プログラム言語論
オブジェクト指向

亀山幸義

筑波大学コンピュータサイエンス専攻

筑波大学 情報科学類 講義

オブジェクト指向

Object Orientation
Object Oriented (OO) Programming Languages
質問: オブジェクト指向プログラミングとは何か？

- オブジェクトを持つプログラムを作成すること？
- オブジェクトを構成要素としたプログラムを作成すること (制御あるいは手続きを構成要素としたプログラムではない)？
- 「オブジェクト」とは何か？

オブジェクトとは？

それに所属するデータたちと、それを操作する関数たちをまとめて「1つ」にしたもの。

- 操作する関数: メソッド (method, member function)
- 所属するデータ: インスタンス変数 (instance variable, field, data member)
- オブジェクトのインタフェース: メソッドのうち公開されているもの (public) の名前や型。
- オブジェクトの実装: メソッドやインスタンス変数の具体的な実現方法。

クラス: オブジェクトの「型」のようなもの。(ただし,「クラス」という概念がない OO 言語もある。例 JavaScript)

オブジェクト指向の基本概念

- Dynamic lookup 動的ルックアップ
- Abstraction 抽象化
- Subtyping サブタイピング (部分型付け)
- Inheritance 継承

引用元: J. C. Mitchell, “Concepts in Programming Languages”, 2003.

Dynamic Lookup

Lookup とは？

- メソッドの名前) から、実際に起動されるべきメソッドの実装を得ること。
- cf. 変数名から、(現在の環境における) その変数の値を得る。

ルックアップが動的 (dynamic) であるとは？

- ルックアップの結果は、静的に決まるのではない。
- 実行時に決まる。

Dynamic Lookup

```
foo.add(e)
```

- オブジェクト foo に add(e) というメッセージを送信。
- オブジェクト foo が持つ add という名前のメソッドを、e という引数で起動。
- 起動されるメソッドは、オブジェクト foo ごとに決まる。
- プログラム上では同じ変数 foo であっても、あるときは整数オブジェクト、別のときは、集合オブジェクトかもしれない。
- 起動される add メソッドは、実行の時点ごとに (変数 foo の値となるオブジェクトごとに) 異なり得る。

Dynamic Lookup

静的ではなく、動的なルックアップは、プログラミング上、極めて有用。
例: グラフィクスプログラムにおいて、四角形、円、三角形などの図形オブジェクトごとに draw メソッドを用意。

Abstraction

抽象データ型における Abstraction と同様。

- オブジェクトへのアクセスは、インタフェース関数 (メソッド) のみに限定される。
- 実装と仕様 (インタフェース) の分離を達成。

Subtyping (A <: B)

- 型 A が型 B の subtype(部分型) のとき、型 B の式を書くべきところに、型 A の式を書いても良い。[代入可能性]

```
class Point {
  ...
  ... void move (int dx, int dy) { ...}
}
class Circle extends Point {
  ...
  ... void move (int dx, int dy) { ...}
}
```

Point クラスのオブジェクトに対する操作は、Circle クラスのオブジェクトに対しても適用できる。

Subtyping と多相型

OO 言語では:

- move メソッドが Point オブジェクトにも Circle オブジェクトにも適用可能。
- move メソッドは、Point クラスを継承した任意のクラスのオブジェクトに対して適用可能。
- 一種の多相性 (subtyping polymorphism ⇔ ML 言語の parametric polymorphism)

Inheritance

継承によるコード再利用

```
class Point {
  private int x =  ...;
  public  int getX() {...};
  ...
}
class CPoint extends Point {
  private int c;
  public int getC() {...};
  ...
}
```

プログラマは、1 つのコードを 2 回書かない。
処理系内部でも、1 つのコードを 2 重に持たない。

Subtyping vs Inheritance

これらの違いは何か?

- subtyping: 2 つのオブジェクト (やクラス) のインタフェースの間の関係。
- inheritance: 2 つのオブジェクト (やクラス) の実装の間の関係。

いくつかの OO 言語 (C++など) では、両者は緊密な関係にあるが、一般的には、必ずしも一致しない。(継承関係にある 2 つのクラスが、subtyping の関係にないことがある、等。)

大規模ソフトウェアの設計

- 関数 (手続き) 指向 vs オブジェクト指向
- デザインパターン

OO 言語たち

- Simula [1960 年代, K. Nygaard]
- Smalltalk [1970 年代, Xerox PARC 研究所, Alan Kay]
- C++ [1984-, Stroustrup]
- Java [1990-, Gosling]
- Ruby [1993-, Matsumoto]

ML Module vs Object

module と object の比較:

- 基本的な違い: module は内部状態 (OO 言語のインスタンス変数) を持たない。
- 抽象化: 同じ。
- 関数のルックアップ: module は静的, object は動的。
- 継承: module に継承はないが、実装の再利用は可能。
- サブタイピング: module にはサブタイピング機能はない。

まとめ

- オブジェクト指向の 4 つの基本概念
- モジュールとの共通点、相異点

Short Quiz

質問 1. 「動的ルックアップ」とは何か，説明せよ．
質問 2. Java では，変数束縛は静的である一方で，method のルックアップは動的である．なぜそのような設計が良いのか，考えなさい．