OCL

2011年6月14日

海谷 治彦

復習

復習 教科書 p.18〜19

1. コンピュータに行わせたいことを理解する
2. 理解したことを説明できるレベルまで整理する.
3. コンピュータのわかる言葉へ翻訳する

- 私たちが現実世界で行うことをコンピュータに肩代わりさせるためにソフトウェアがある. (教科書 p.18改)

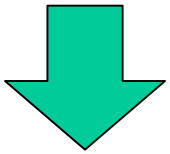
UML図は整理の道具

- 説明しやすいように図になってる.
- クラスやパッケージ等，ソフトウェアの区分けもある.
- クラス間の関係も定義されている.
- それぞれのクラスに属性や操作も定義されている.

- クラス，属性，操作等の意味は名前任せ
- どう実現するかはプログラマの英語日本語読解能力に依存

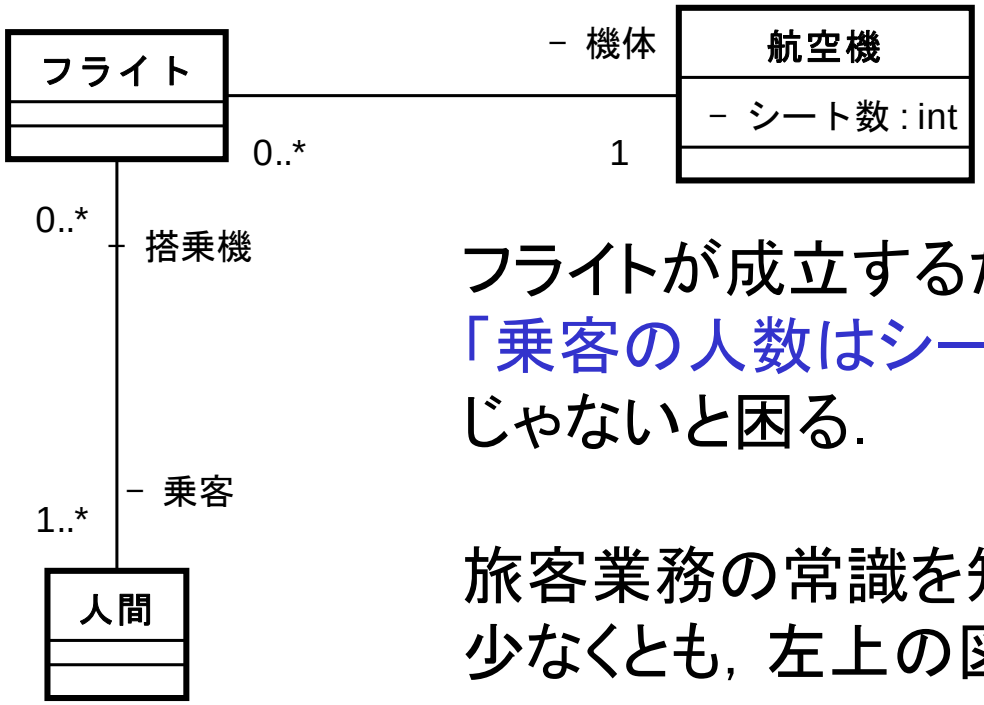
クラス等の意味をより明確に定義

- 説明内容(UML記述等の仕様)の読み手(プログラマ等)の理解が文書読解力任せじゃぁ心もとない.
  - 誤解されて，異なるコードをかかれちゃう.
- そこで，より，あいまい性の無いクラス等の意味の説明をしたい.



- Object Constraint Language (OCL)の登場

動機付けのための例

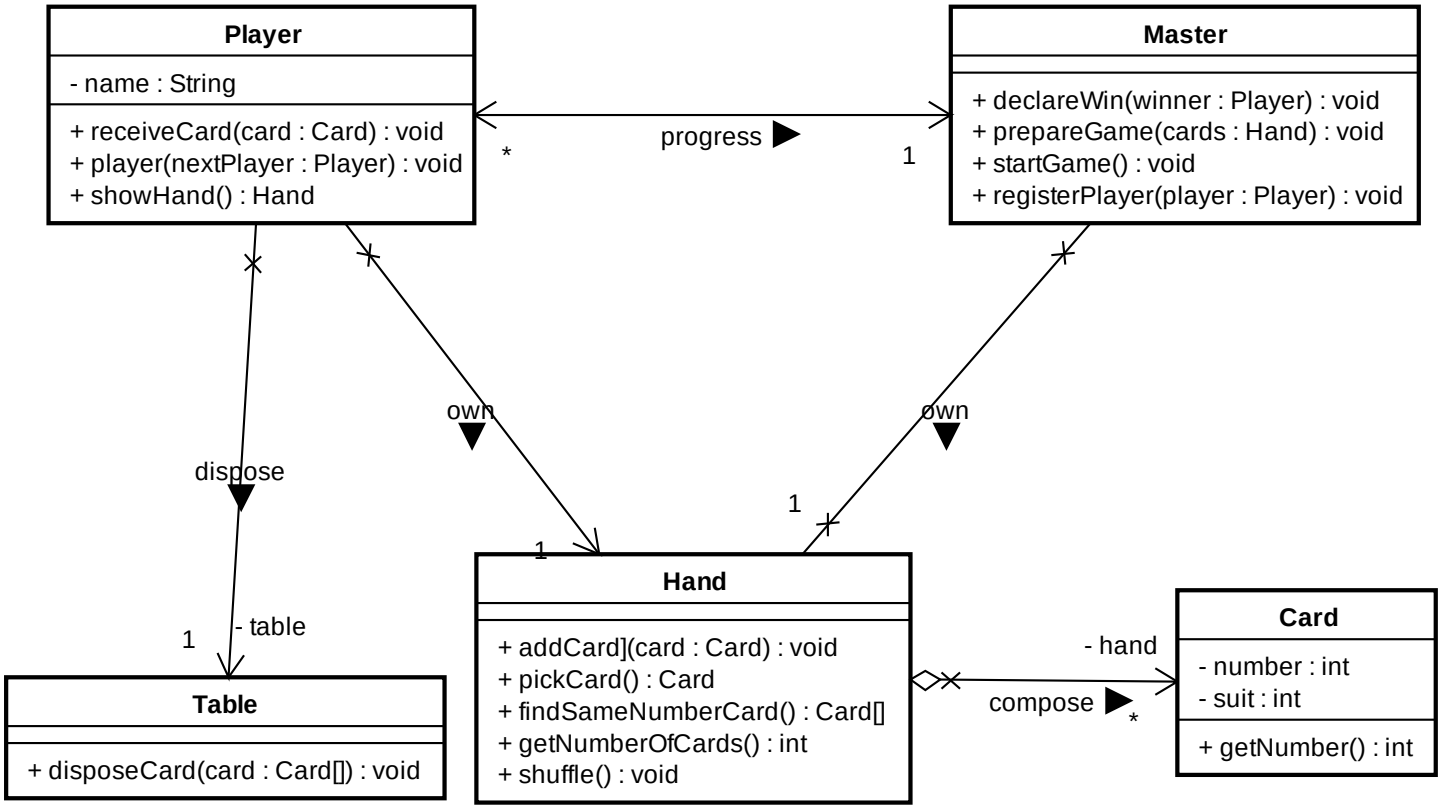


フライトが成立するためには，当然，
「乗客の人数はシート数以下」
じゃないと困る.

旅客業務の常識を知っていれば済む話だが，少なくとも，左上の図には，上記の情報が無い.
上記のような情報を書く言語がOCL.

```
context フライト
inv: 乗客->size() <= 機体.シート数
```

例題 (教科書のばば抜き)


Player
- name : String
+ receiveCard(card : Card) : void
+ player(nextPlayer : Player) : void
+ showHand() : Hand
Master
+ declareWin(winner : Player) : void
+ prepareGame(cards : Hand) : void
+ startGame() : void
+ registerPlayer(player : Player) : void
progress
*
1
own
own
dispose
1
1
1
- table
Table
+ disposeCard(card : Card[]) : void
Hand
+ addCard](card : Card) : void
+ pickCard() : Card
+ findSameNumberCard() : Card[]
+ getNumberOfCards() : int
+ shuffle() : void
- hand
compose
*
Card
- number : int
- suit : int
+ getNumber() : int

不変命題 Invariant

- クラスやその周辺でソフトウェアの動作中にずっと成り立っていなければならないことを書く.
- クラスの特徴と考えてもよい.

例

- トランプの手 Hand には(ババ抜きでは)同じ番号のカードを含まれて居ない.

```
context Hand
inv hand->forAll( a, b |
    a<>b implies a.number <> b.number)
```

以下の論理式をテキストで書いているに同じ

∀a,b (a≠b → a.number≠b.number)

※ 実際は捨てる直前に同じ番号のカードが含まれるのでinvでは無い.

事前事後条件 pre/post

- メソッド(や関数)の意味を形式的に定義する常套手段
- 事前条件 pre: メソッド適用前に成り立つことが述べられている
- 事後条件 post: メソッド適用後に状態がどうかわったかが述べられている.

例

- Hand.addCardメソッド
  - 事前条件
    - 特に無し
  - 事後条件
    - すでにあったカード群に，今追加したカードを足したのが，今のカード

```
context: Hand::addCard(card: Card)
hand@pre->including (card) = hand
```

本授業でのスタンス

- OCLの文法自体は結構独特(一般の集合演算等と記号が異なる)ので，覚える必要は無い．
- ただ，クラスやメソッドの意味を特徴付けるのに，不変命題や，事前，事後条件という考え方をするのは覚えておいて欲しい．
  - 別に日本語での説明でよい．