オブジェクト指向プログラミング

- 設計から実現へはどう持っていくのですか？
- 既存クラス・ライブラリーが膨大すぎて、どこから手を付けてよいのか分からないのですが？
- 設計モデルのイベントを、イベントとして実現するか操作にするかの指針は何ですか？
- プログラミング上の注意点は何ですか？
- オブジェクト指向言語で実現するのは遅くありませんか？
- どのオブジェクト指向言語が良いのですか？
- コンポーネントウェアを使うと、プログラムが早く作れるのでしょうか？

設計から実現へはどう持っていくのですか？

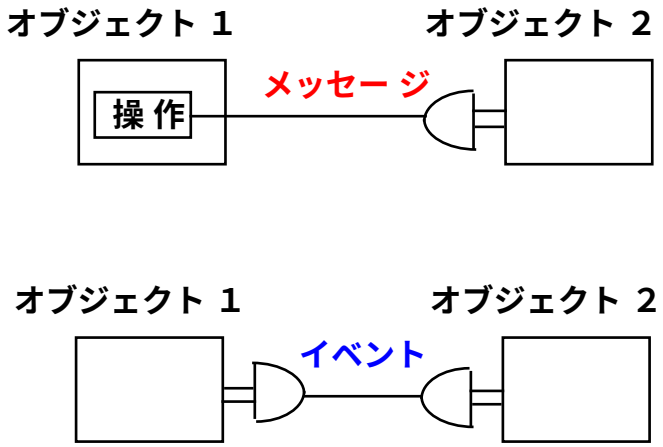
- 設計段階のオブジェクト図は言語独立なので、クラス階層をプログラミング言語にあわせて修正する
  - ここでは、独自のクラス階層を作り出すより、既存クラス・ライブラリーを調査し、既存のクラスのサブクラスとして実現することが多い。
- 属性の型をプログラミング言語に最適なものにする
  - すべての属性について、型あるいは所属クラスを決めなければならない。
- 動的モデルのイベントの実現方法を決める
  - イベントとして実現するか操作にするか決める。

既存クラス・ライブラリーが膨大すぎて、どこから手を付けてよいのか分からないのですが？

- 大きく３つのクラス群に分けて考える
  - 対象問題領域のモデルを記述しているクラス群
    - コンテナー・クラス（Smalltalkの場合Collectionクラス）のサブクラスとなる場合が多い
  - モデルの見え方（ビュー）を記述しているクラス群
    - 最近では、GUIに関わるクラスを設計する必要はあまり無く、コンポーネントウェアやGUI構築ツールを使えば、クラスをあまり意識せずにGUIを構築できることも多い
  - アプリケーションの構造を記述するクラス群
    - コンポーネントウェアなどでは、アプリケーション・フレームワーク・クラスまったく意識せずにアプリケーションを構築できるようになってきた

設計モデルのイベントを、イベントとして実現するか操作にするかの指針は何ですか？

- 操作は、メッセージという形で２つのオブジェクトを接続するため、効率は良いものの、２つのオブジェクトは比較的密に結合してしまう
- イベントは、２つのオブジェクトとは独立の概念なので、効率は悪いものの、付け替えが比較的容易であり、２つのオブジェクトの結合は緩やかになり、独立性は高まる



プログラミング上の注意点は何ですか？

- 再利用性
- 拡張性
- 頑丈さ

オブジェクト指向言語で実現するのは遅くありませんか？

- 初期のSmalltalkやLispベースの言語がインタープリタだったため
  - 現在はインクリメンタル・コンパイラ
  - 動的束縛が遅い原因のひとつ
    - メソッドキャッシュとかハッシュ表の使用により、動的束縛のコストは一定に抑えられるようになった
      - 現在では高々５０%遅いだけ
    - 十分な情報が与えられれば、ほとんどのメソッド呼び出しは動的にならず、静的に行うことができる
- 成熟したクラスライブラリーを持つOO言語では、非OO言語より速いこともある
  - OO言語のオーバーヘッドより、成熟したクラスのデータ構造やアルゴリズムの実現による効率向上の方が大きい
    - 例えばほとんどのプログラマはハッシュ表やバランス木を使おうとしないが、良いクラスライブラリーではこれらが用意されている

どのオブジェクト指向言語が良いのですか？

	C++	Smalltalk	CLOS	Eiffel	Objective-C
強い型付けによるチェック	Y	N	N	Y	Y
可視性					
顧客からアクセスの制御	Y	Y	N	Y	Y
サブクラスからのアクセスの制御	Y	N	N	Y	N
パラメータ化クラス	Y	不要	不要	Y	N
表明と制約	N	N	N	Y	N
実行時のメタデータ	N	Y	Y	N	Y
ガーベージコレクション	N	Y	Y	Y	N
効率					
可能なとき静的束縛	Y	常に動的束縛	常に動的束縛	Y	Y
標準クラスライブラリー	N	Y	N	Y	Y
多重継承	Y	N	Y	Y	N

コンポーネントウェアを使うと、プログラムが早く作れるのでしょうか？

- ファンクションポイント法
  - 1FP当たりのソースコード行数

言語	行数/1FP
Assembler	320
C	150
COBOL	106
FORTRAN	106
Pascal	91
PL/I	80
Ada	71
Prolog	64
APL	32
Smalltalk	21
Visual Smalltalk	10
SpreadSheet	6