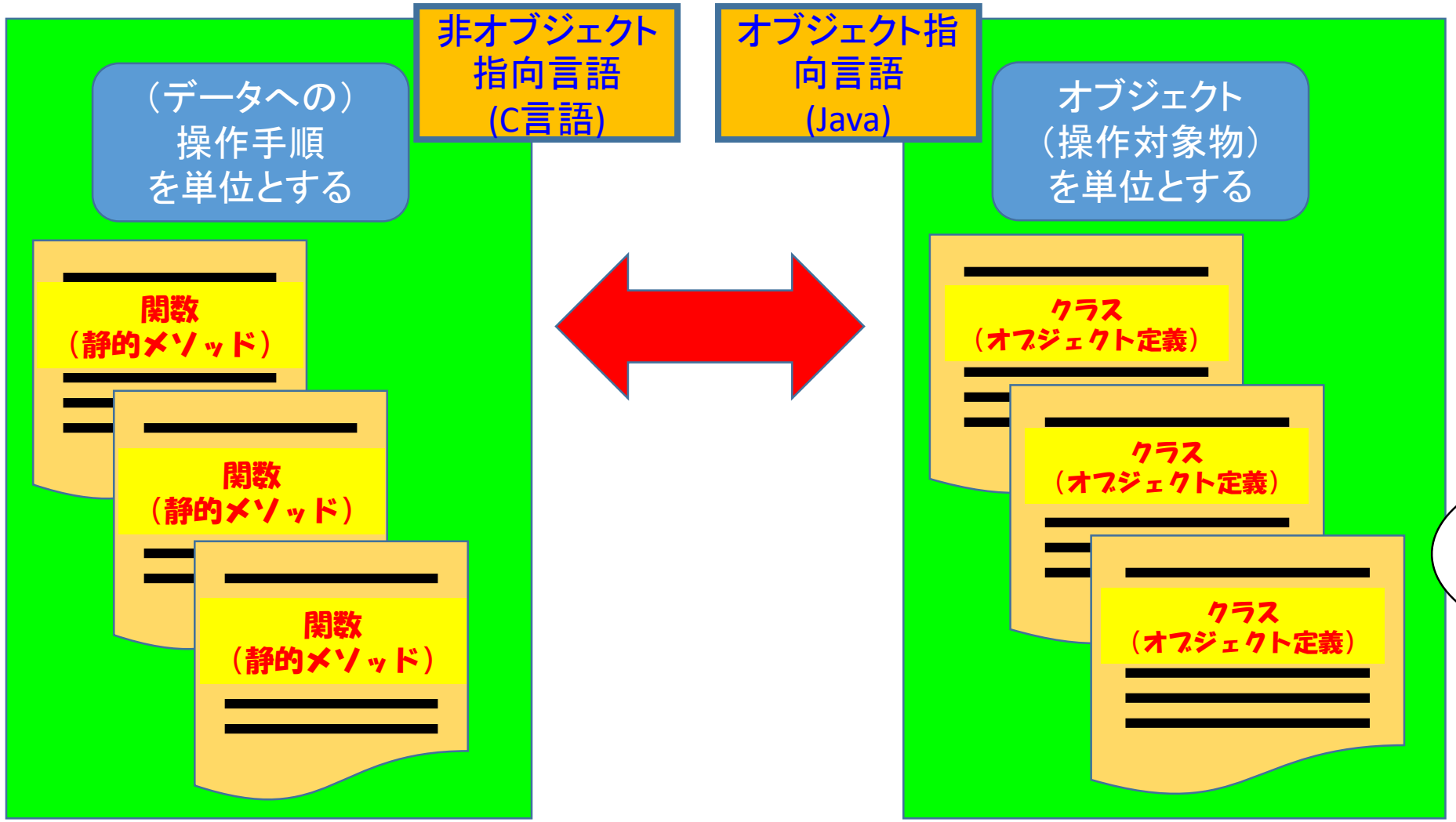


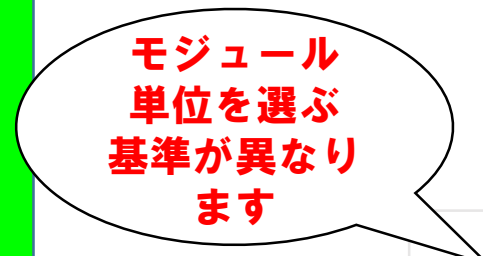


iijima@ae.keio.ac.jp

モジュール単位の違い

- C言語とJavaでは，**モジュール単位を選ぶ基準（意味）**が異なる





iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

↓

エレベータのモデリング

iijima@ae.keio.ac.jp

オブジェクト（Object）

・オブジェクト
　＝その定義がクラス

・オブジェクトを
　スイカ（西瓜）
　で考えましょう

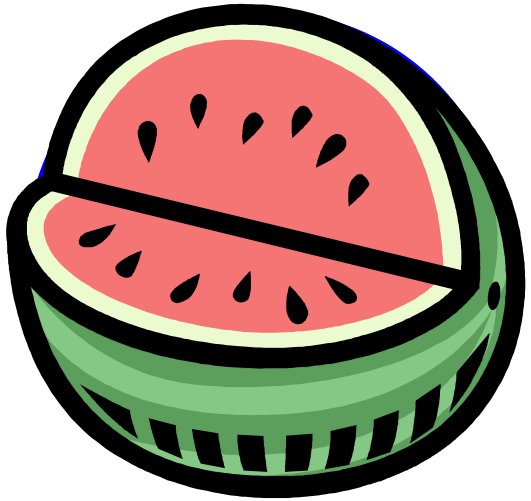
・プログラムは，
　スイカの集まりです

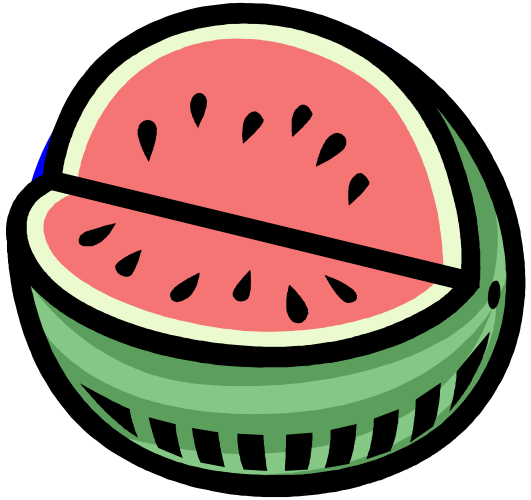
iijima@ae.keio.ac.jp

オブジェクト（Object）

・オブジェクト
　＝その**定義**が**クラス**

・スイカ（西瓜）を割って，
　中身の構造
　を見てみます





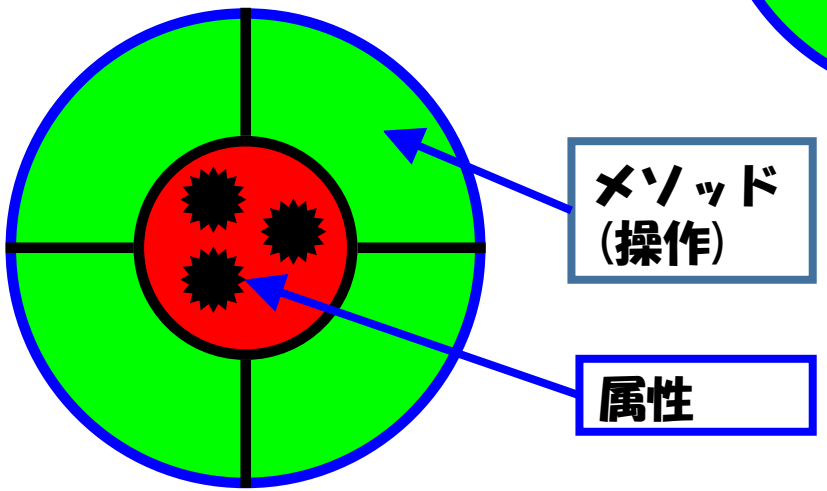
・スイカの中身の構造を
　定義するのが**クラス**です

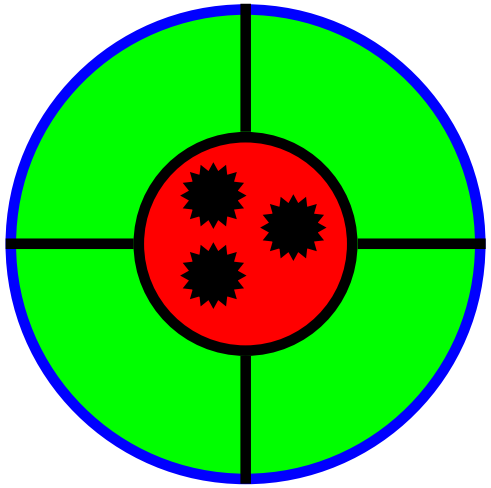
iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

- スイカのように皮（メソッド）が
  種（属性）を守っている
- メソッドを介して属性にアクセスする



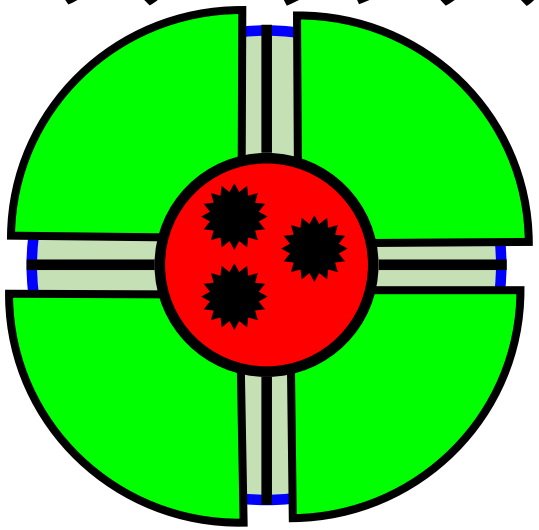


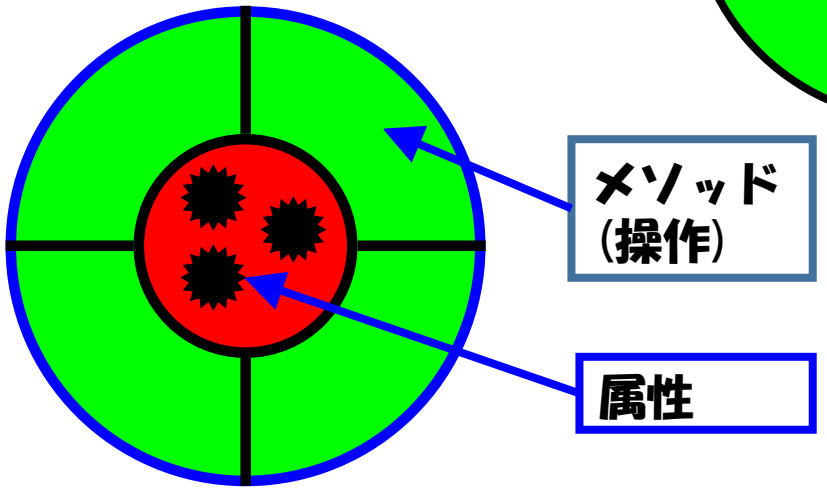
iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

- スイカのように皮（メソッド）が
  種（属性）を守っている
- メソッドを介して属性にアクセスする



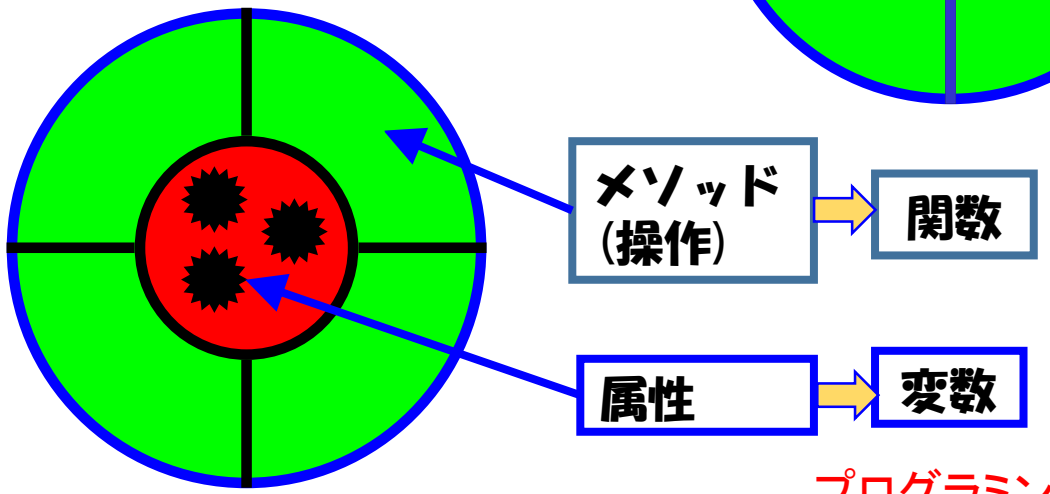


iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

・スイカのように皮（メソッド）が
種（属性）を守っている
・メソッドを介して属性にアクセスする



プログラミングの
概念で言えば…

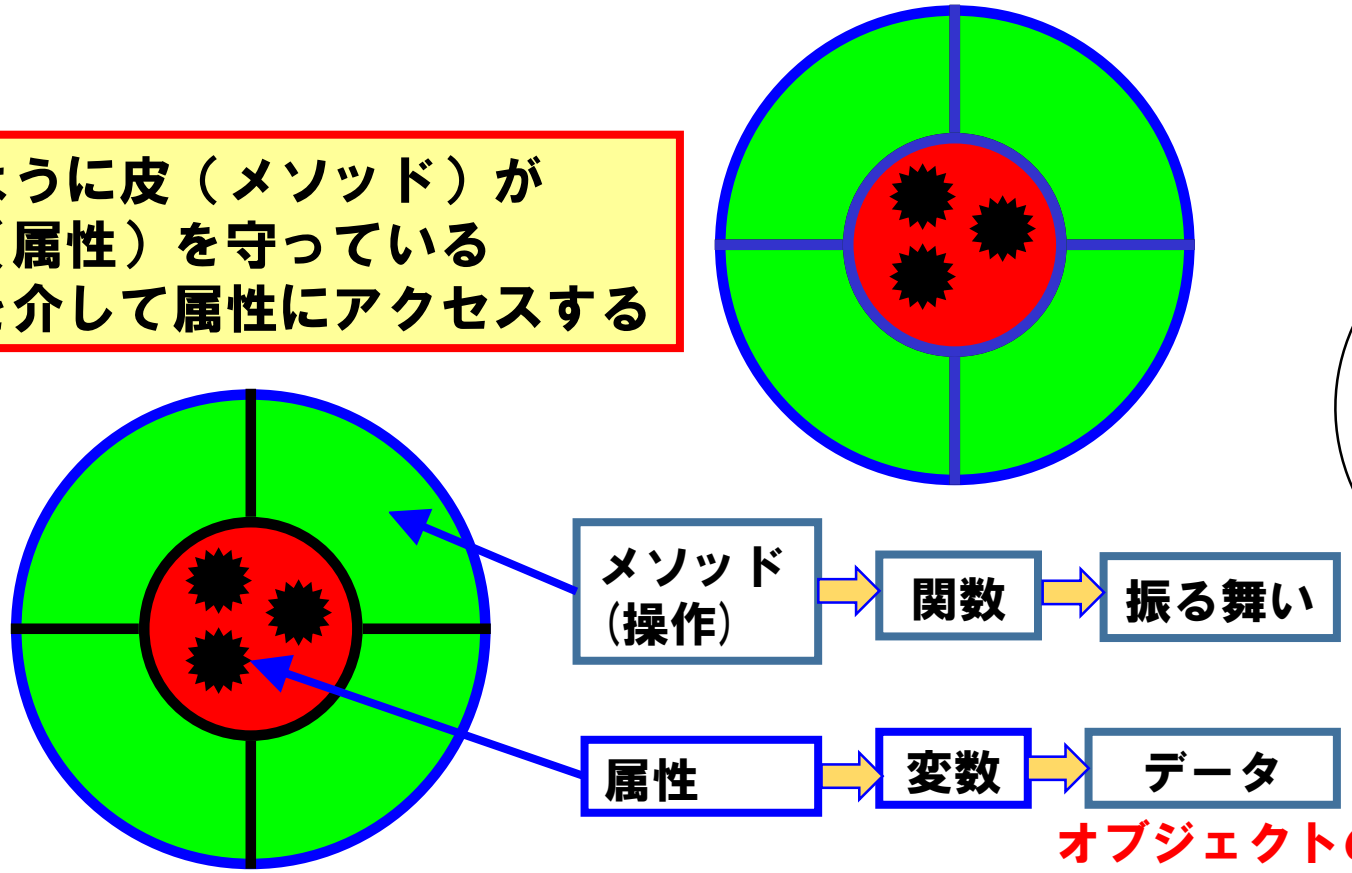
メソッドは
関数，
属性は変数に
相当します

iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

・スイカのように皮（メソッド）が
種（属性）を守っている
・メソッドを介して属性にアクセスする



それぞれ
オブジェクト
の持つ
振る舞いと
固有のデータ
を意味します

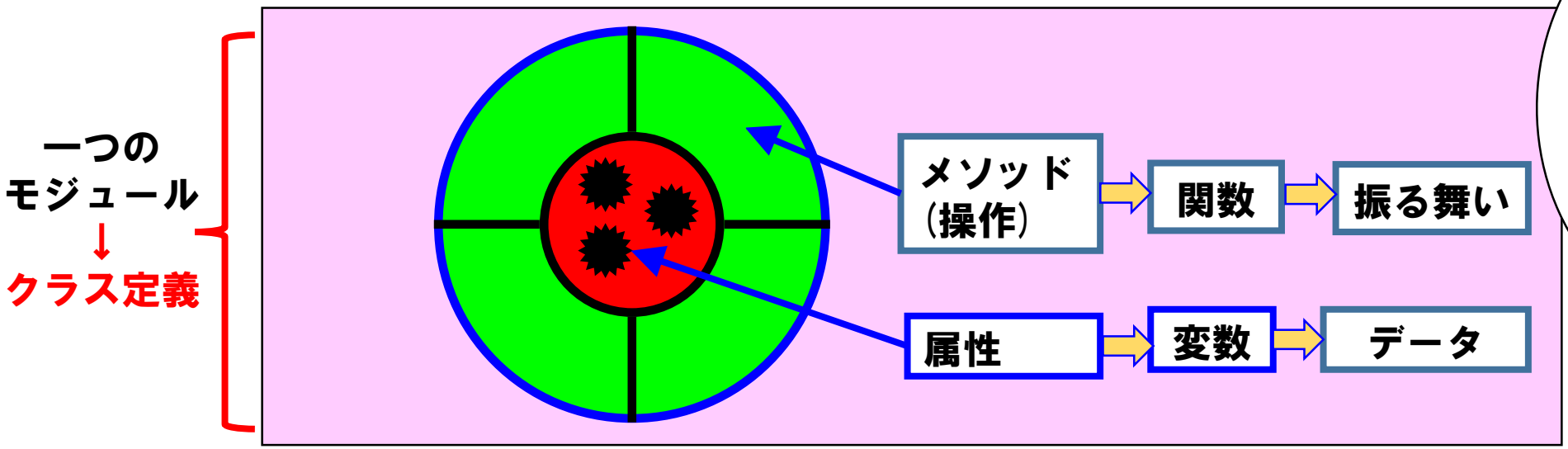
オブジェクトの
どういう側面を
表現するかと言うと…

iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

- スイカのように皮（メソッド）が
  種（属性）を守っている
- メソッドを介して属性にアクセスする



強く関連する
属性と
それに対する
操作を
クラスに
まとめます

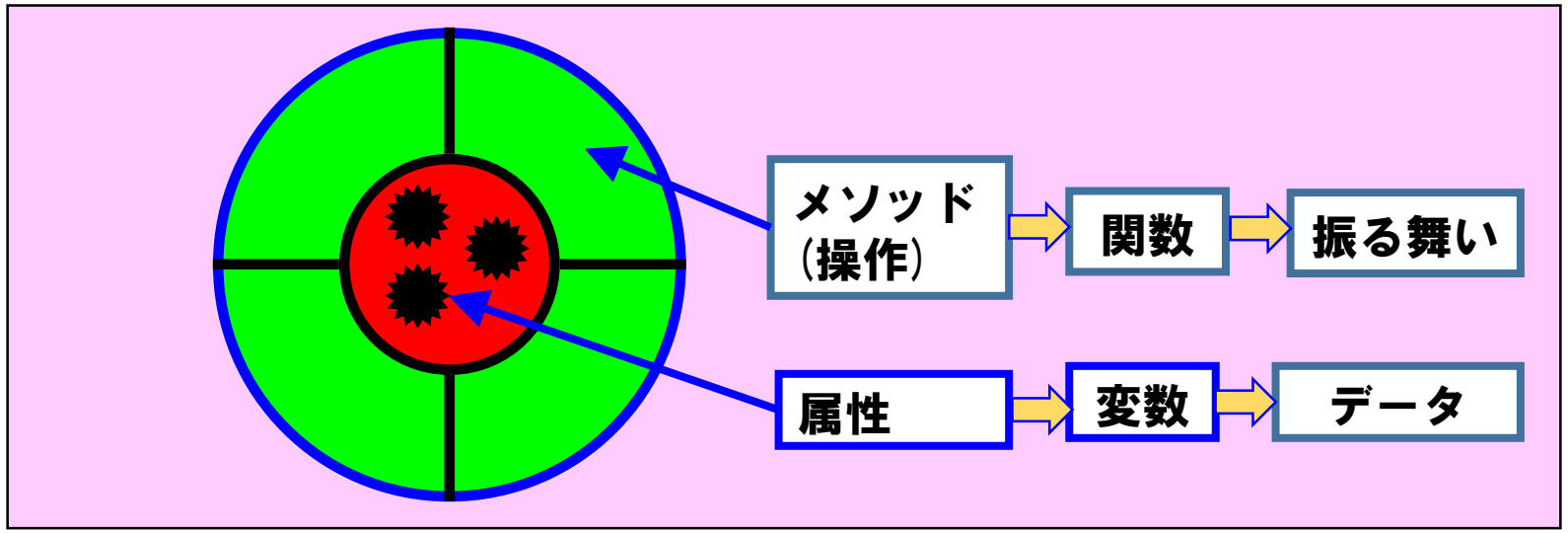
iijima@ae.keio.ac.jp

オブジェクト（Object）

- **カプセル化**
  ＝関連する**属性**と**メソッド**をパッケージングすること

・スイカのように皮（メソッド）が
種（属性）を守っている
・メソッドを介して属性にアクセスする

一つの
モジュール
↓
クラス定義



属性には，
メソッドを
介さないと
アクセス
できません

iijima@ae.keio.ac.jp

話の流れ

- オブジェクト指向とは…モジュール単位
- オブジェクトはスイカ…内部構造は属性とメソッド
- **ウィンドウもオブジェクト**
- オブジェクトはメッセージに反応する
- クラスは鯛焼き器…オブジェクトはクラスから作る
- クラスとインスタンス
- 実世界のモデル化
- エレベータのモデリング

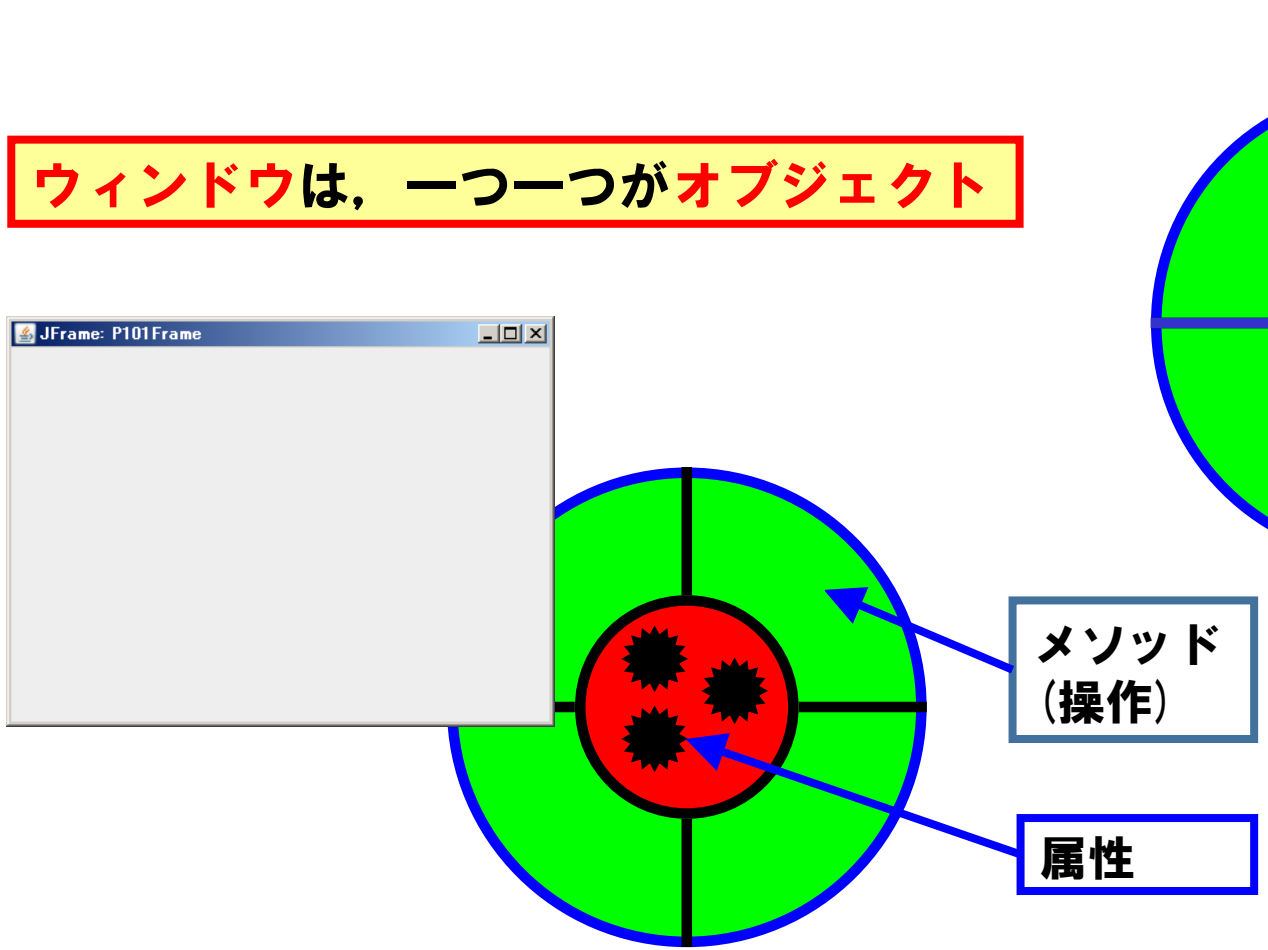
実際の
プログラムで
考えましょう.

iijima@ae.keio.ac.jp

ウィンドウもオブジェクト

- ウィンドウをプログラミングするとき…

ウィンドウは，一つ一つがオブジェクト

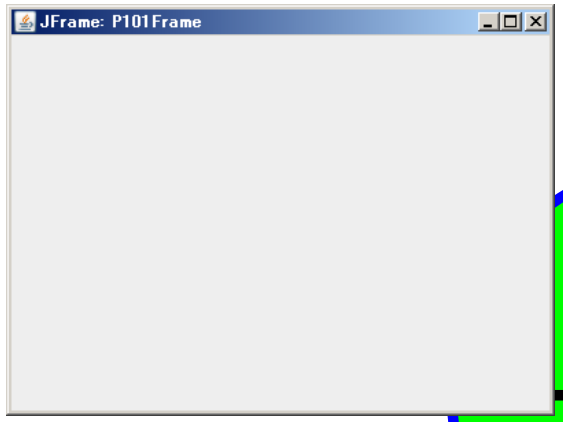


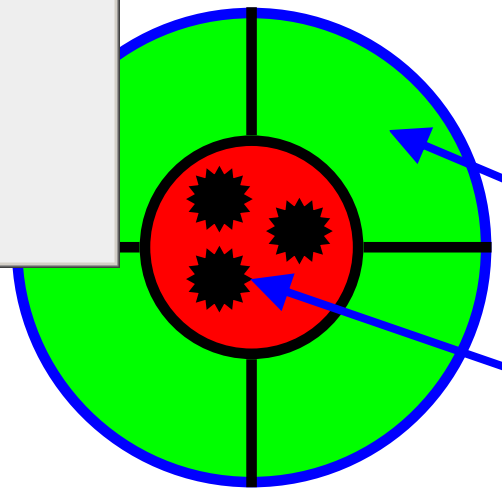
iijima@ae.keio.ac.jp

ウィンドウもオブジェクト：属性を持つ

- ウィンドウ・オブジェクトの**属性（内部状態）**

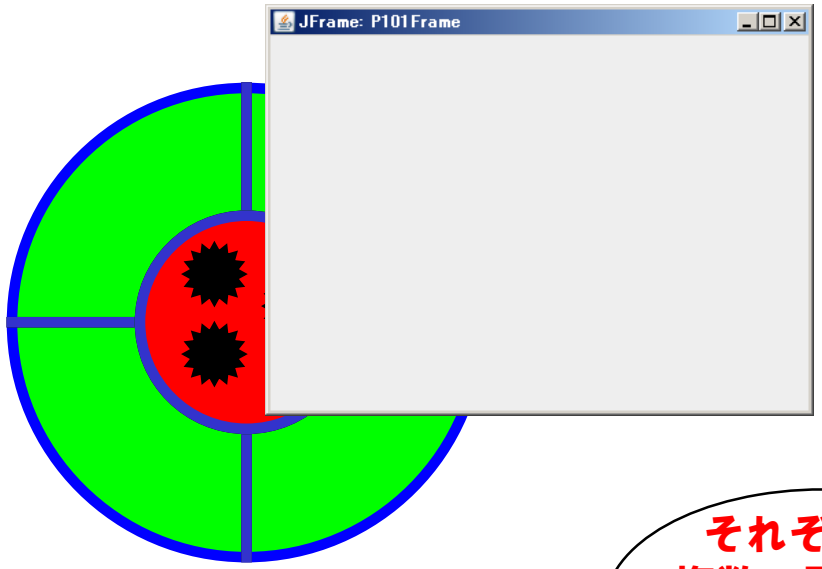
ウィンドウ・オブジェクトは，それぞれ
大きさや位置，タイトルバーの文字列
といった**固有の属性（データ；内部状態）**
を持っている





メソッド
（操作）

属性



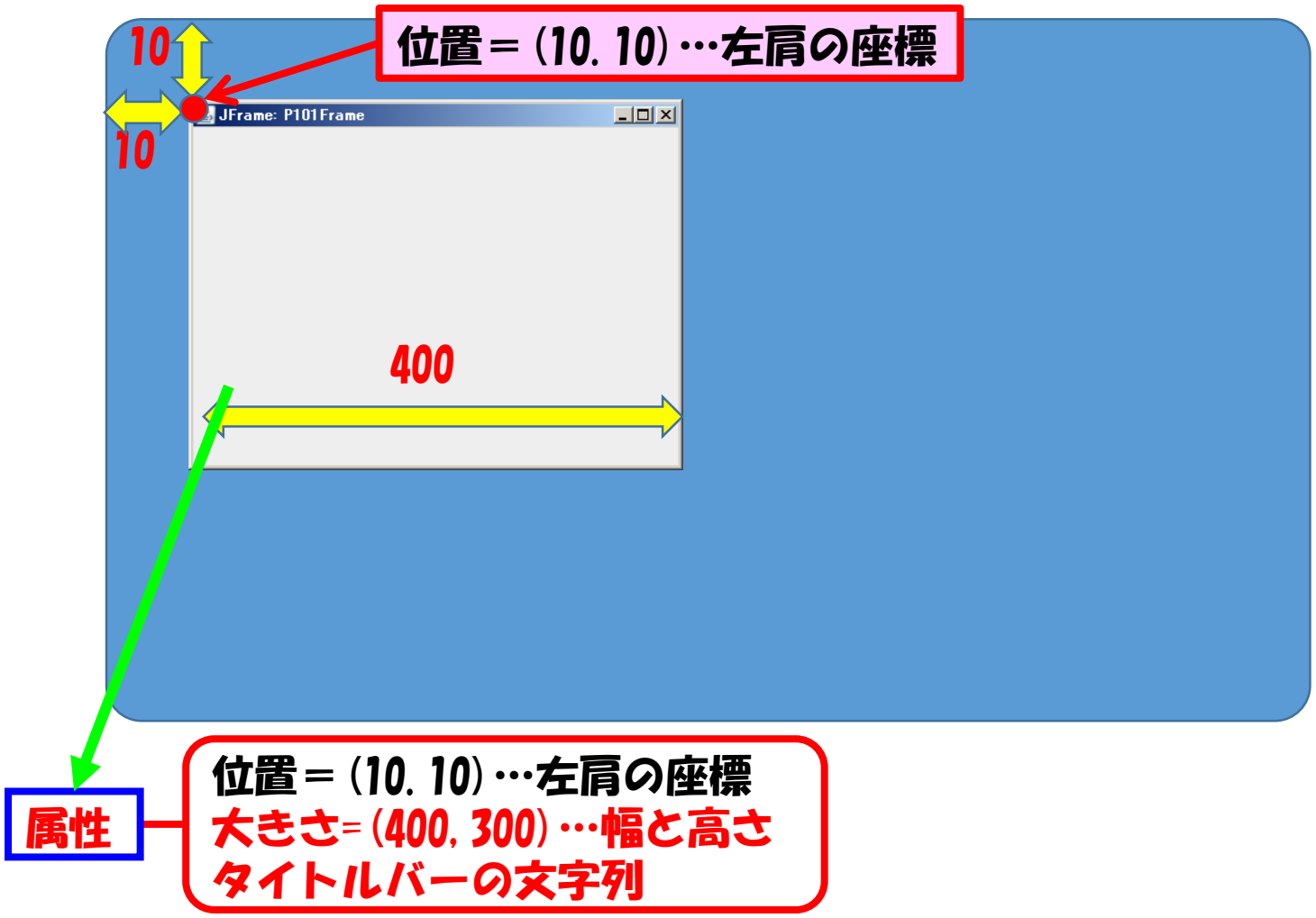
位置＝(10,10)…左肩の座標
大きさ=(400,300)…幅と高さ
タイトルバーの文字列

それぞれ，
複数の属性と，
対応する値を
持ちます

iijima@ae.keio.ac.jp

ウィンドウもオブジェクト：属性を持つ

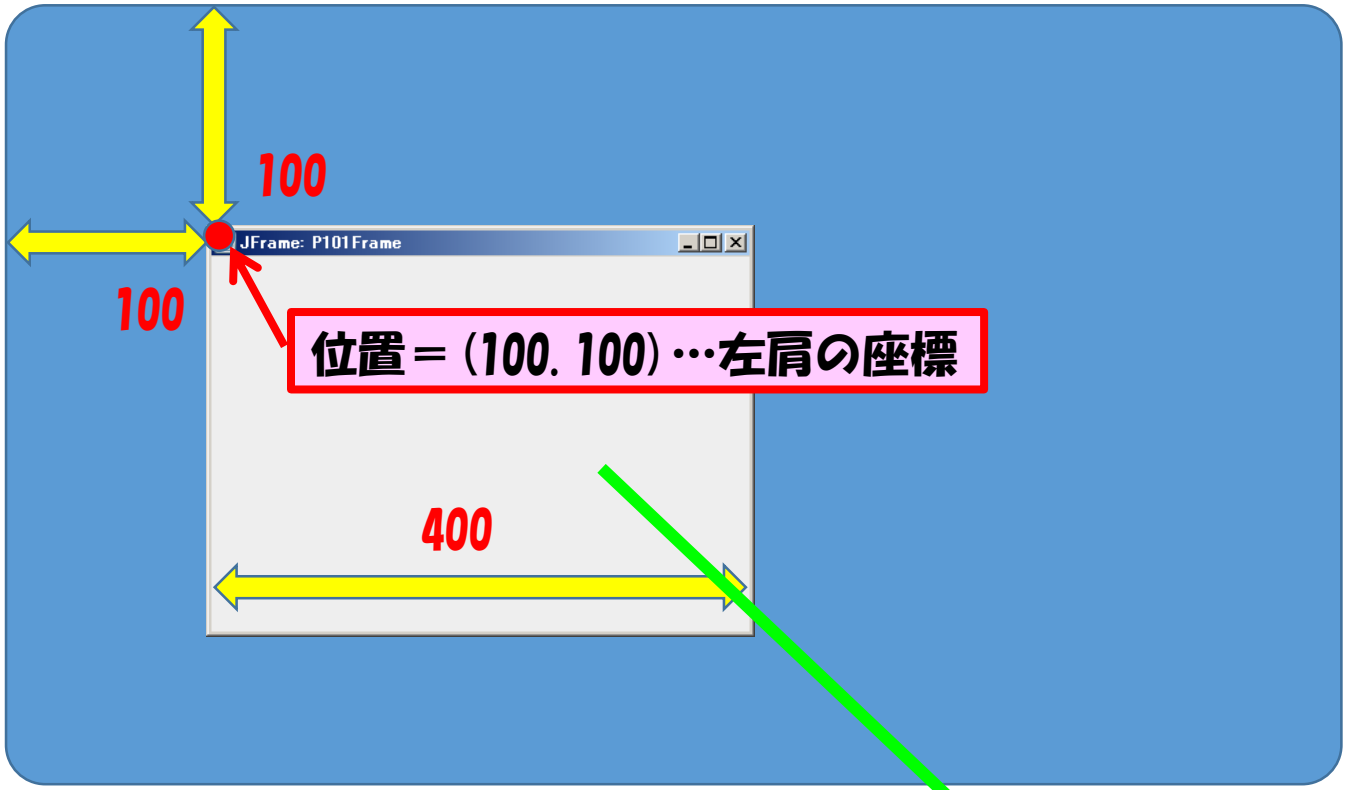
- あるウィンドウの位置は，座標(10,10)



iijima@ae.keio.ac.jp

ウィンドウもオブジェクト：属性を持つ

- 別のウィンドウの位置は，座標(100,100)



属性

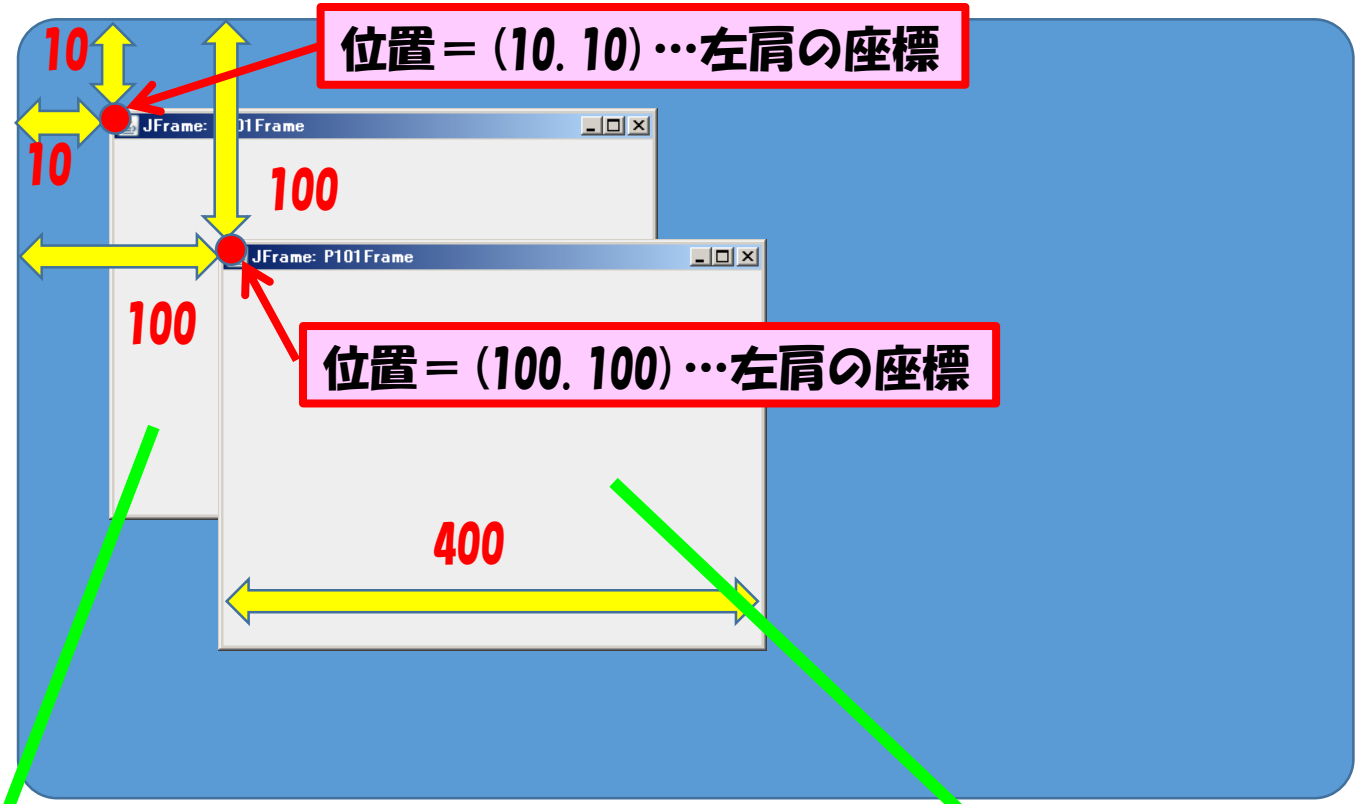
位置＝(100, 100)…左肩の座標
大きさ=(400, 300)…幅と高さ
タイトルバーの文字列

別の
ウィンドウの
位置は，必ず
しも同じでは
ありません

iijima@ae.keio.ac.jp

ウィンドウもオブジェクト：属性を持つ

- 同じクラス定義から作られたオブジェクトでも属性値が異なる



属性
位置＝(10, 10)…左肩の座標
大きさ=(400, 300)…幅と高さ
タイトルバーの文字列

属性
位置＝(100, 100)…左肩の座標
大きさ=(400, 300)…幅と高さ
タイトルバーの文字列

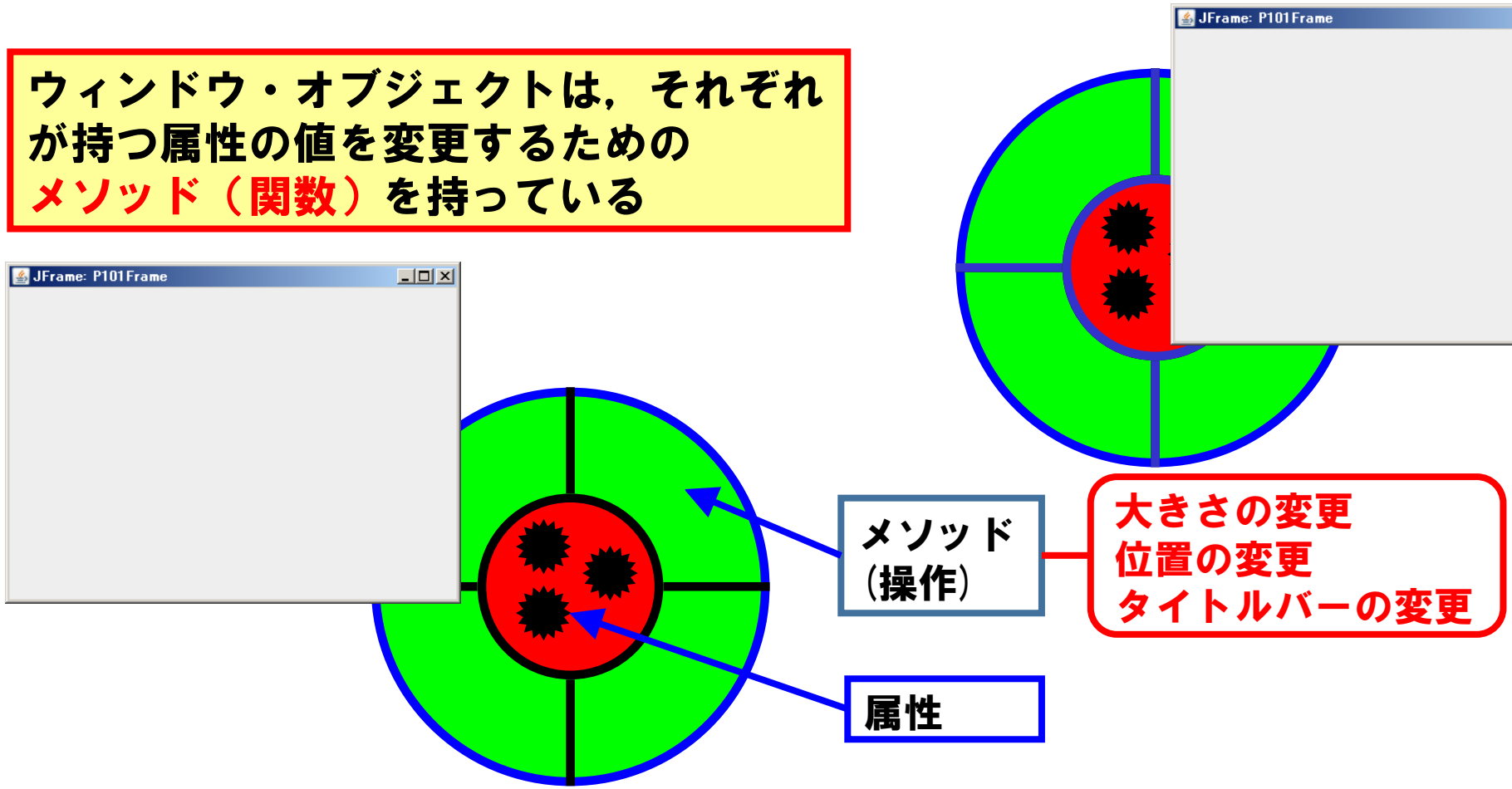
同じ属性を
持っていても
属性値は
異なっていて
構いません

iijima@ae.keio.ac.jp

ウィンドウもオブジェクト：メソッドを持つ

- 属性（内部状態）はメソッドを介してアクセスされる

ウィンドウ・オブジェクトは，それぞれが持つ属性の値を変更するための**メソッド（関数）**を持っている



iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

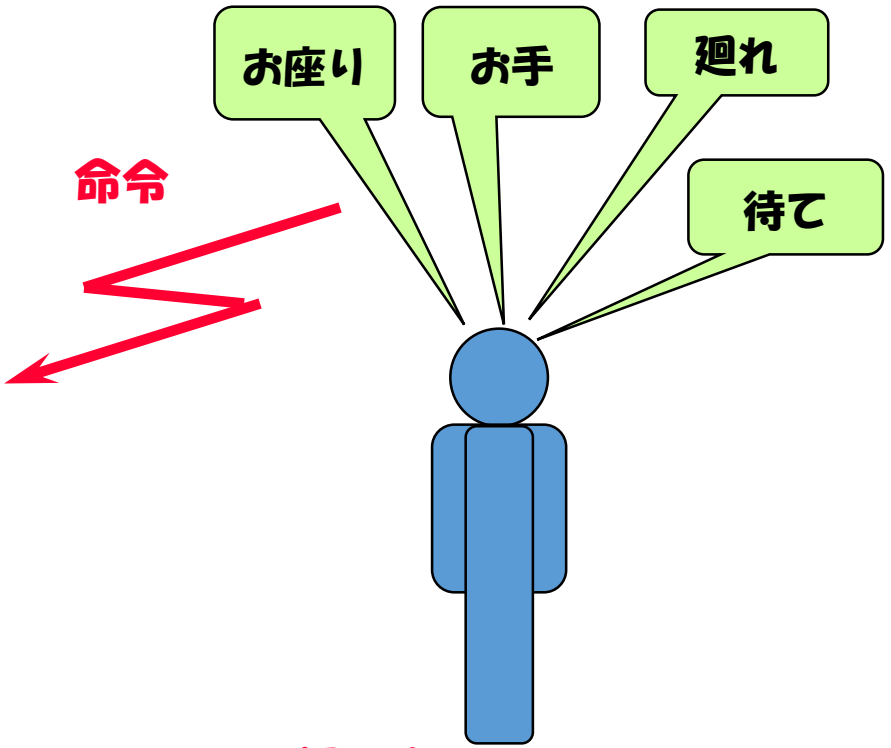
実世界のモデル化

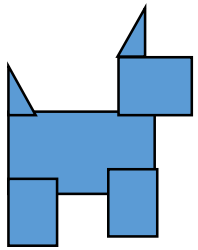
↓

エレベータのモデリング

メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数





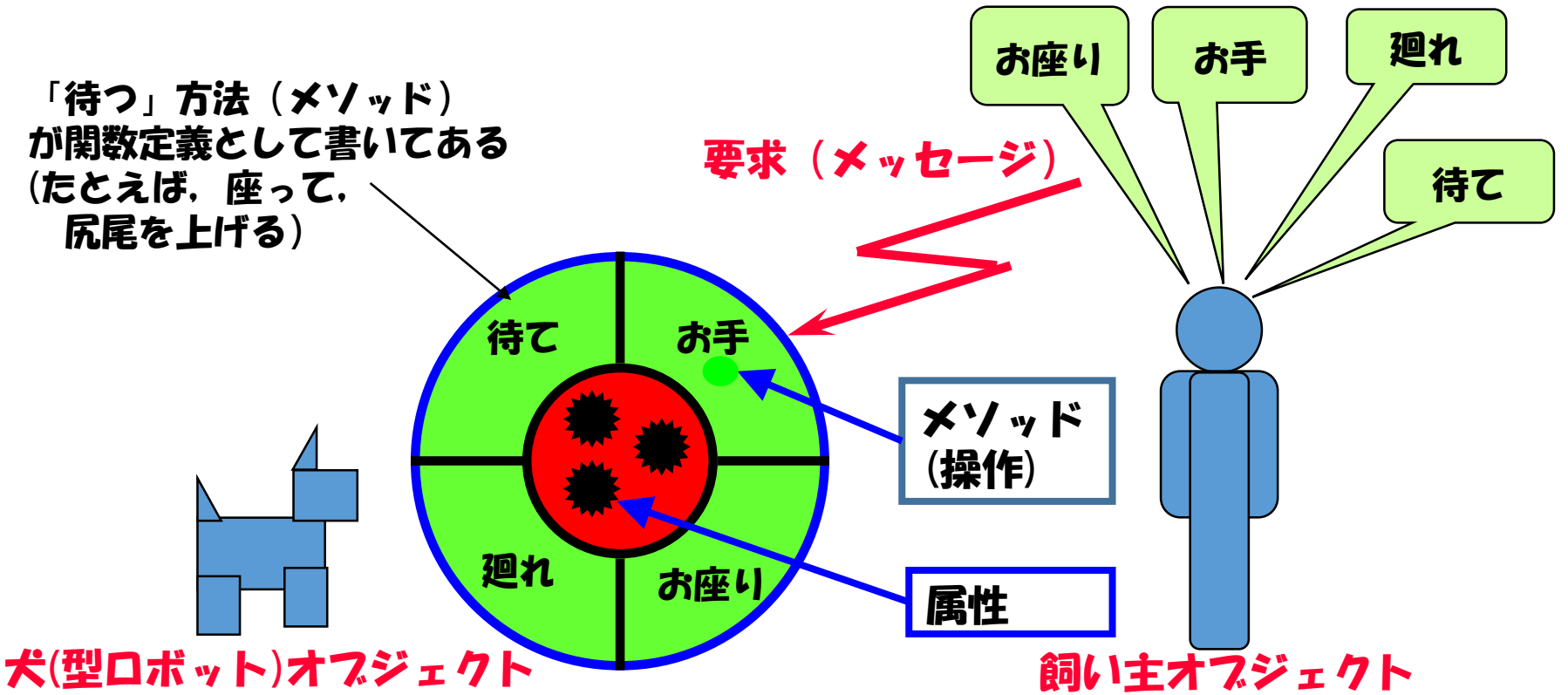
犬(型ロボット)オブジェクト

飼い主オブジェクト

iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

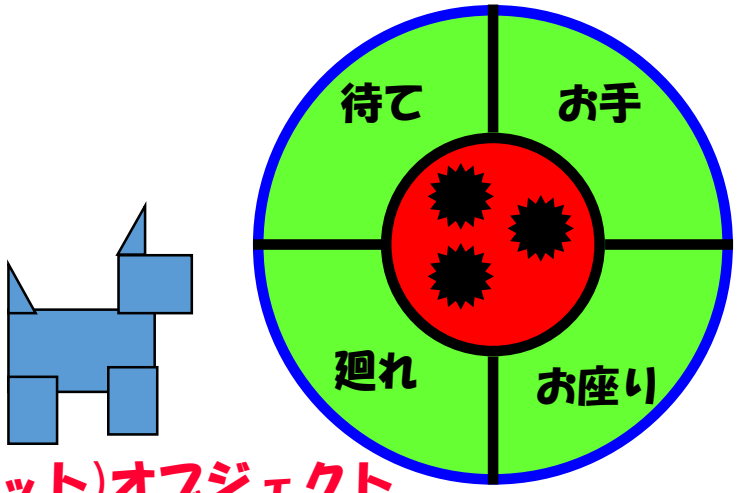
- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



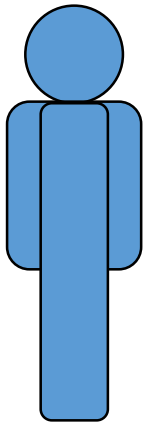
iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



犬(型ロボット)オブジェクト

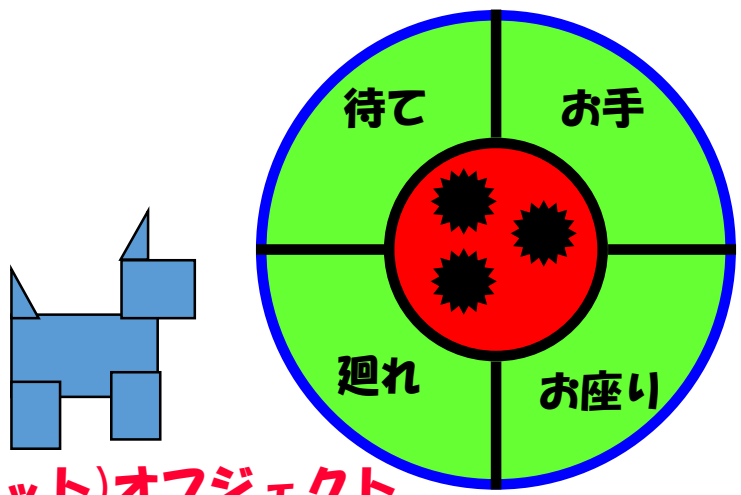


飼い主オブジェクト

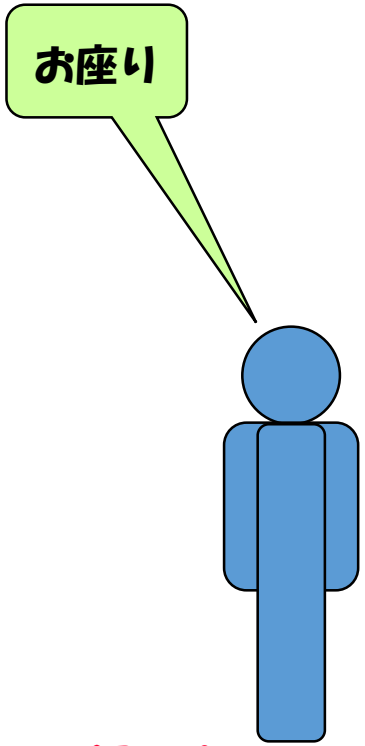
iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



犬(型ロボット)オブジェクト

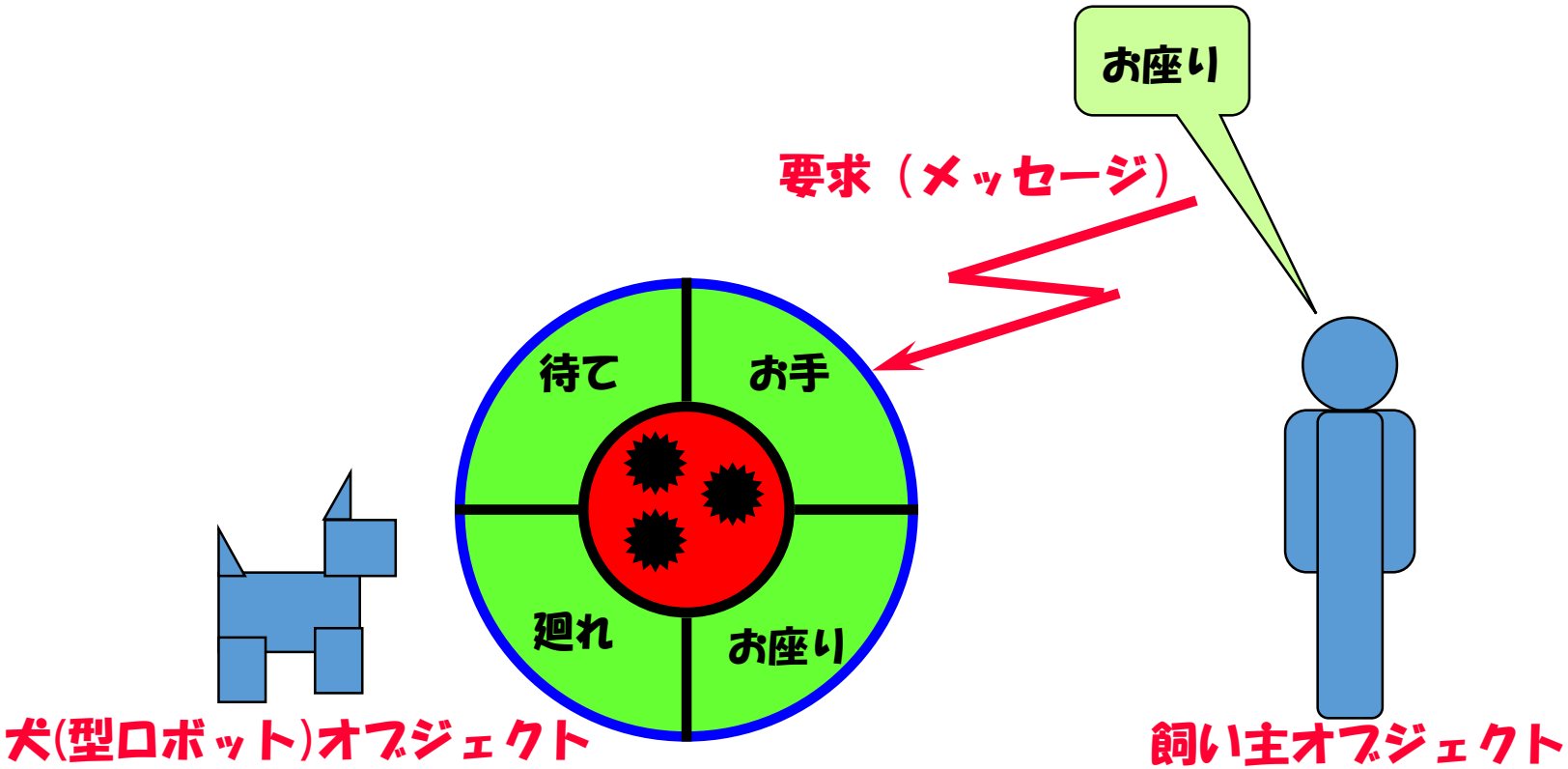


飼い主オブジェクト

iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

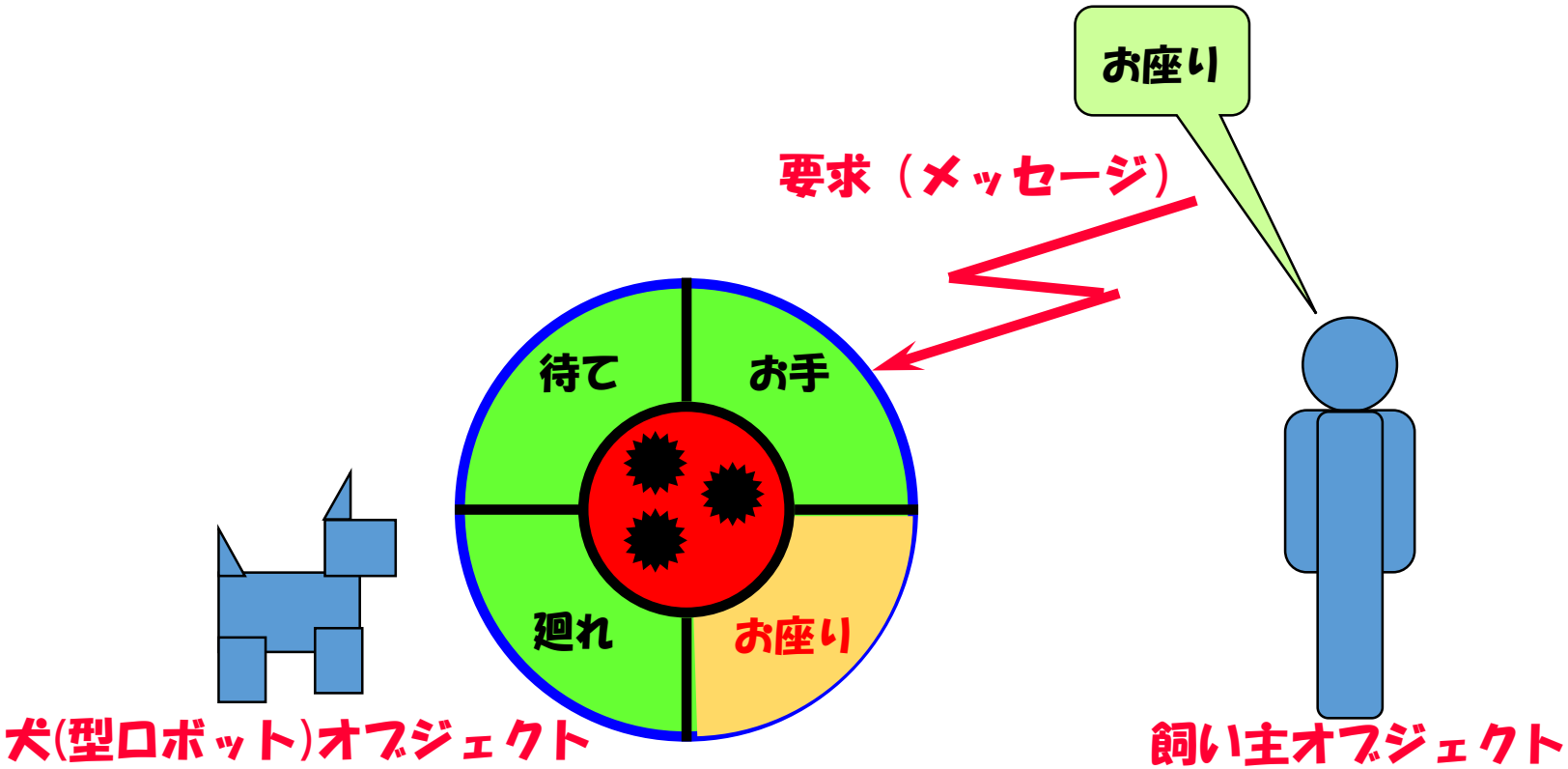
- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

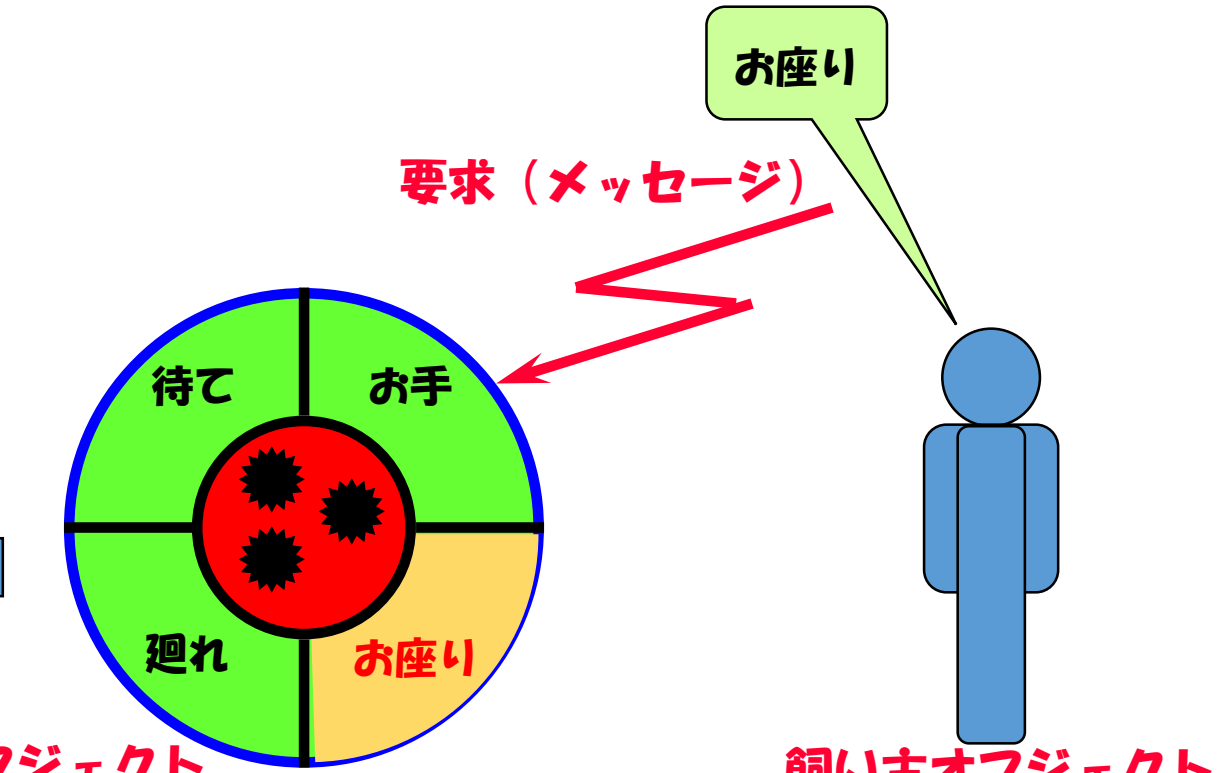
- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



iijima@ae.keio.ac.jp

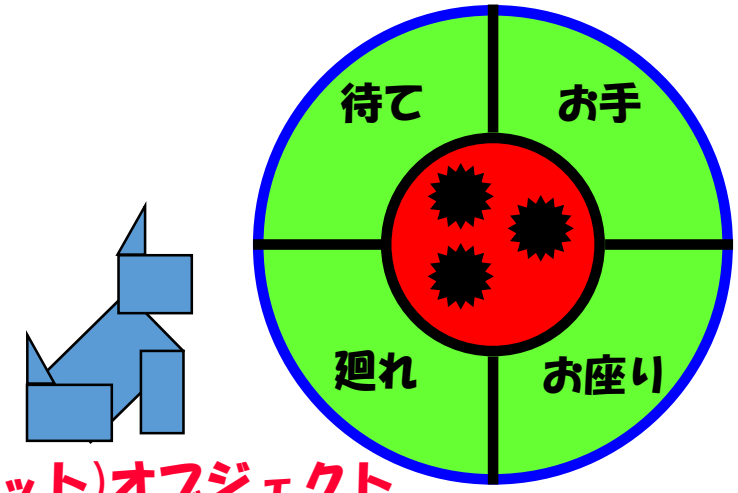
メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数

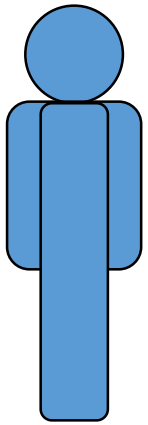


メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



犬(型ロボット)オブジェクト

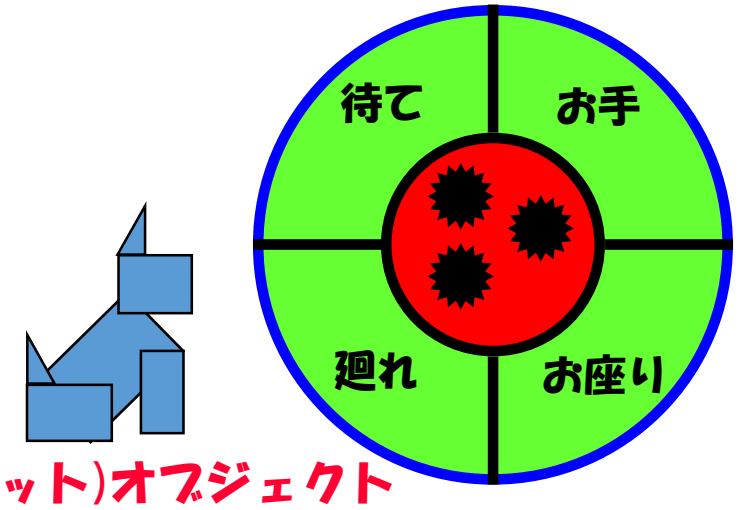


飼い主オブジェクト

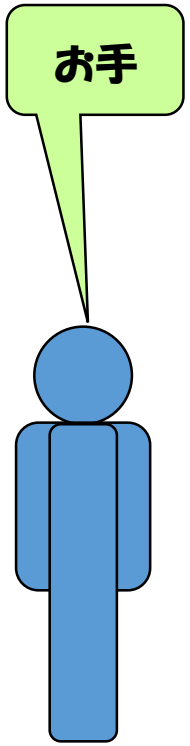
メソッドが
内部状態
（属性）を
変化させ，
モーターを
動かして，
対応する
ポーズを
とらせます

メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



犬(型ロボット)オブジェクト

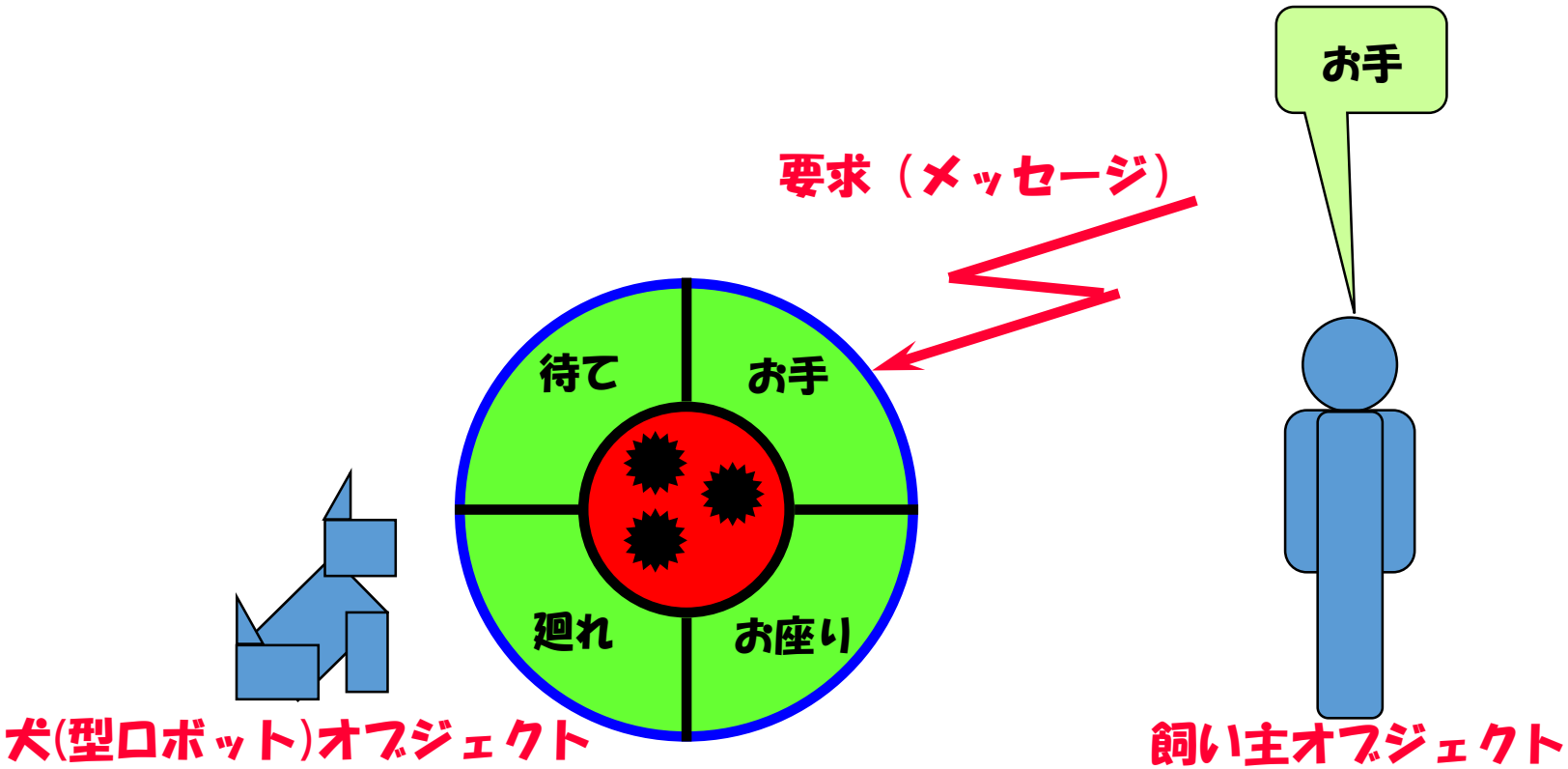


飼い主オブジェクト

iijima@ae.keio.ac.jp

メソッド（振る舞い，メンバ関数）

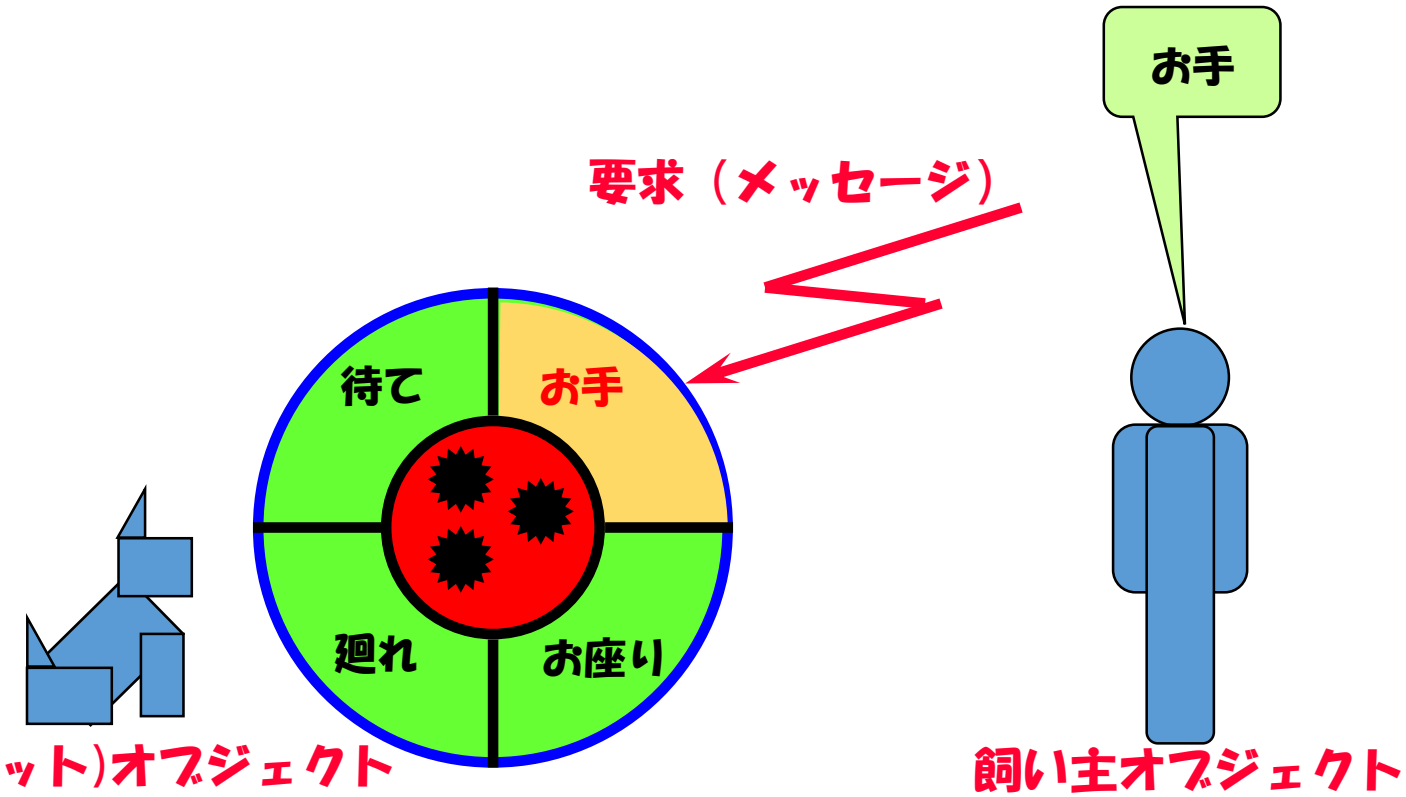
- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



iijima@ae.keio.ac.jp

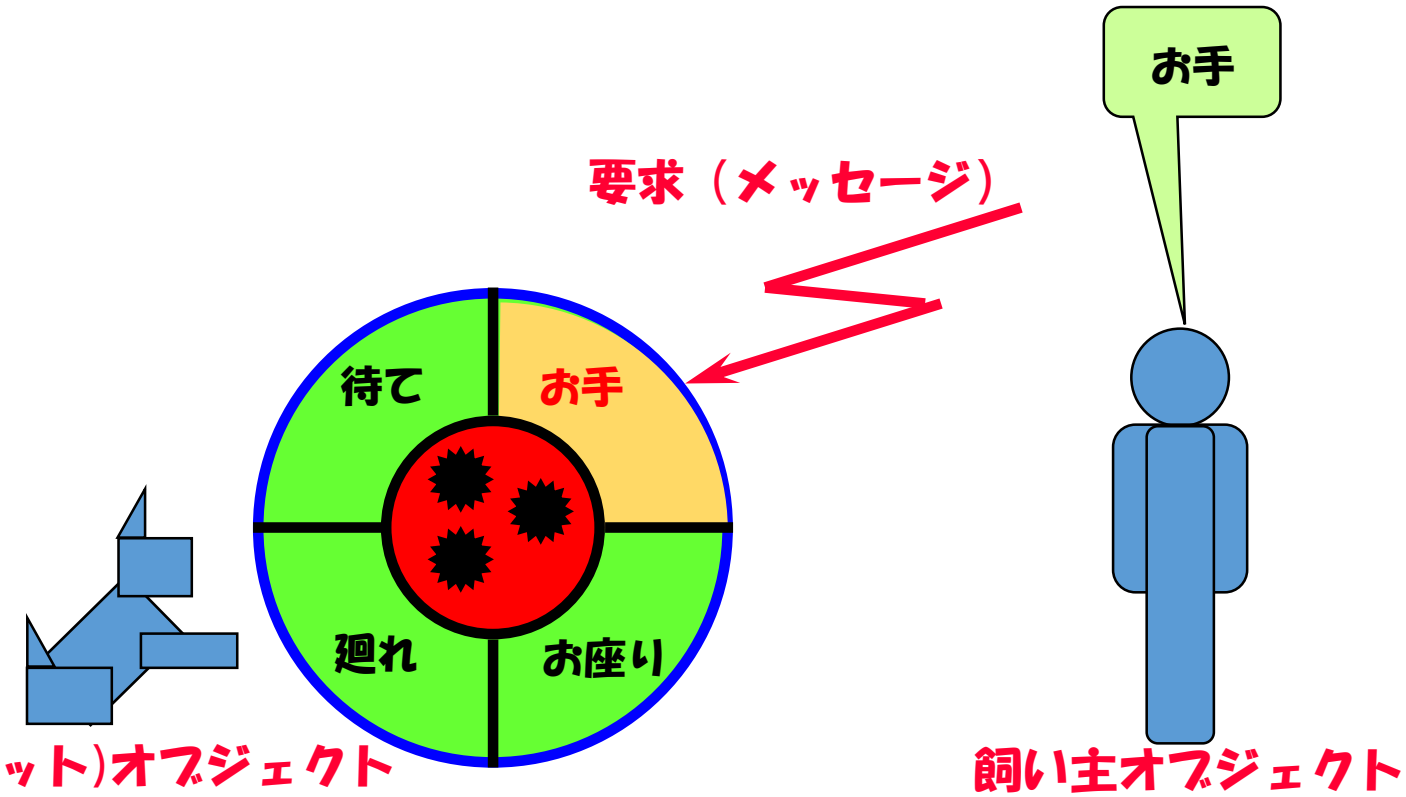
メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



メソッド（振る舞い，メンバ関数）

- メソッド⇒オブジェクトの振る舞い
  - 外部へ作用する機能
  - 内部状態変数を書き換えるデータ操作
- 外部から起動できる個々のオブジェクト固有の関数



話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

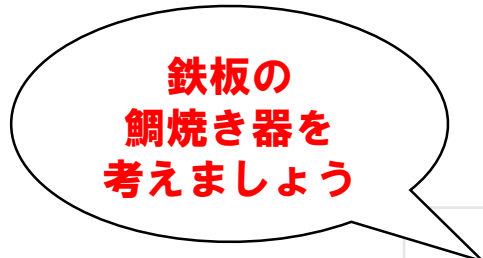
実世界のモデル化

↓

エレベータのモデリング

オブジェクトはクラスから作る

- クラス定義は，「鯛焼き器」の鉄板のようなもの



iijima@ae.keio.ac.jp

オブジェクトはクラスから作る

- クラス定義は，「鯛焼き器」の鉄板のようなもの

クラス定義

属性
- 位置…左肩の座標
- 大きさ…幅と高さ
- タイトルバーの文字列

メソッド（操作）
- 大きさの変更
- 位置の変更
- タイトルバーの変更

型押しすることで同じ構造のたい焼きが焼けます

iijima@ae.keio.ac.jp

オブジェクトはクラスから作る

- クラス定義は，「鯛焼き器」の鉄板のようなもの

クラス定義

属性
- 位置…左肩の座標
- 大きさ…幅と高さ
- タイトルバーの文字列

メソッド（操作）
- 大きさの変更
- 位置の変更
- タイトルバーの変更

オブジェクト（インスタンス）

作られた鯛焼きはオブジェクトに相当します

iijima@ae.keio.ac.jp

オブジェクトはクラスから作る

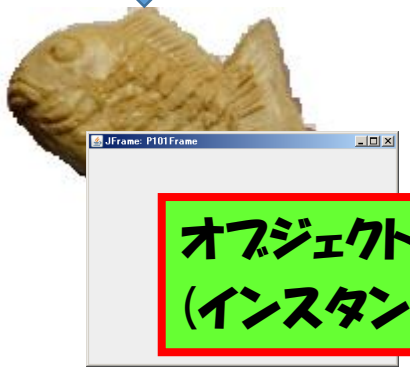
- クラス定義は，「鯛焼き器」の鉄板のようなもの

クラス定義

属性
- 位置…左肩の座標
- 大きさ…幅と高さ
- タイトルバーの文字列

メソッド（操作）
- 大きさの変更
- 位置の変更
- タイトルバーの変更

オブジェクト（インスタンス）



オブジェクト（インスタンス）

いくつでも作れます

iijima@ae.keio.ac.jp

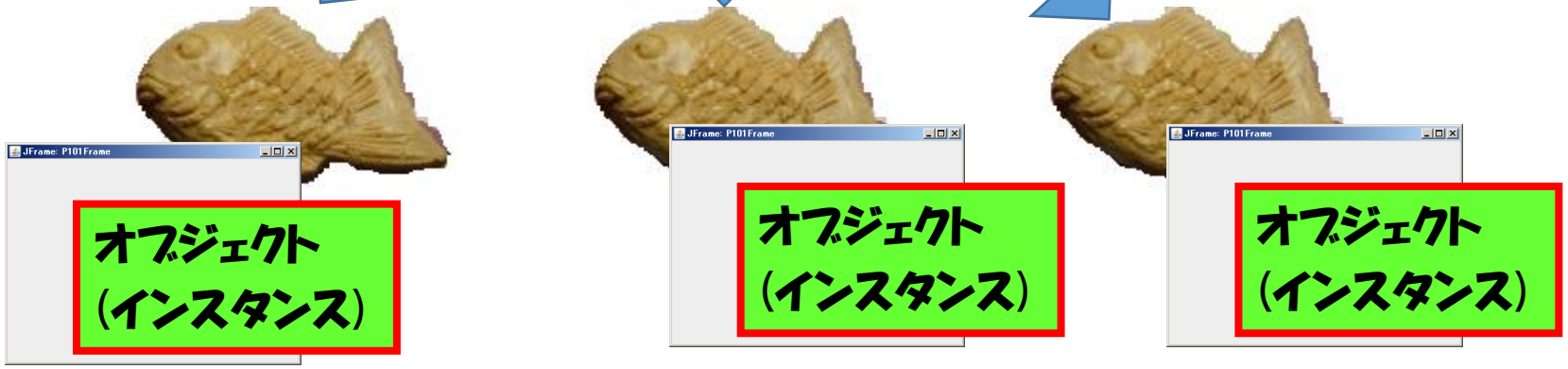
オブジェクトはクラスから作る

- クラス定義は，「鯛焼き器」の鉄板のようなもの

クラス定義

属性
- 位置…左肩の座標
- 大きさ…幅と高さ
- タイトルバーの文字列

メソッド（操作）
- 大きさの変更
- 位置の変更
- タイトルバーの変更



オブジェクト（インスタンス）

オブジェクト（インスタンス）

オブジェクト（インスタンス）

こしあんや
つぶあん
クリーム鯛焼
きも作れます

iijima@ae.keio.ac.jp

オブジェクトはクラスから作る

- クラス定義は，「鯛焼き器」の鉄板のようなもの

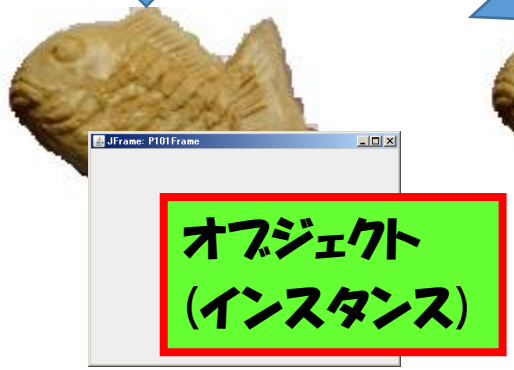
クラスに対して，
クラスから生成した
オブジェクトのことを
インスタンス
（実例/実体）
と呼ぶ

クラス定義

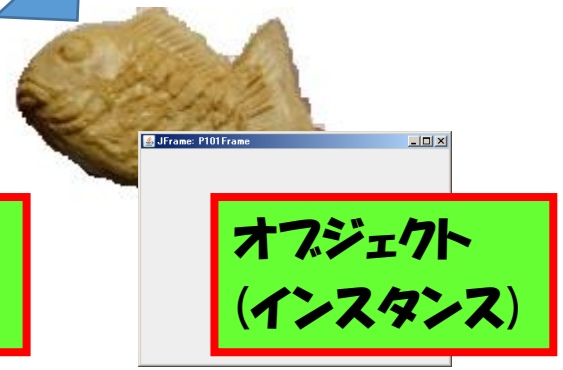
属性
位置…左肩の座標
大きさ…幅と高さ
タイトルバーの文字列

メソッド
(操作)
大きさの変更
位置の変更
タイトルバーの変更

オブジェクト
(インスタンス)



オブジェクト
(インスタンス)



オブジェクト
(インスタンス)

クラスから
生成した
オブジェクト
のことを
そのクラスの
インスタンス
と
言います

iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

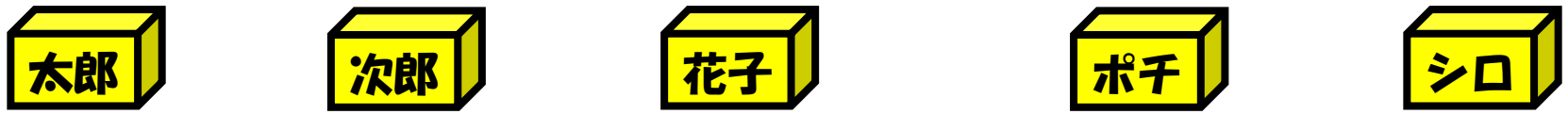
↓

エレベータのモデリング

クラスとインスタンス(1/2)

- オブジェクトは個体なので，固有名詞が付きます

具象

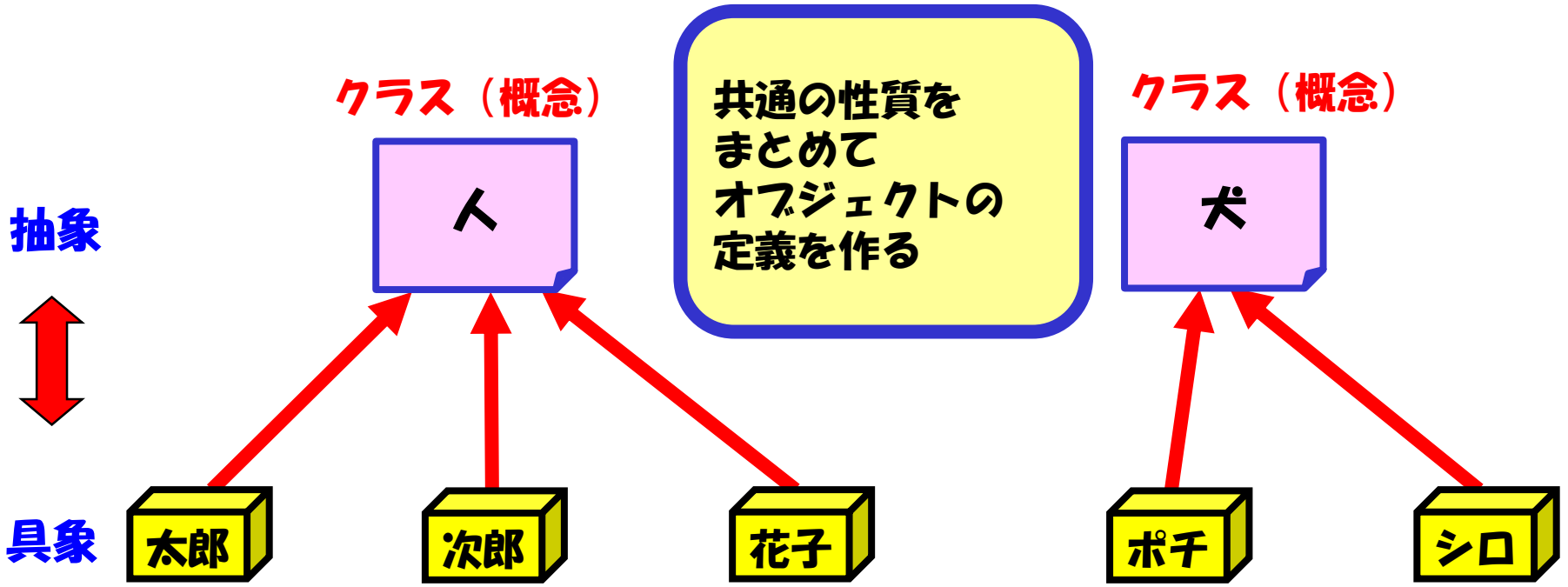


オブジェクト

iijima@ae.keio.ac.jp

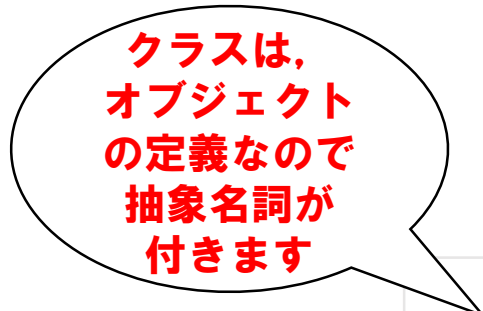
クラスとインスタンス(1/2)

・クラスは，オブジェクトの定義なので，抽象名詞が付きます．



オブジェクト

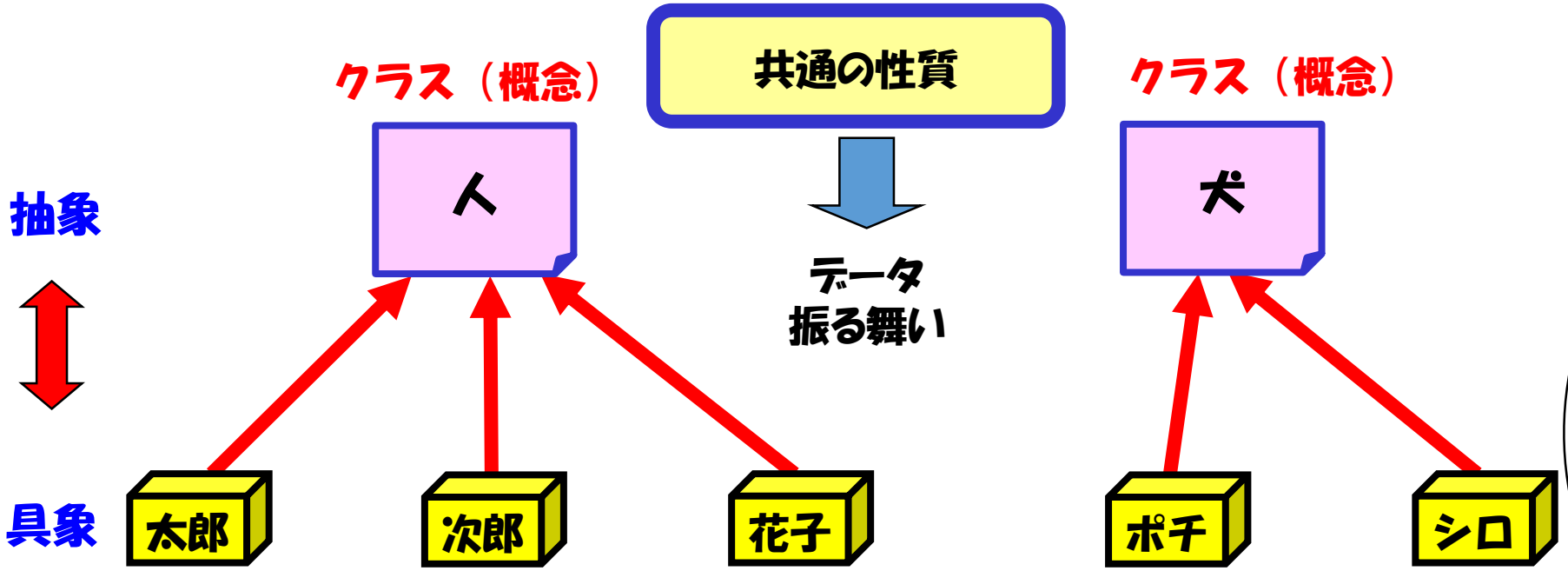
クラスのインスタンス(個々の実体)



iijima@ae.keio.ac.jp

クラスとインスタンス(1/2)

・オブジェクトに共通な性質を，データと振る舞いの観点から定義します



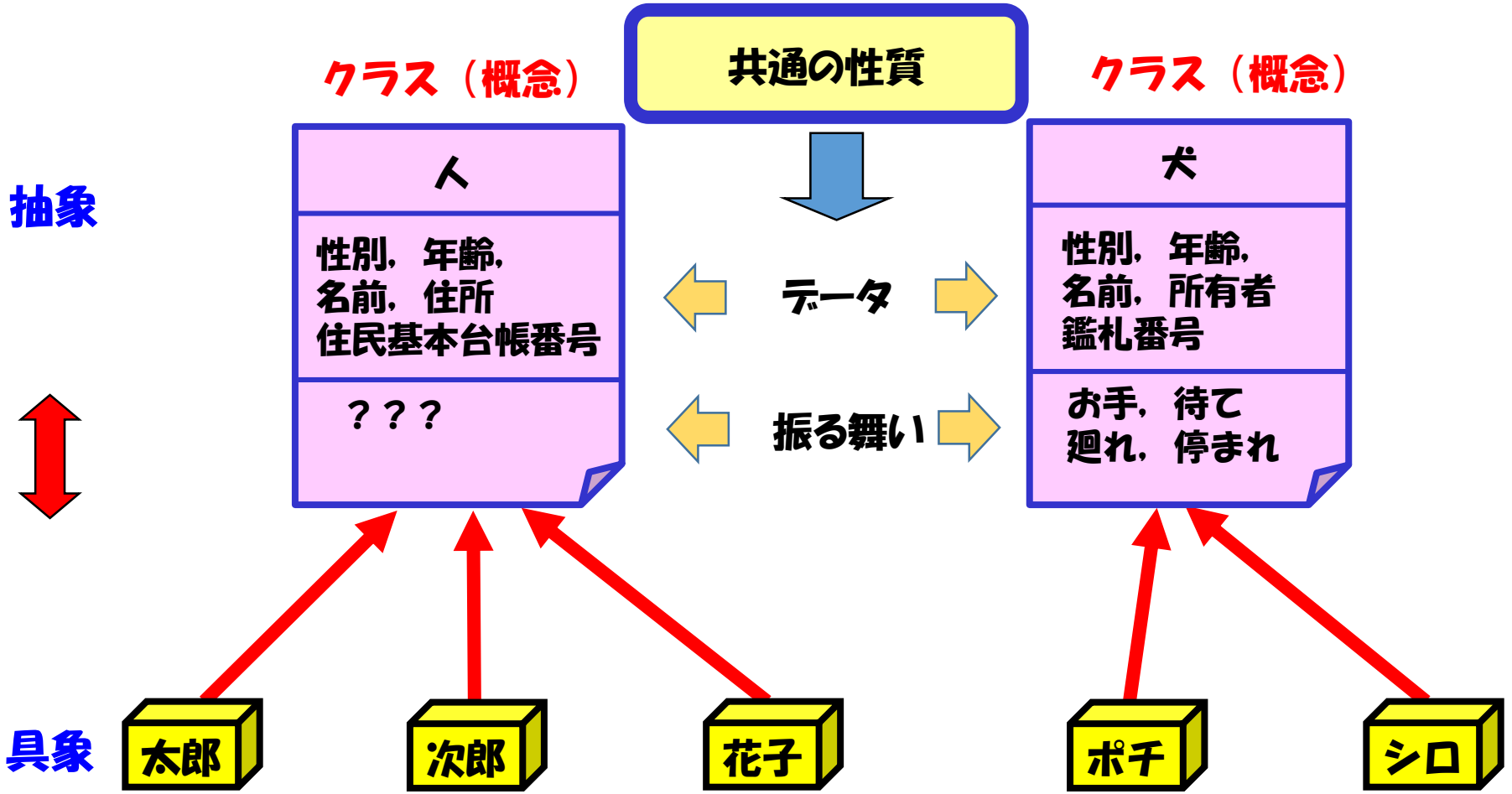
クラスでは，
オブジェクト
に
共通な性質を
データと
振る舞い
の二つの観点
から
定義します

オブジェクト

クラスのインスタンス(個々の実体)

クラスとインスタンス(1/2)

- データは属性，振る舞いはメソッドとして定義しましょう．



クラスとインスタンス(2/2)

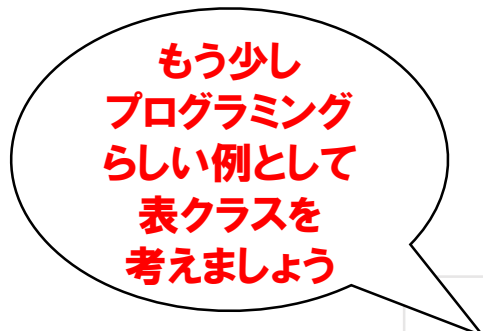
- もう少し，プログラミングらしい例を考えましょう

クラス＝テンプレート（定規）→枠組，仕様

宣言とか定義

表

属性のデータ型
＋
操作の関数定義

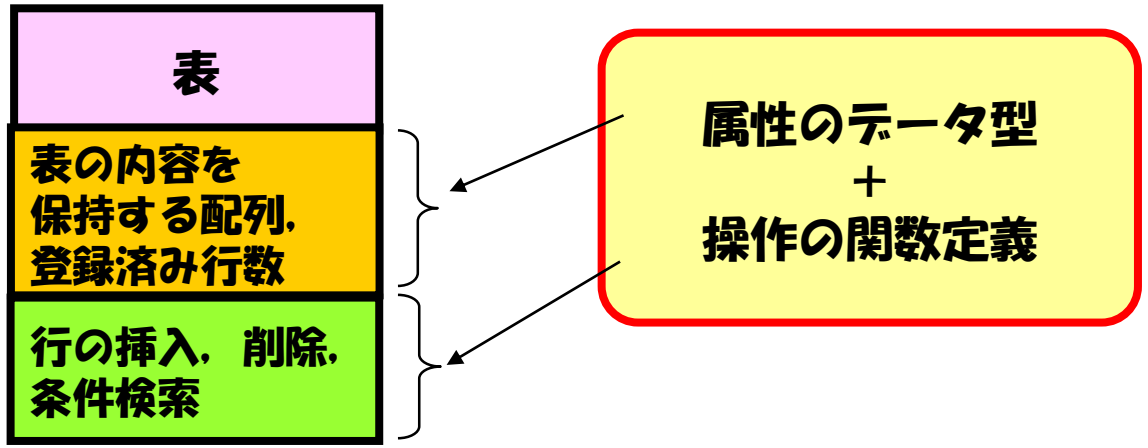


iijima@ae.keio.ac.jp

クラスとインスタンス(2/2)

クラス＝テンプレート（定規）→枠組，仕様

宣言とか定義

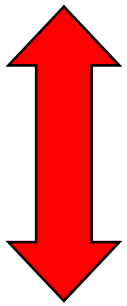


iijima@ae.keio.ac.jp

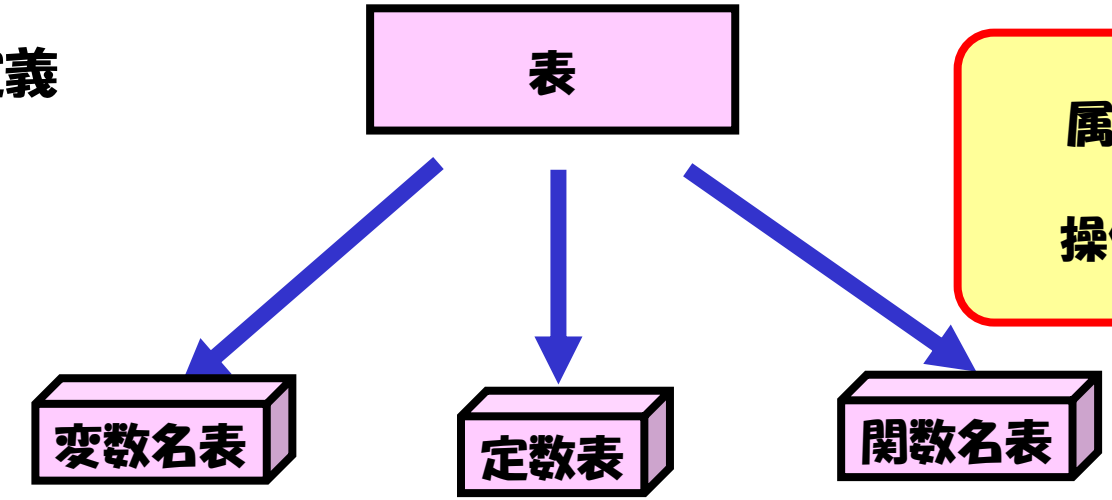
クラスとインスタンス(2/2)

クラス＝テンプレート(定規)→枠組，仕様

宣言とか定義



個々の値



属性のデータ型
＋
操作の関数定義

(例)
コンパイラ

インスタンス＝実体→実際のデータが入っている

同じカテゴリに属す(同じ性質を持つ)オブジェクトは，
クラスから生成される(同じクラスに属する)

一つ表クラスを
作ると
いろいろな表に
利用できます

iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

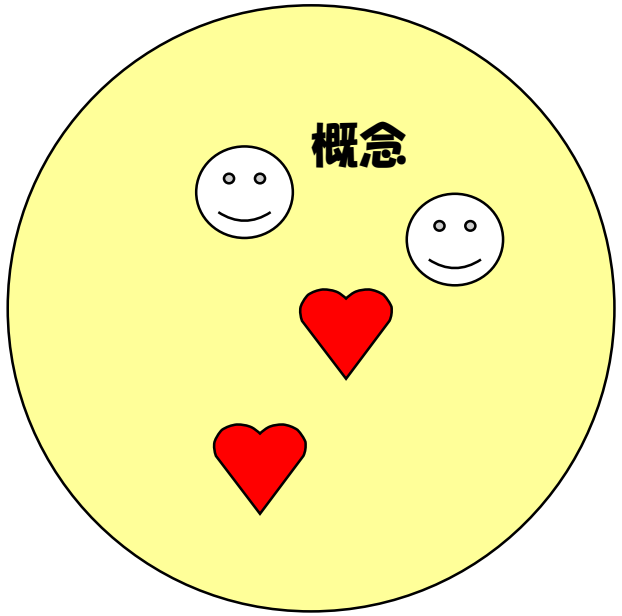
↓

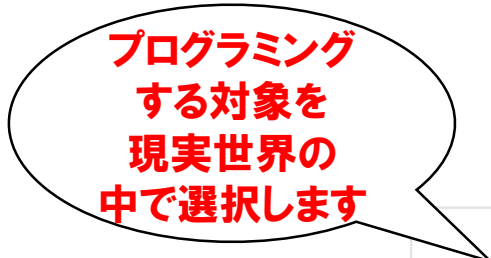
エレベータのモデリング

iijima@ae.keio.ac.jp

概念→クラス→オブジェクト

- プログラミングする現実世界の概念（モノ，対象物）を選択します

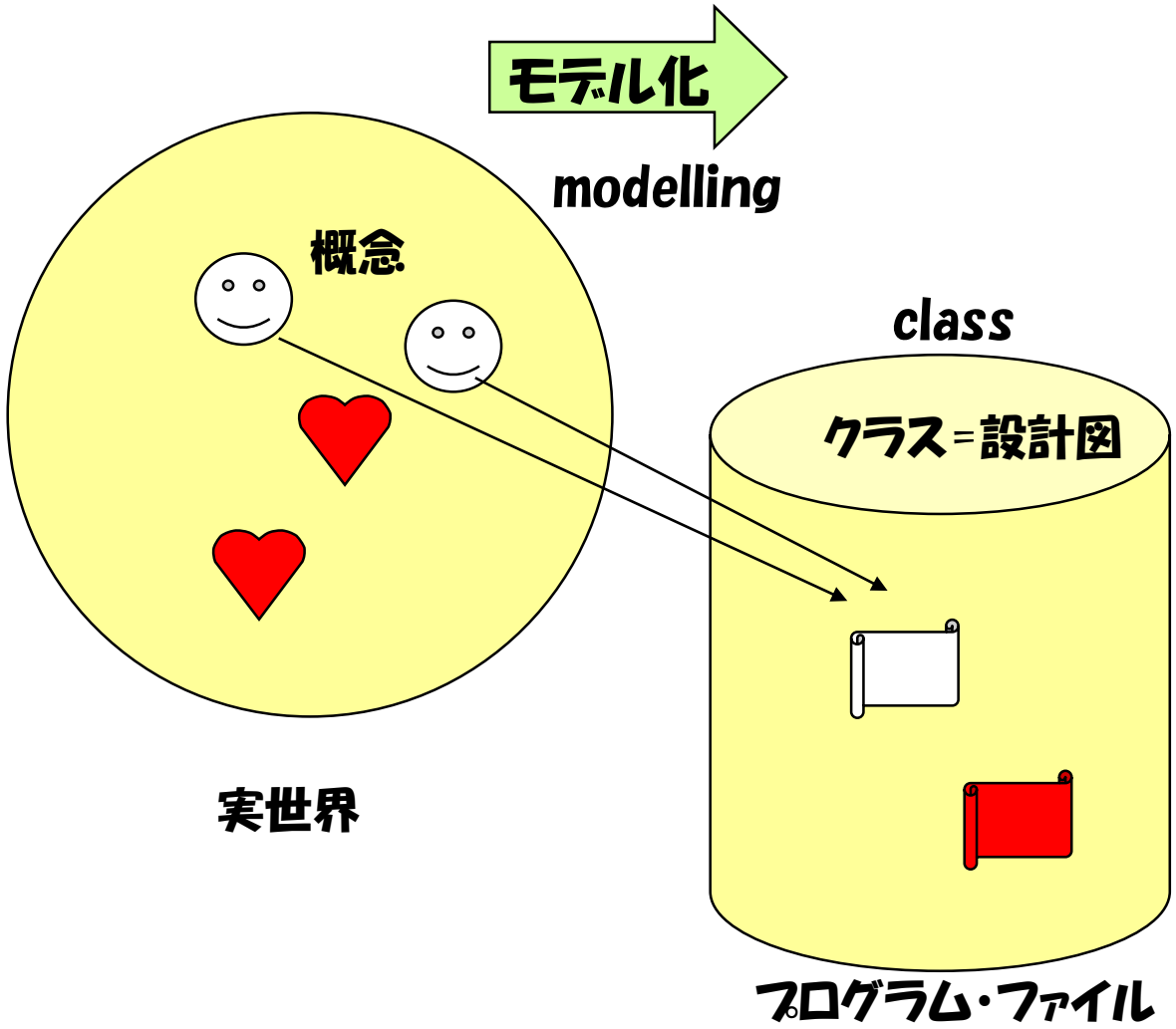




iijima@ae.keio.ac.jp

概念→クラス→オブジェクト

- [モデル化] 次に，現実世界の概念をモデル化してクラスを定義します

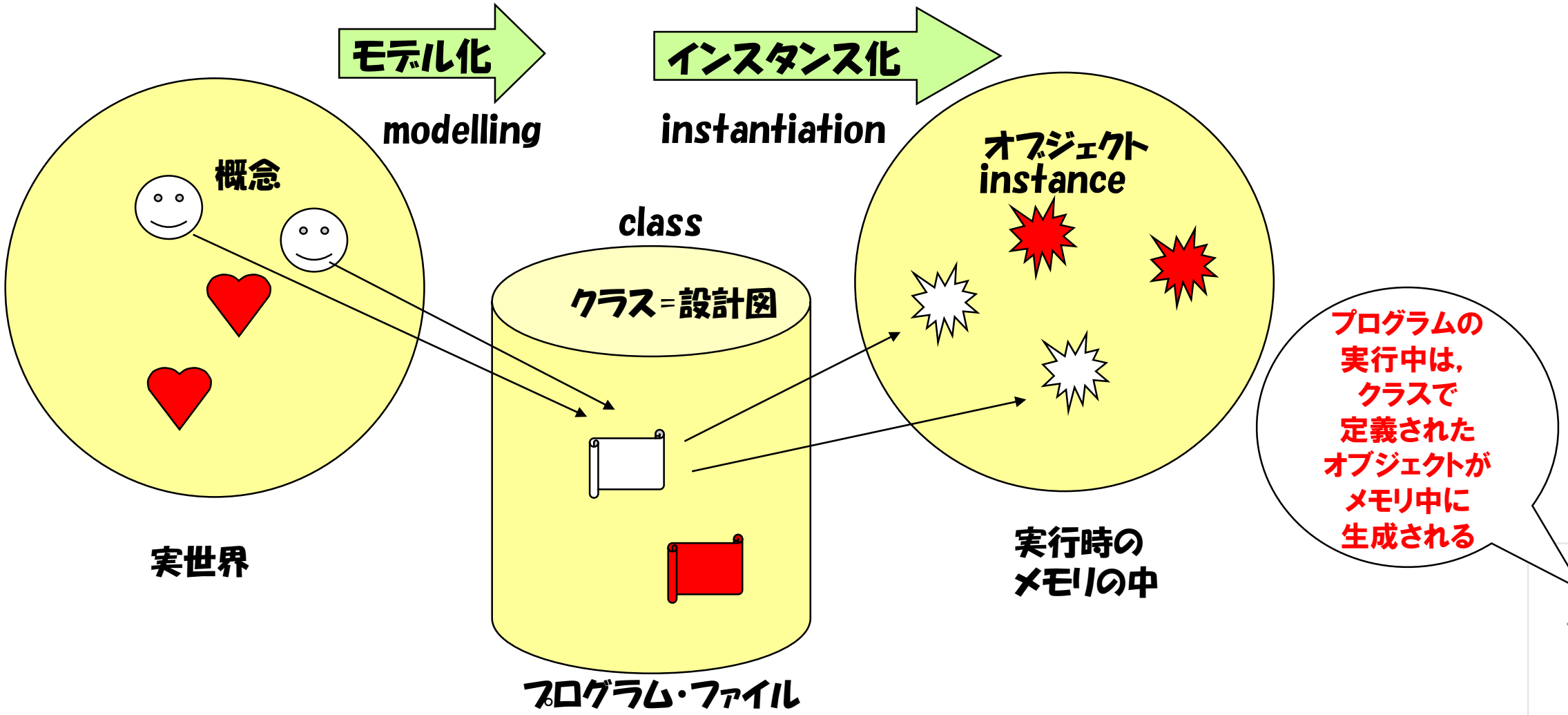


プログラム
として
ファイルに書き
込むのは，
オブジェクトの
設計図である
クラスです

iijima@ae.keio.ac.jp

概念→クラス→オブジェクト

- ［インスタンス化］プログラムの実行時には，クラスからオブジェクトが作られます



iijima@ae.keio.ac.jp

話の流れ

オブジェクト指向とは…モジュール単位

↓

オブジェクトはスイカ…内部構造は属性とメソッド

↓

ウィンドウもオブジェクト

↓

オブジェクトはメッセージに反応する

↓

クラスは鯛焼き器…オブジェクトはクラスから作る

↓

クラスとインスタンス

↓

実世界のモデル化

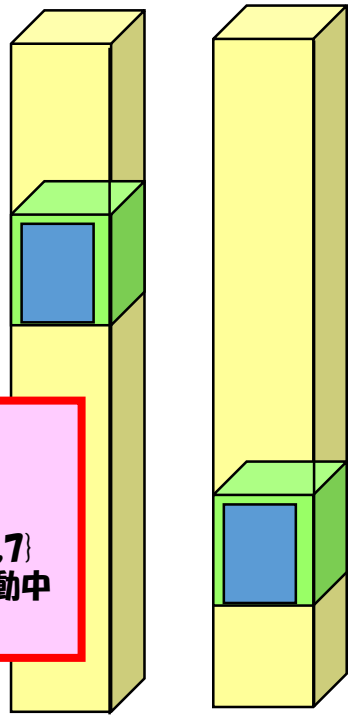
↓

エレベータのモデリング

iijima@ae.keio.ac.jp

例: エレベータのモデル化 (“what”)
～オブジェクトとクラス～

- 個々のエレベータの客室（キャビン，籠）はオブジェクト
- 共通仕様をモデル化したクラスから生成したインスタンス
- 個々のオブジェクト（インスタンス）は，それぞれに**固有の内部状態**を持つ



1号機

属性:番号＝1
属性:階＝3
属性:目標階リスト＝{6,7}
属性:移動方向＝上移動中
属性:ドア＝閉

2号機

属性:番号＝2
属性:階＝1
属性:目標階リスト＝{}
属性:移動方向＝停止中
属性:ドア＝開

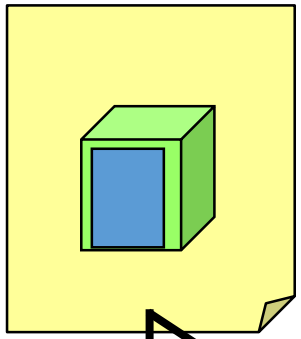
生成時に決まる属性
途中で変化する状態

エレベータの共通仕様⇒クラス

属性:番号
属性:階
属性:目標階リスト
属性:移動方向(上，下，停止)
属性:ドア(開，閉)

1号機と2号機では，異なる属性値を持ちます

例: エレベータのモデル化 (“what”)
～オブジェクトとクラス～

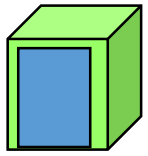


クラス

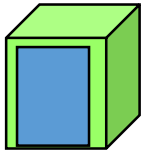
	名前	データ型
属性	番号	正の整数
	階	1～7の整数
	目標階リスト	1～7の整数のリスト
	移動方向	上移動中，下移動中，停止中のいずれか
	ドア	開か閉のいずれか
メソッド	X階へ移動	入力:X，出力:なし
	移動方向取得	入力:なし，出力:{上，下，停止}
	目標階追加	入力:1～7の整数
	…	…

このクラスのインスタンス⇒オブジェクト

	名前	値を格納するメモリ領域
属性	番号	
	階	
	目標階リスト	
	移動方向	
	ドア	



1号機



2号機

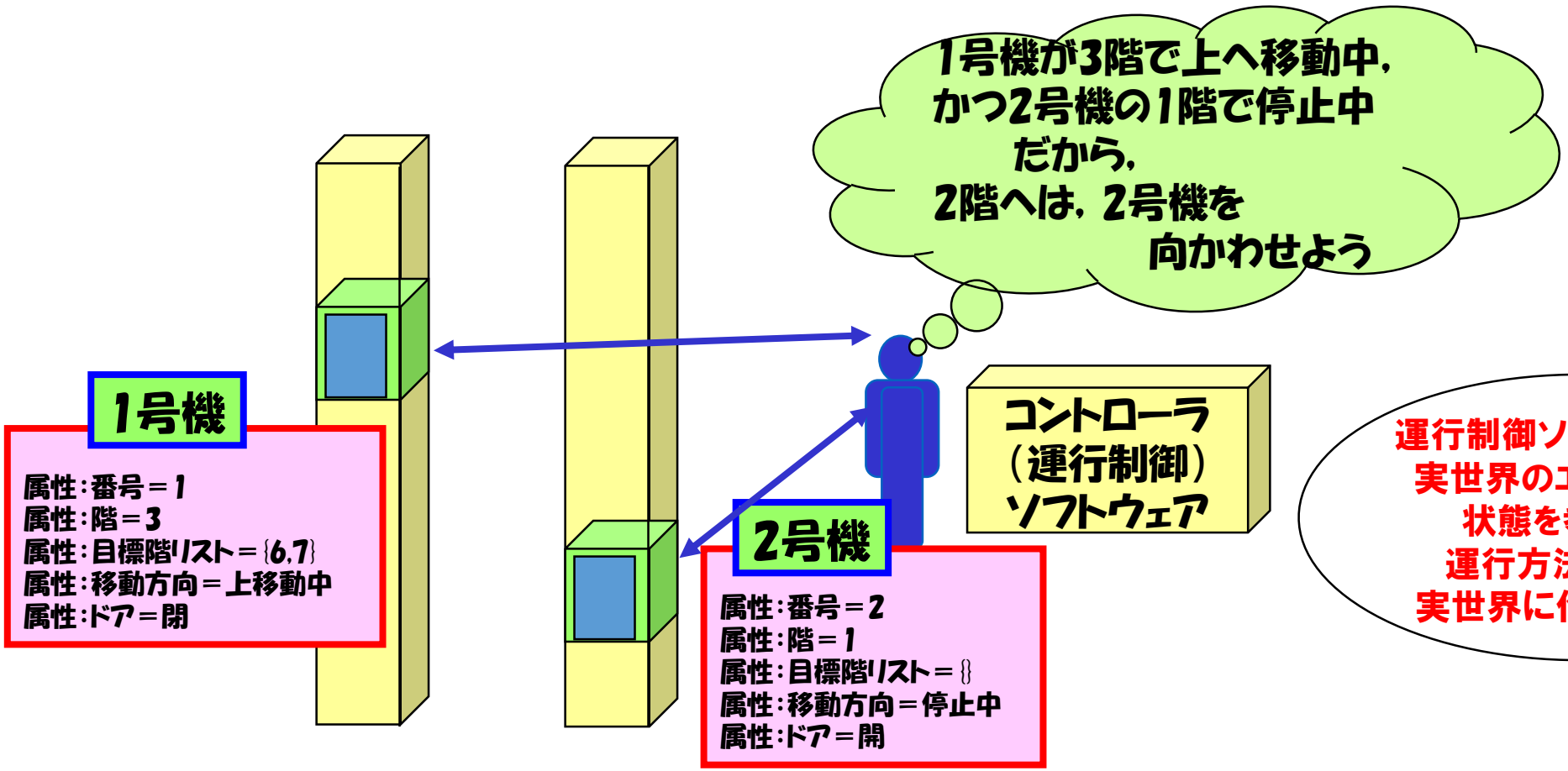
	名前	値を格納するメモリ領域
属性	番号	
	階	
	目標階リスト	
	移動方向	
	ドア	

属性値を
格納するための
メモリ領域は
インスタンス毎に
必要です

iijima@ae.keio.ac.jp

例: エレベータのモデル化 (“what”)
～オブジェクトとクラス～

- コントローラのプログラム
  ⇒シミュレーションの要素が含まれている

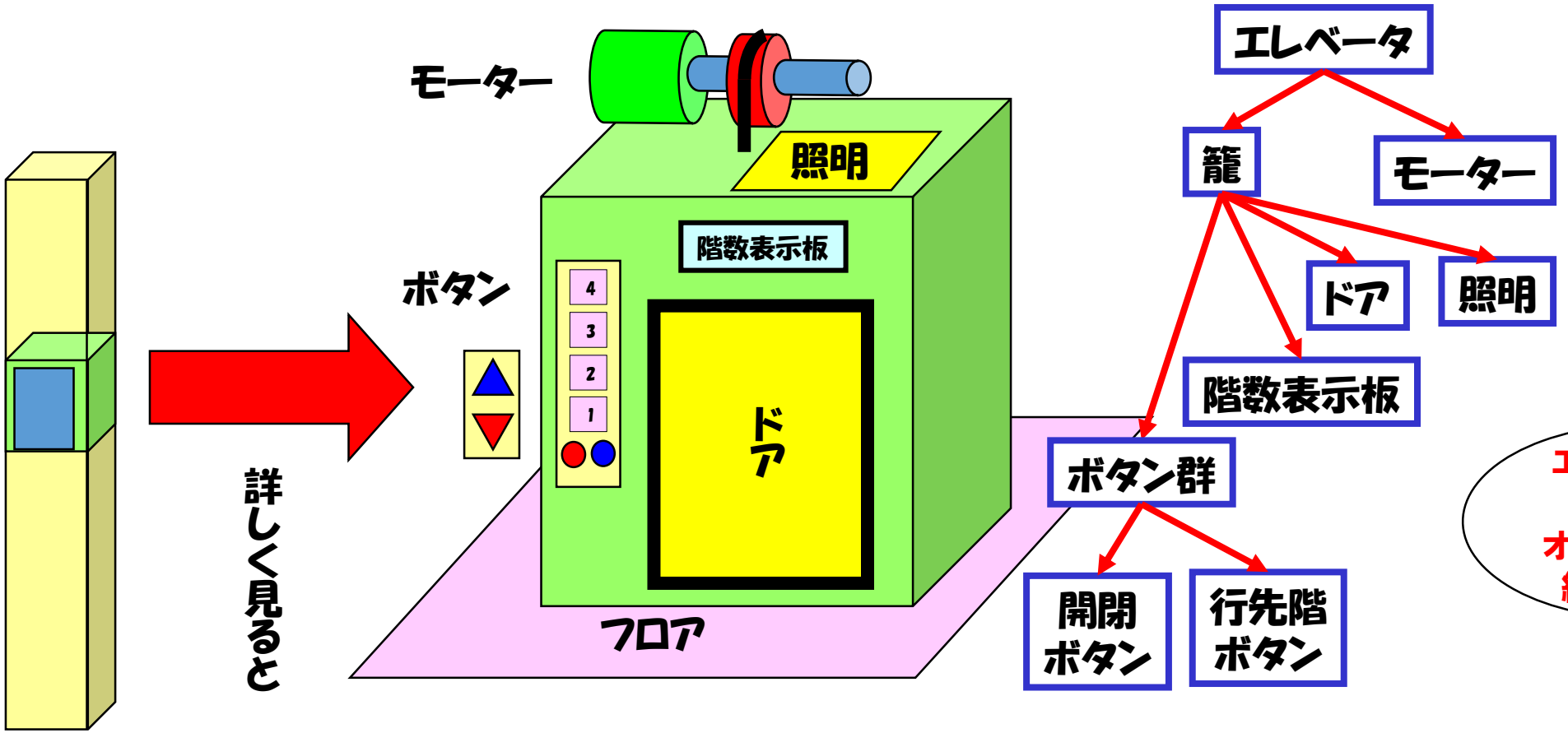


運行制御ソフトウェアは，
実世界のエレベータの
状態を参照して
運行方法を決め，
実世界に作用します．

iijima@ae.keio.ac.jp

例:エレベータのモデル化⇒分析を続けて
～最初は粗くモデル化して，少しずつ詳細化していく～

- 幾らでも詳細化できてしまうが，
  必要最小限＋α（将来を見越して…）にとどめる

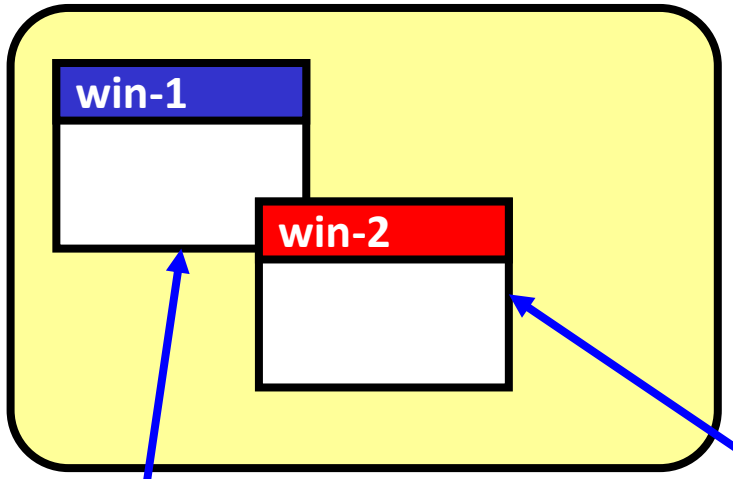


エレベータは
たくさんの
オブジェクトの
組合せです

iijima@ae.keio.ac.jp

例:ウィンドウ・システムのモデル化
～オブジェクトとクラス～

- 画面に表示されている個々のウィンドウはオブジェクト
- 共通仕様をモデル化したクラスから生成したインスタンス
- 個々のオブジェクト（インスタンス）は，それぞれに固有の状態を持つ



ウィンドウの
共通仕様⇒クラス

生成時に
決まる属性

途中で変化
する状態

属性:番号
属性:位置(x座標,y座標)
属性:大きさ(縦，横)
属性:タイトル・バーの色
属性:タイトル・バーの文字列
属性:アイコン化状態

属性:番号＝1
属性:位置＝（10,10）
属性:大きさ＝（60,100）
属性:タイトル・バーの色＝青
属性:タイトル・バーの文字列＝"win-1"
属性:アイコン化の状態＝展開

属性:番号＝2
属性:位置＝（100,100）
属性:大きさ＝（60,100）
属性:タイトル・バーの色＝赤
属性:タイトル・バーの文字列＝"win-2"
属性:アイコン化の状態＝展開

ウィンドウは，
エレベータと
似ています．

iijima@ae.keio.ac.jp