連載　オブジェクト指向と哲学
第 40 回　集合と写像（７）- 実現態の変化

河合 昭男

オブジェクトの状態変化について考えてきました。オブジェクトの状態は、①属性の値、②他オブジェクトとのリンクの有無、③分類関係の３つであらわすことができるということが前回までの議論です。

前回、オブジェクトの状態をアリストテレスの可能態（デユナミス）と実現態（エネルゲイア）にあてはめて考えましたが、今回は、実現態は変化できるのかについて考えます。また、実現態の変化は、オブジェクト指向ではオブジェクトが属するクラスの変化になりますが、クラスから生成されたオブジェクトを別のクラス型に変更できるのかという問題につながります。

●実現態は変化できるのか

前回挙げた例で、A さんはあるとき入学するというイベントで学生になり、卒業するというイベントで学生でなくなり、入社するというイベントで会社員になります。A さんは生まれたときはまだ可能態であり、成長してから学校を、卒業してからは職業を選択することができます。学生という状態は実現態とみることもできるし、卒業後様々な職業を選択できるという意味では可能態とみることもできます。就職して会社員という状態になったとしても、それは実現態ですが将来独立して起業する可能性もあると言う意味では可能態です。

このように可能態から実現態への状態変化の連鎖が続きます。原因と結果の連鎖と似ています。アリストテレスは終局態（エンテレケイア）というもう１つの状態をつくりました。これは最終目標に到達した状態であり、最早そこから状態は変化しません。人生の終局態というと死になってしまいそうで、それが人生の最終目標というのは「ちょっと違うかな」と思います。人生のゴールはその時点で人様々です。入学、就職、結婚...などいずれも通過点にすぎません。死は目標ではなく人生のタイムリミットです。

人生目標は人様々で、年齢を重ねると段々考えなくなり、次のゴールが明確でなくなってきます。では人生の終局態は何か、アリストテレスに問いかけてみたいです。どのように考えたのでしょうか？

●クラスは変更できるのか

状態変化を分類関係（classification relationship）でみるなら、A さんの例ではあるとき学生クラスとつながり、次に会社員クラスとつながります。ところでオブジェクトの属する

クラスは生成後変更できるものなのかという疑問が生じます。オブジェクト指向の考え方なら、まず鋳型（クラス）がありそこからオブジェクトが生成されます。異なる鋳型からは異なるクラスのオブジェクトが生成されます。生成された後では原則として属するクラスを変更することはできません。

●上位クラス型への変更（アップキャスト）

２つのクラスの間に汎化関係があるとき、サブクラスのインスタンスはスーパークラスのインスタンスに変更することは可能です。プログラミングの世界ではアップキャストと表現します。

例えば図 1 のようにクラス「人」のサブクラスを「学生」および「会社員」とします。学生は人の一種なので、そのインスタンスは人のインスタンスに変更することはできます。図 2 のようにクラスを集合で考えれば A さんは学生という集合の要素ですが、学生は人という集合の部分集合なので同時に集合「人」の要素です。

逆方向はプログラミング用語ではダウンキャストと表現しますが、これは一般的にはできません。集合「人」の要素であっても集合「学生」の要素とは限りません。つまりクラス「人」のインスタンスを無条件にサブクラス「学生」のインスタンスにはできません。

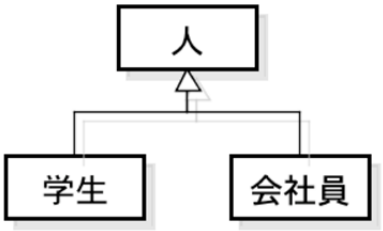

図 1　汎化関係で表す

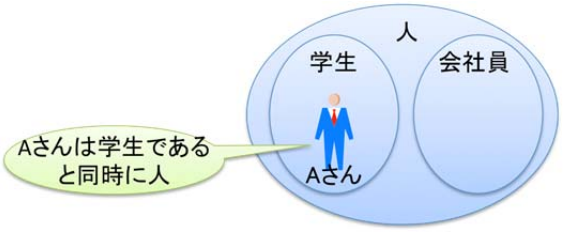

図 2　部分集合で表す

●封建制社会型モデル

まず、クラス型が変更できないモデルを考えます。封建制社会では親の身分・職業が子に引き継がれ職業選択の自由はありません。武士の子は武士、百姓の子は百姓、庄屋の子は庄屋です。このようなモデルをここでは封建制社会型モデルと呼びます。ここで社会制度を議論するつもりではなく、クラス型が変更できないオブジェクト指向のひとつのモデルを便宜上メタファとしてそのように名付けます。

学生、会社員あるいは公務員などをそれぞれ独立したクラスにしてしまいます。このモデルでは一旦生成されたオブジェクトは別のクラス型に変更することはできません。学生というクラスから生成された A さんというオブジェクトは会社員型にはなれません。クラス「会社員」から生成された A さんはそのまま公務員型にはなれません。

このモデルでは学生の A さんが会社員となるためには、会社員オブジェクトを生成し、そ

こに A さんの人としての属性値を会社員オブジェクトにコピーし、学生の A さんを削除しなければなりません。連続的にはできません。人としての属性値は同じですが、学生の A さんと会社員の A さんは全く別のオブジェクトです。

集合で表すなら、図 3 のように学生の A さんと会社員の A さんは別の要素であり、そのまま移れないということです。UML で表すなら図 1 のように人のサブクラスに学生や会社員などの職業のクラスが並ぶ形になります。

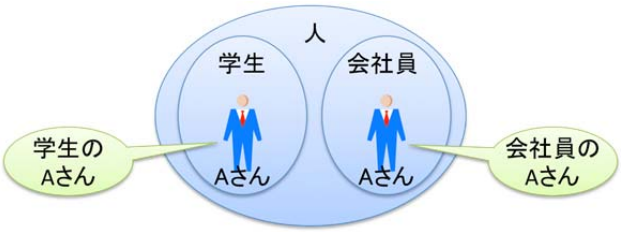

図 3　職業が変更できないモデル

●自由主義社会型モデル

次は職業が変更できるモデルです。本人の生まれや育ちとは関係なく職業を選択でき、転職も自由にできる社会制度をメタファとしたオブジェクト指向モデルを、ここでは自由主義社会型モデルと呼びます。

人の本質的属性と付随的属性を分離します。付随的属性とは職業により異なる属性です。例えば職業として学生、会社員、公務員を考えます。とりあえず職業という集合の要素は「その他」を追加して 4 つのみとします。

職業 ＝{学生, 会社員, 公務員, その他}

図 4 のように A さんと学生を対応付けます。就職すればその対応を会社員に変更します。A さん自体に変更はありません。職業との関係が変化するのみです。関連をタプルで表すなら（A, 学生）が（A, 会社員）に変化しますが、A は存続します。

UML で表すなら図 5 のようになります。このモデルは、すべての人はただひとつの職業を持つことを表しています。社会人学生のような複数の職業を持つ人は表していません。

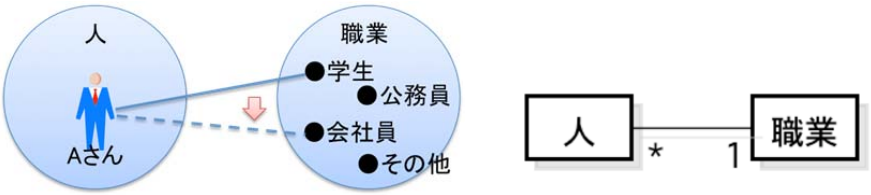

図 4　職業は変更できるモデル　　**図 5 職業変更できるモデルを UML で表す**

図 4 のモデルでは職業は指定できますが、職業固有の様々な属性を指定することはできません。例えば学校名や会社名などの情報は設定できません。図 6 ではそれができるように学生や会社員などを集合「職業」の要素ではなく部分集合とします。

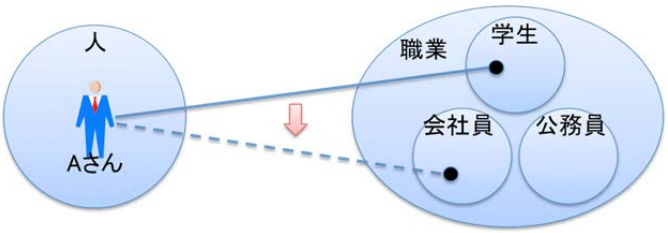

図 6　職業固有の属性を設定できるモデル

このモデルを UML で表すなら図 7 のようになります。

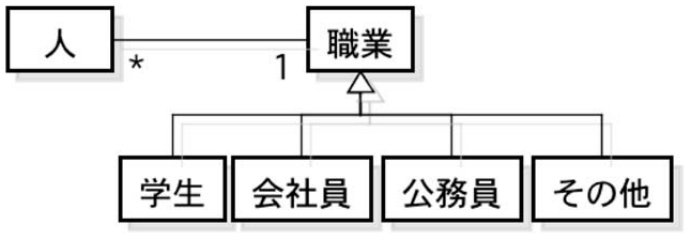

図 7　職業変更できるモデルを UML で表す

●まとめ

今回は状態変化をオブジェクトのクラス型に注目して考えました。

①封建制社会型モデル

本質的属性と付随的属性がひとつのクラスの中に混在するモデル。あるクラスから生成されたオブジェクトを別のクラス型に変更できない。柔軟性に欠けるモデル。

②自由主義社会型モデル

本質的でない付随的属性を切り離し、クラスを２つに分離するモデル。本質は変更することなく、付随的オブジェクトを変更することができる。柔軟的なモデル。

以下次回。