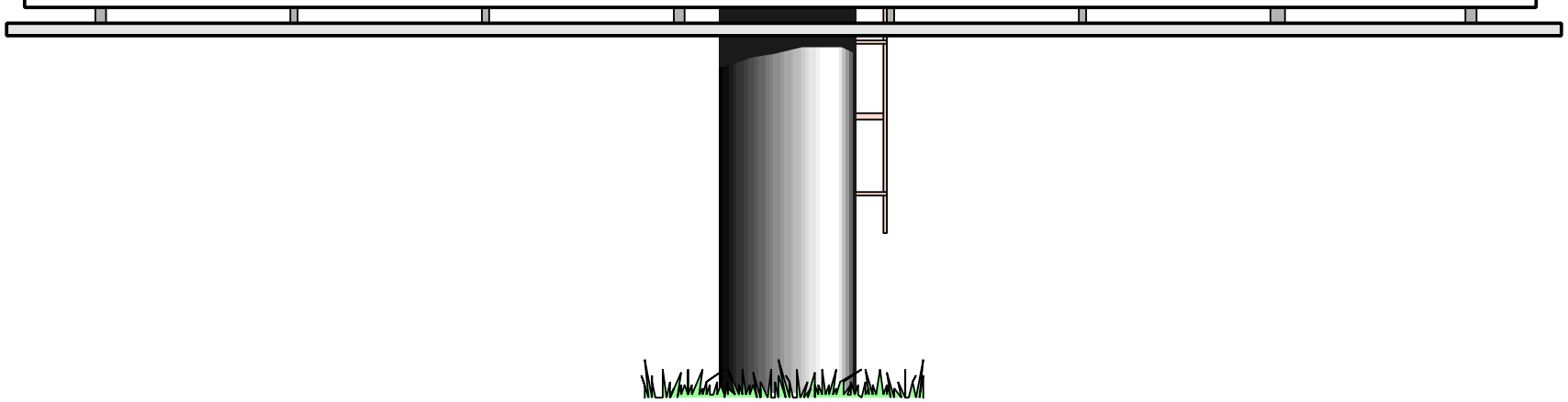
第7章

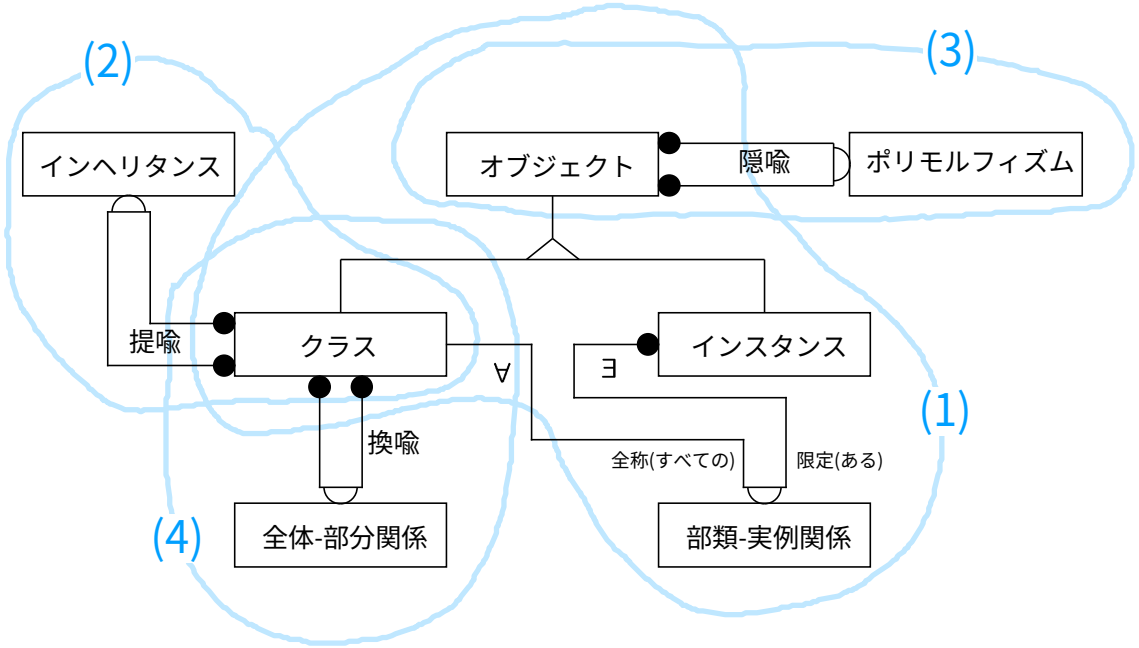
OMTの三つのモデルのメタモデル



メタレベルのオブジェクト指向

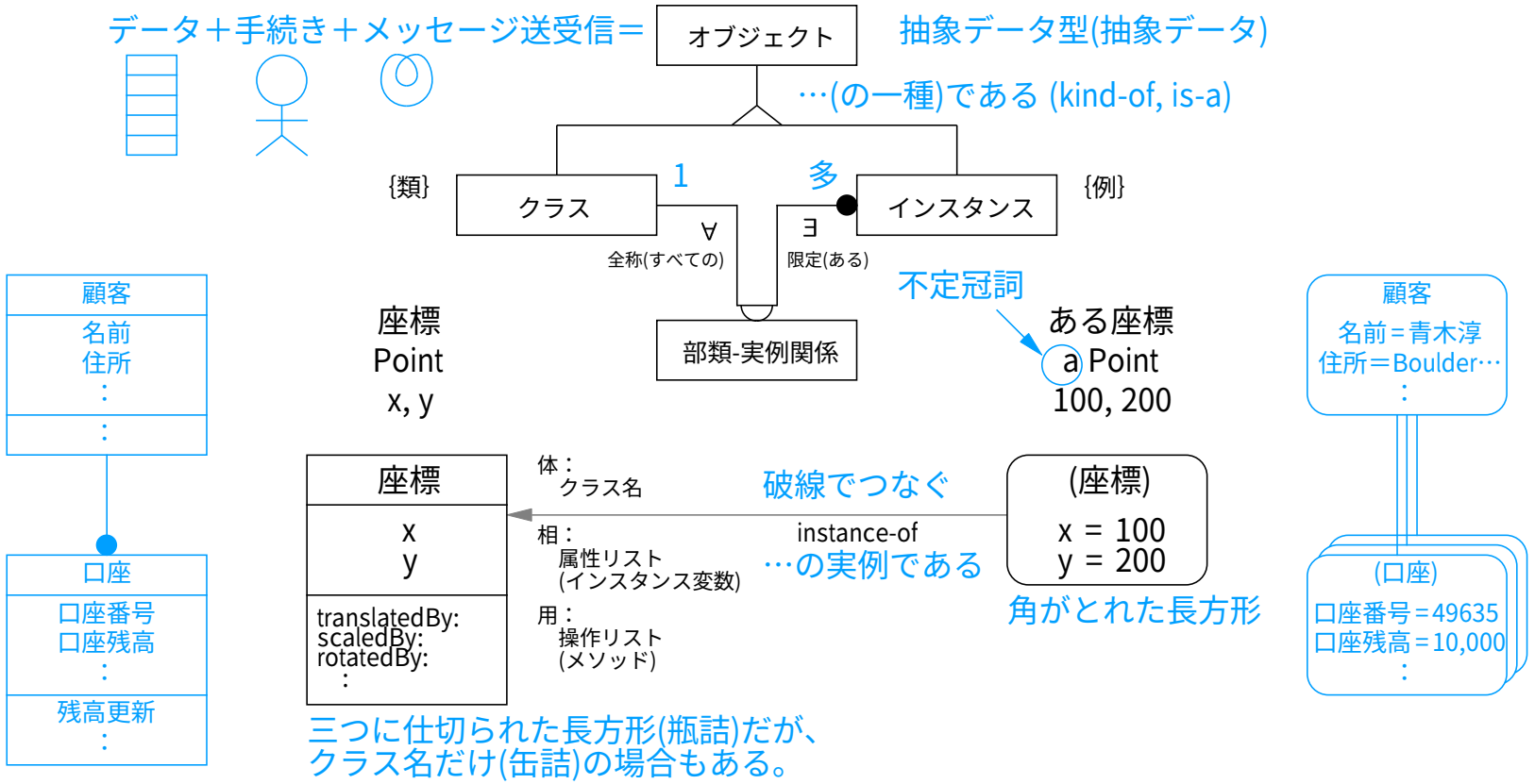
オブジェクト指向をOMTで分析してみましょう。
皆さんの思考をメタレベルへ誘います。

(1) オブジェクト (object), クラス (class), インスタンス (instance)
(2) インヘリタンス (inheritance)
(3) ポリモルフィズム (polymorphism)
(4) 全体-部分関係 (aggregation)



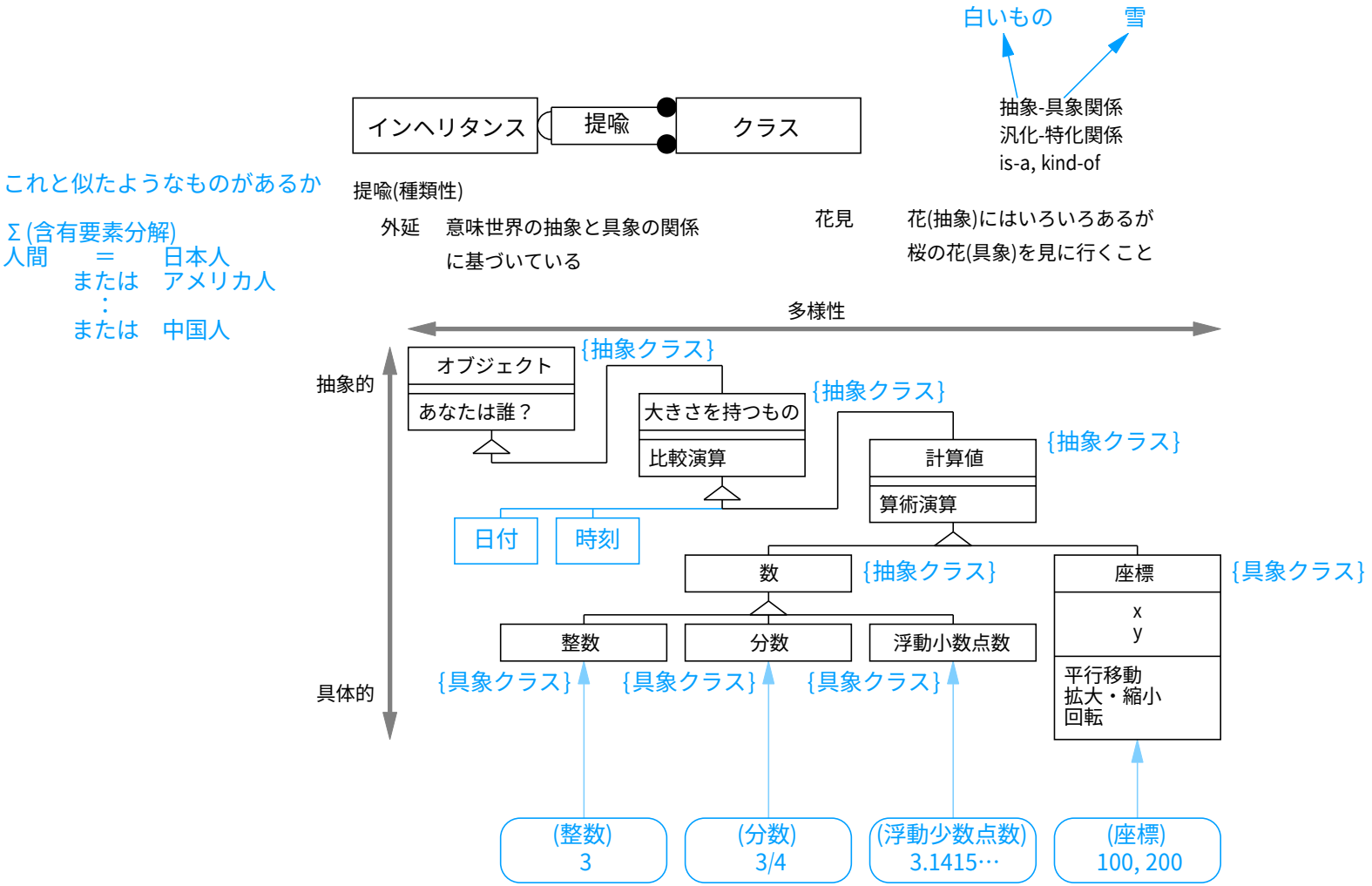
メタレベルのオブジェクト指向

(1) オブジェクト (object), クラス (class), インスタンス (instance)



メタレベルのオブジェクト指向

(2) インヘリタンス (inheritance)　　==>　オブジェクトを分類し整理する機構



メタレベルのオブジェクト指向

(3) ポリモルフィズム (polymorphism)　==>　多相…多くの姿を表現する数学用語

お互いに異なるものを
似ていると考える

ポリモルフィズム　隠喩　オブジェクト

相似や近似の関係

のろまな人　＝　亀

これでどんなことができるか

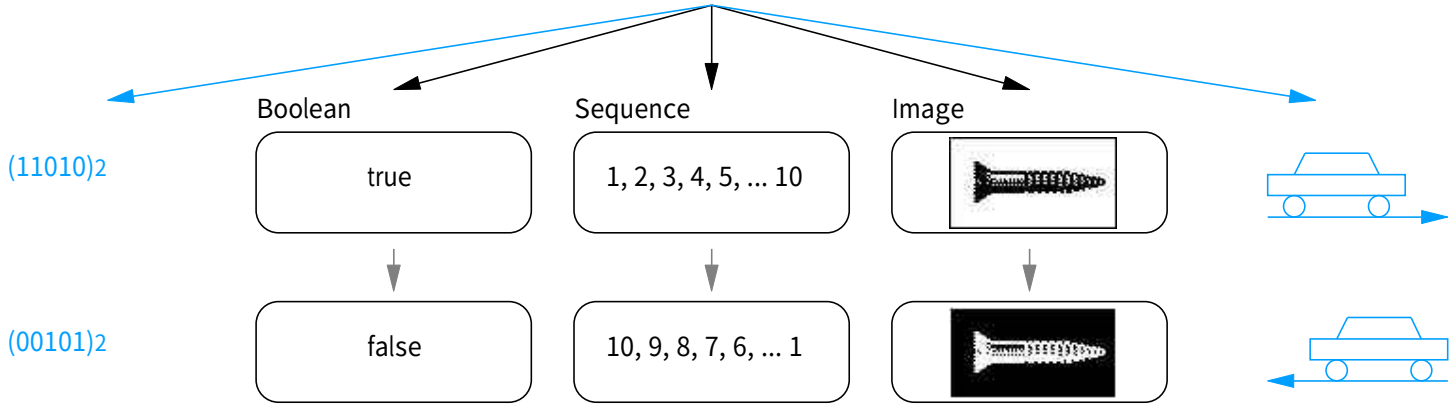
π(意味要素分解)
人間　＝　道具作成
　　かつ　直立二足歩行
　　：
　　かつ　言語活動

隠喩(類似性)

内包　現実世界と意味世界の間の橋
　　渡しをしている

月見　うどんの中に落とした卵と
　　夜空にはえる月とを同一視

Reverse (反転して下さい)　…ポリモフィックメッセージ



同じメッセージでも受手のオブジェクトによって処方が異なる
様々なオブジェクトを統一的に扱うことを可能にする

メタレベルのオブジェクト指向

(4) 全体-部分関係 (aggregation)　　==>　集約…あつめてまとめつづめること

has-a, part-of

これは何からできるか

Π(構成要素分解)
人間　＝　頭
　　および　首
　　：
　　および　足

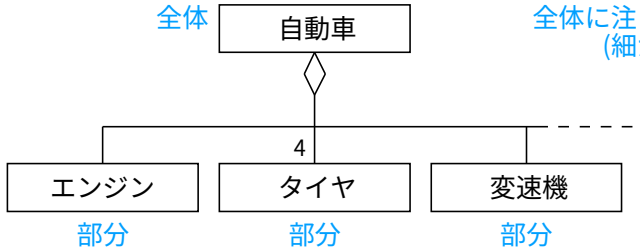
換喩(隣接性)

属性　現実世界の全体と部分の関係に基づいている

形見　身近に付けていたもの(部分)から亡くなった人(全体)を偲んでいる

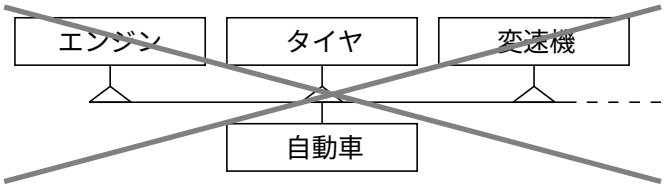
白バイに捕まる
＝警官に捕まる

全体



全体に注目すると部分が隠蔽される
(細かいことを言うな！)

多重継承の誤った使い方　　よくある間違い！



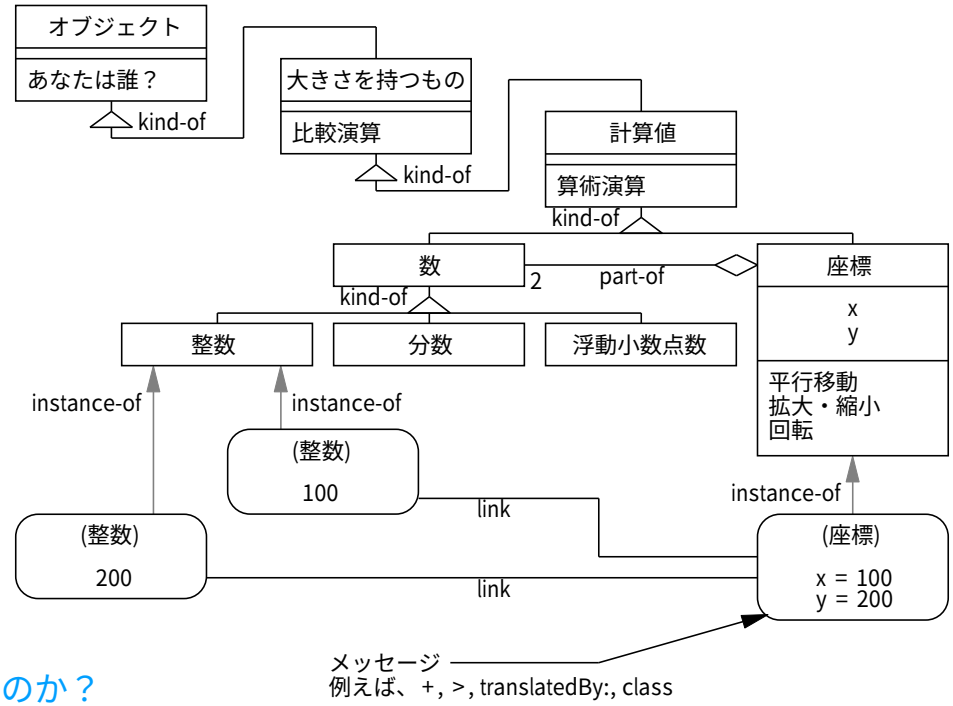
全体-部分関係をインヘリタンス(抽象-具象関係)と混同しないこと

メタレベルのオブジェクト指向

インスタンスに送られたメッセージは、そのメッセージ名を頼りに、
クラス(インヘリタンス)の操作リストの中から検索される。

メッセージ送信はサブルーチンコールではない！

ポリモフィックメッセージは重要だぞ！



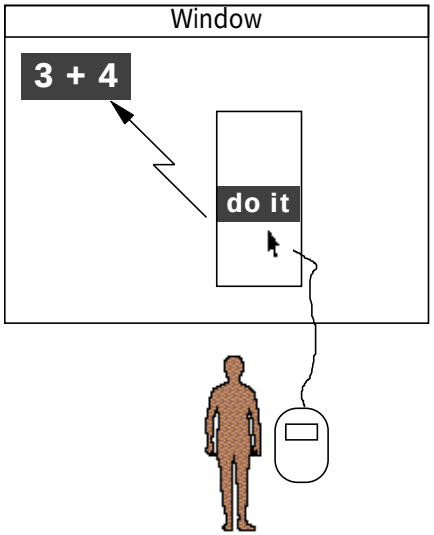
誰が繰り返しをするのか？

誰が条件分岐をするのか？

メッセージがすべて文字列と見なせることを得心していますか？

オブジェクト指向プログラミング(OOP)

メッセージ送信　＝　オブジェクト指向インタフェース (OO-GUI)



1. センダがいる (アプリオリ)
2. まず、レシーバを決める
3. 次にメッセージを送信する

オブジェクト指向インタフェースではない

どこかのワープロ

1. 削除　　2. 削除範囲指定

ほとんどのOS

1. コマンド　　2. 操作されるもの

オブジェクト指向インタフェースの原理
レシーバを最初に決める　＝　操作されるものを最初に指定する

オブジェクト指向プログラミング(OOP)

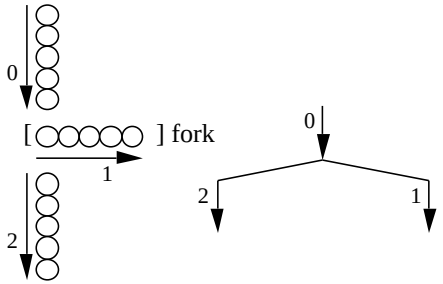
まともなオブジェクト指向言語なら、
これら三つのメッセージ送信が可能です。

メッセージ送信形態

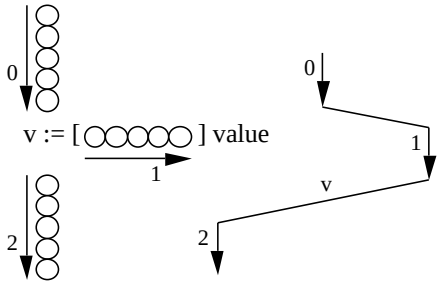
並行
(async.)

約束
(sync.)

1. fork (past message processing)

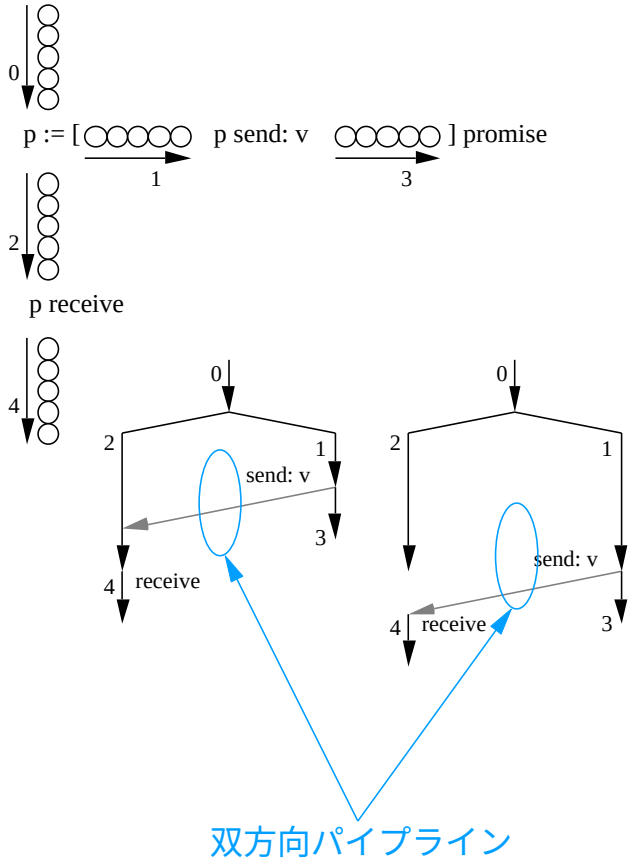


2. value (current message processing)



サブルーチンコール
に似ている
(seq.)

3. promise (future message processing)



双方向パイプライン

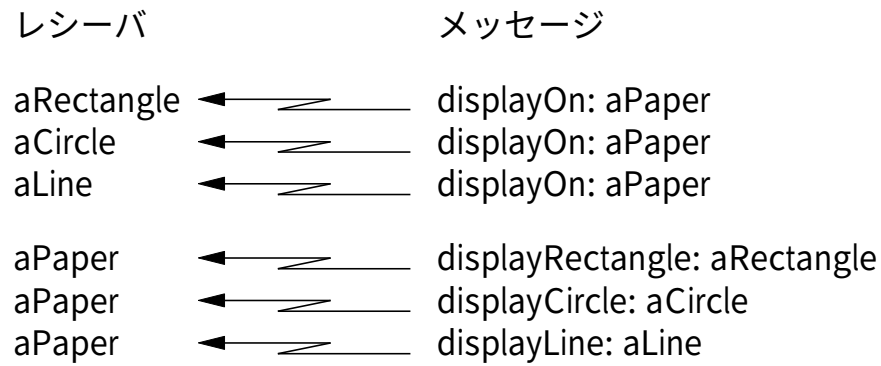
オブジェクト指向プログラミング(OOP)

多相制御構造

ポリモフィックなのはどちらですか？

1. 長方形さん、この紙の上に、あなたを描いて下さい。

2. 紙さん、あなたの上に、この長方形を描いて下さい。



レシーバを統一するよりも、メッセージを統一する方が、OOPらしい！

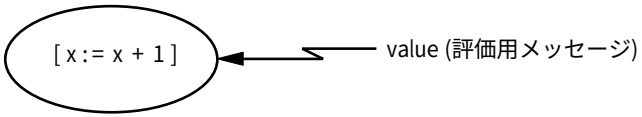
オブジェクト指向プログラミング(OOP)

繰返制御構造

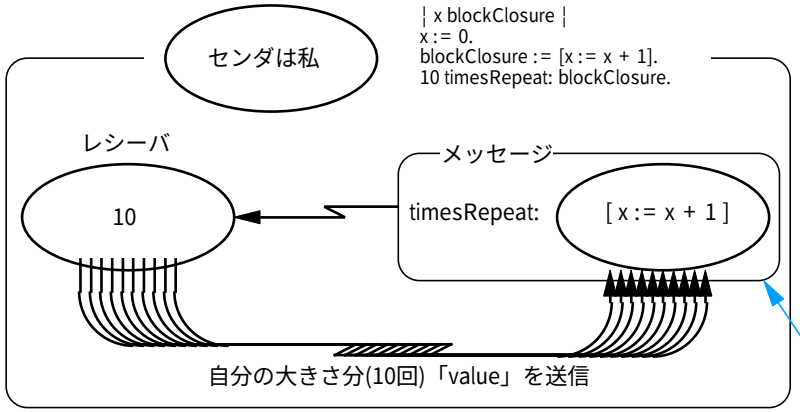
閉包と遅延評価

x := x + 1　　通常のメッセージ式

[x := x + 1]　　ブロック・クロージャ
(アルゴリズムを表すオブジェクト)



あなたの値は何ですか？
計算して下さい。



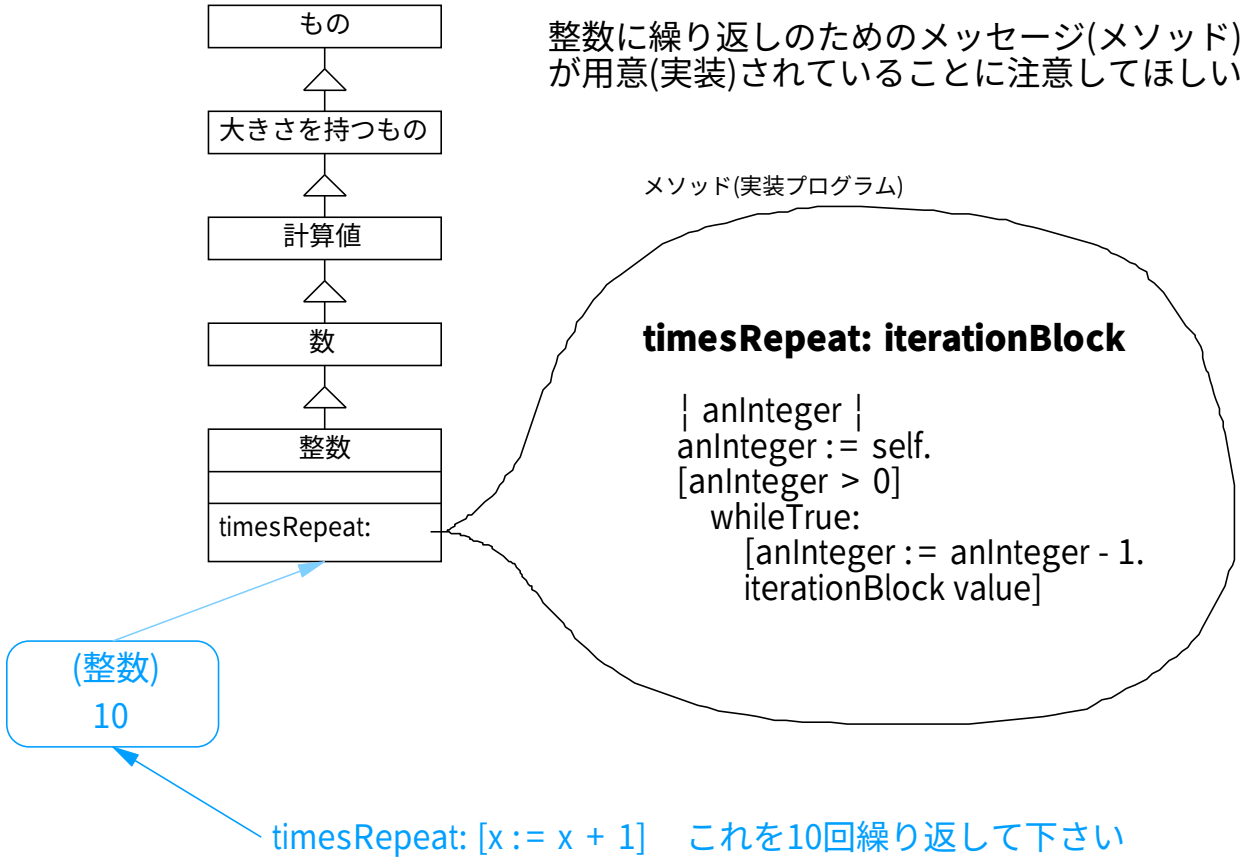
10という整数が「あなたの値は何ですか？」
と10回叫んで繰り返しが行なわれている。

10というオブジェクト自身が繰り返しをしている。

10さん、あなたの大きさ分、
繰り返して下さい。

オブジェクト指向プログラミング(OOP)

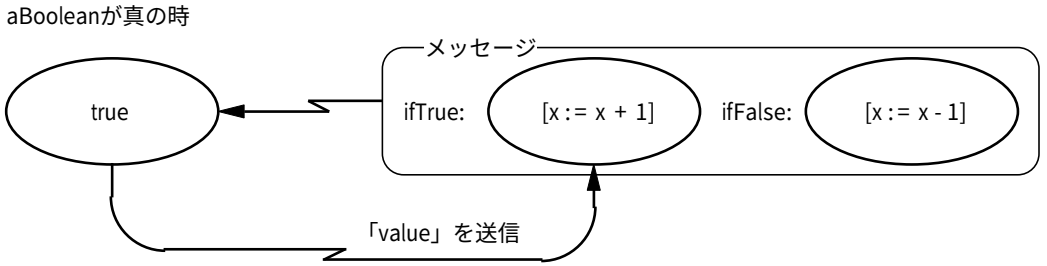
繰返制御構造 (つづき)



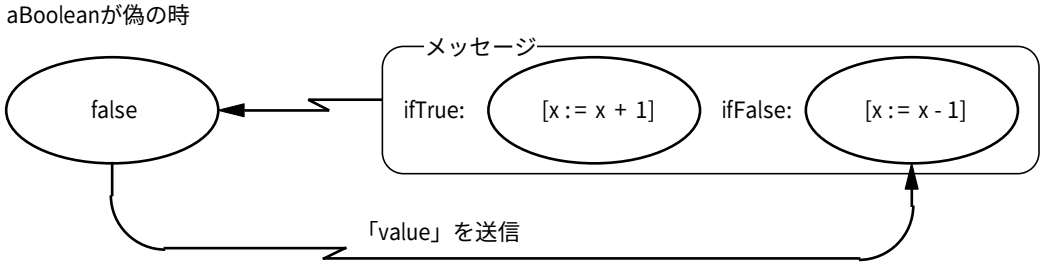
オブジェクト指向プログラミング(OOP)

分岐制御構造

あなたが真ならは、これを実行して下さい。



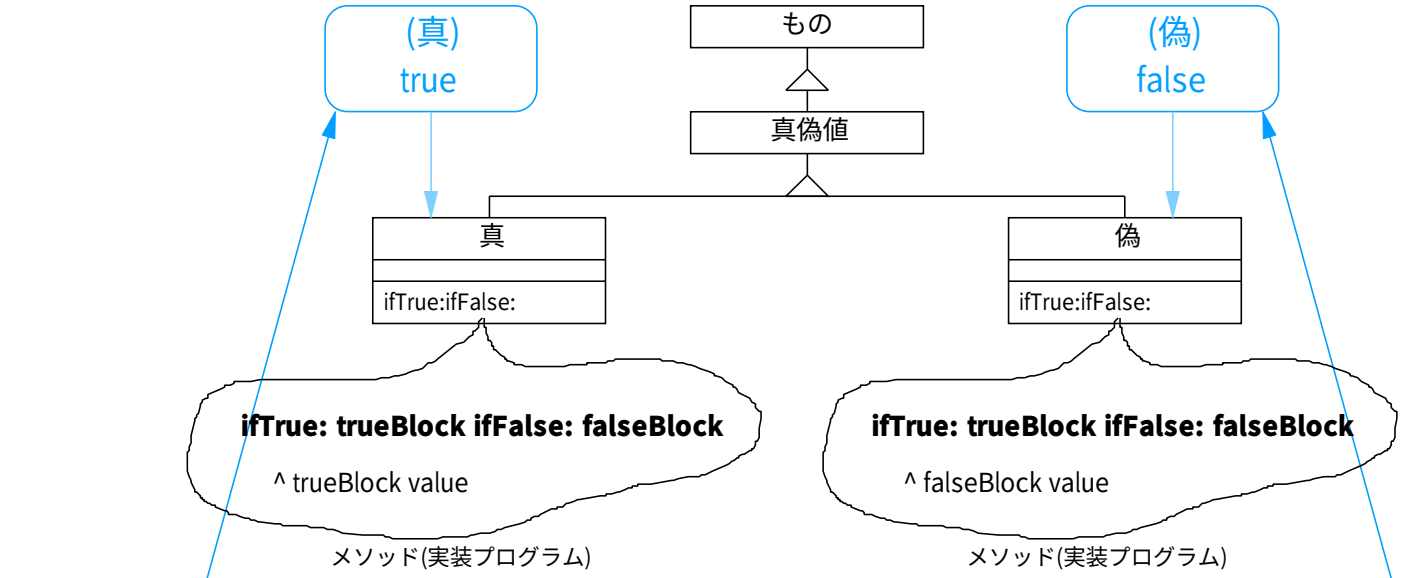
レシーバが
真偽値になっている



あなたが偽ならば、それを実行して下さい。

オブジェクト指向プログラミング(OOP)

分岐制御構造 (つづき)



真と偽に条件分岐のためのメッセージ(メソッド)
が用意(実装)されていることに注意してほしい

ifTrue: [x := x + 1] ifFalse: [x := x - 1]

ifTrue: [x := x + 1] ifFalse: [x := x - 1]

オブジェクト指向プログラミング(OOP)

★ メッセージ送信はサブルーチンコールではない！

手続きにデータを渡すのではない（手続きに引数でデータを渡すのではない）
オブジェクトに手続きの名前を渡す（オブジェクトにメッセージを送信する）

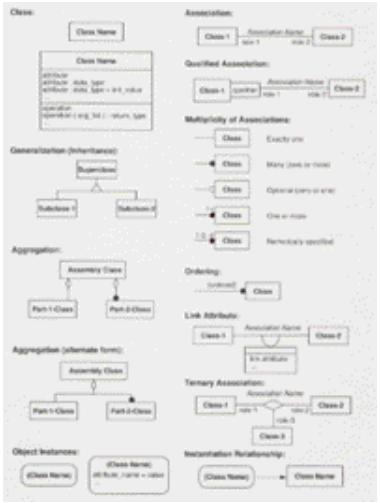
★ ポリモフィックメッセージは重要だぞ！

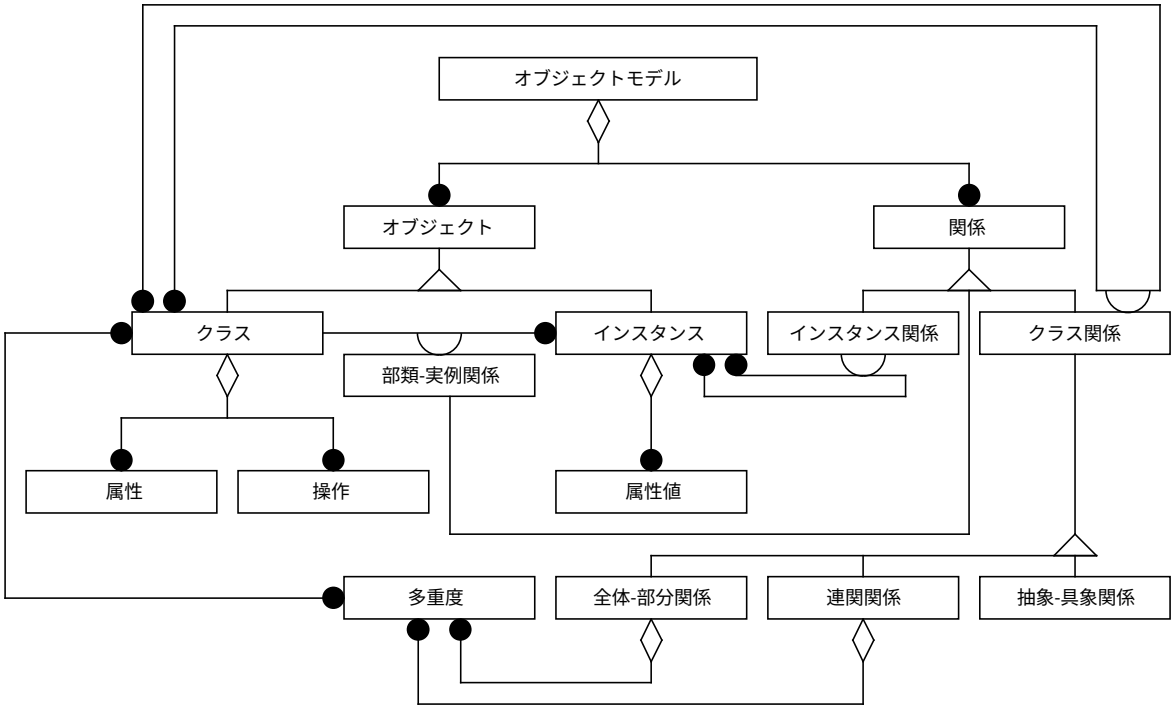
自分勝手なメッセージを実装するのではなく、メッセージ名は出来るかぎり統一すること
オブジェクト指向プログラミングは語彙の辞書(シソーラス)を作り上げる

★ 誰が繰り返しをするのか？　　★ 誰が条件分岐をするのか？

言語に用意された「do〜while(until)〜」や「if〜then〜else〜」などの予約語ではない
どんなプログラム制御構造が相応しいかは、データ(オブジェクト)自身が知っている

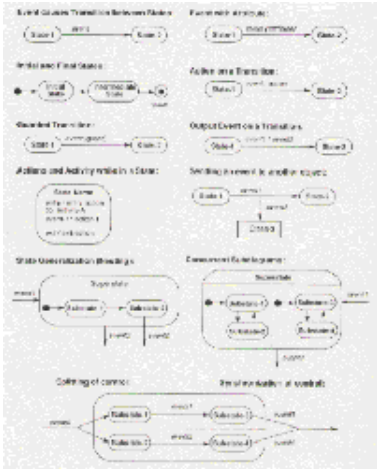
OMTメタモデル(オブジェクトモデル)

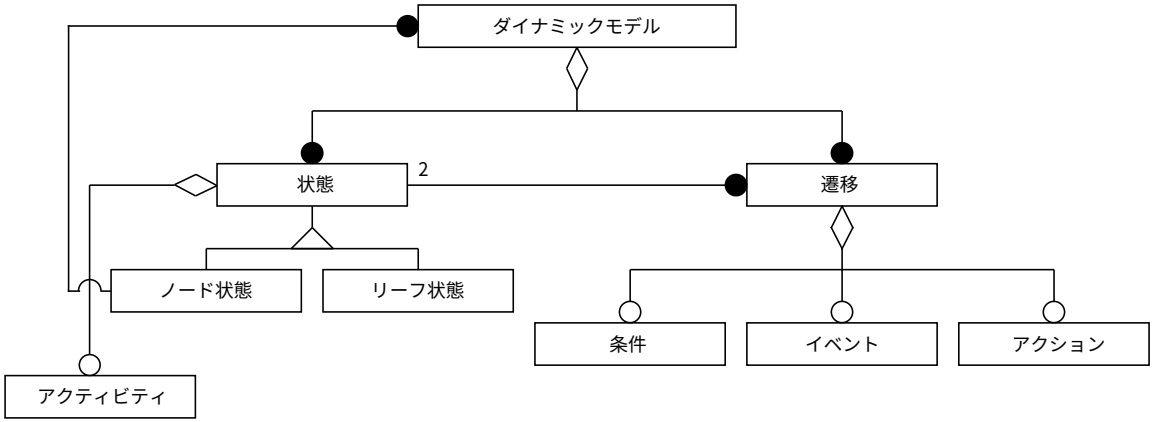




オブジェクトモデルのオブジェクトモデル

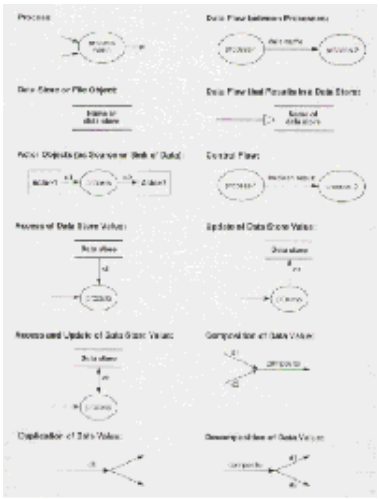
OMTメタモデル(ダイナミックモデル)

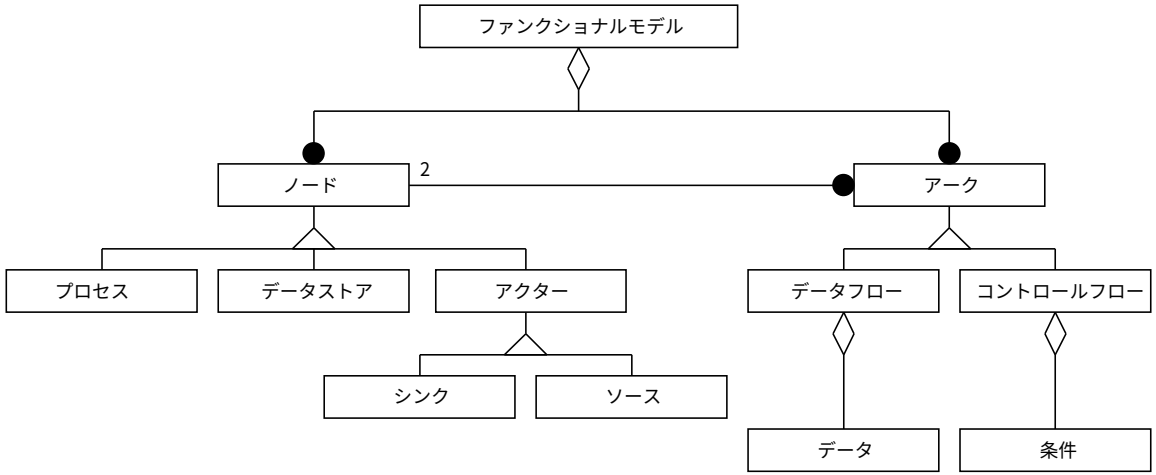




ダイナミックモデルのオブジェクトモデル

OMTメタモデル(ファンクショナルモデル)





ファンクショナルモデルのオブジェクトモデル