連載　オブジェクト指向と哲学
第 42 回　オブジェクト指向を分析する -概念とクラスの形成

河合 昭男
http://www1.u-netsurf.ne.jp/~Kawai

オブジェクト指向のキーワードは何と言ってもオブジェクトであり、その具体的なオブジェクトを抽象化したクラスです。

プログラミングはクラスの記述から始まりますが、モデリングではクラスの前に概念抽出から始まります。モデリングは様々なレベルで行いますが、上流では問題領域の概念を抽出しその関係を明確にしてゆきます。この概念は開発工程を通してクラスとしての形を整えてゆき、プログラムのクラスにつながってゆきます。

前回は概念の意味付けについて辞書の記述法を参考に考えました。今回は概念とクラスの形成について考えてみたいと思います。

●祖国という概念

第 2 次大戦下ドイツが降伏した頃のヨーロッパで、ポツダム宣言が発せられ祖国の存在が危うくなってきた日本人 A と祖国再興を願うポーランド人 B の会話です。

--

A : あんたはじっさいの故国のありようを愛しているというよりは、・・・祖国という考え方そのものが好きなようだな。そいつがポーランドでも、日本でもいいようだ

B: ・・・わたしが愛しているのは、実体としての故国ではなく、祖国という概念かもしれん。・・・

A : 実体のないものに人生と愛を捧げることは、不毛じゃないのか

B : そうかな。目の前の・・・しか愛せないってことよりは、はるかに意味のあることだと思うが

--

「ストックホルムの密使」（佐々木譲著、下巻 P10-11）より引用。ちなみにこの小説は「ベルリン飛行指令」「エトロフ発緊急電」につづく第 2 次大戦秘話三部作の完結篇です。

会話している当事者それぞれの実体としての故国には深い思い入れがある筈です。B 自身が認識している自分の故国という具体概念には様々なプロパティが紐付いています。名前はポーランド、場所は中央ヨーロッパ、B の住んでいた町は xxx、人口は xxx、現在の状況は xxx、その他民族の文化や歴史など無数にある筈です（図 1）。A 自身が認識している自分の故国という具体概念

にも無数のプロパティが紐付いています。

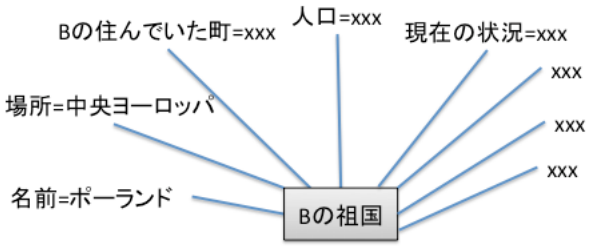

図 1 B の祖国という具体概念

会話しているうちに A は気付きます。B は「じっさいの故国」「実体としての故国」であるポーランドという具体概念ではなく、祖国という抽象概念を愛しているのではないのか。B もそれを肯定します。会話はさらに続きます。B の祖国への思い入れは体験のない人には分かりません。

--

B：きみも、ポーランドのような国に生まれてみろ。繰り返し繰り返しまわりの大国に侵略され、切り刻まれ、収奪されて、自国語での教育すら禁じられたような国にだ。革命と戦争が唯一の希望であったような国にだ。そんな国に生まれたら、祖国、という言葉が、どれほど美しく甘い響きに聴こえることか。その言葉に、どれほど力づけられ、奮い立たされることか

--

具体概念は様々なプロパティを持っています。無数のしがらみが付着しています。それらのしがらみを取り除いた純粋なものが故国の本質つまり抽象概念です。不純物を取り除いたきれいなものを愛することにこそはるかに意味があると考えたのでしょうか。しかもその抽象概念の意味が B のその後の行動原理になってしまうくらいの重みがあるのです。ここでは小説の余計な解説は無用にします。概念とは何かを考える素材として引用しました。

●概念の 3 点セット

前回、概念は名前と内包／外延の 3 点セットで形作られているとしました。この概念は抽象概念で、外延に具体概念が列挙されます。辞書でも文房具の例で説明しましたが、言葉の意味を内包と外延で説明されている場合があります。

上に挙げた例の「故国」「祖国」を辞書で調べるとどちらも「生まれ育った国」と説明されていて、それぞれがおなじ概念の別名です。この説明は内包のみで、外延はありません。それぞれの人によって具体的故国は異なるので、外延を日本／ポーランドなどと列挙することはできません。内包の説明は抽象的です。人生体験により意味内容が大きく異なってきます。

●概念とクラスの形成

概念形成の順序は先に具体概念があり、その本質として抽象概念が生まれ名前が付けられる。抽象概念が先にあり、そこから具体概念が生まれることはないというのが一般的な考え方です。

オブジェクト指向用語では具体概念はオブジェクトに抽象概念はクラスに対応します。モデリングを行う場合は問題領域の中にある具体概念に注目し、その意味的関係を見付けていくつかのオブジェクト図を作成します。次にその作業から抽象概念を見付け、それらの間の意味的関係を見付けてクラス図を作成します。

具体概念／オブジェクトが先で、抽象概念／クラスは後です。ついでながら本来的にはそうなのですが、実際の開発作業では同様の問題領域での経験があればオブジェクト図を作成しないで直接クラス図を作成することができます。

ところがプログラミングではクラスの記述が先です。クラスがなければオブジェクトを生成することはできません。クラスを鋳型にしてオブジェクトを生成します。概念形成と順序が逆です。どこで逆転したのでしょう。設計レベルのクラス図はプログラムと同等です。オブジェクト図は具体例を表すことはできますが、プログラミングに直接必要な情報ではありません。

概念とクラス形成の一連の流れを整理すると次のようになります。

① 具体概念
② 抽象概念
③ 分析レベルのクラス
④ 設計レベルのクラス
⑤ プログラムのクラス
⑥ クラスからインスタンスを生成

つまり①から②で具体から抽象に上がることで始まり、その後②から⑤は抽象の状態でモデル変換が行われ、最後に⑤から⑥で抽象から具体に下りてきます。

クラスがオブジェクトに先行するのは「もの造り」でも同じです。「もの造り」とは「設計情報を媒体に転写すること」という考え方があります[2]。設計情報を鋳型として作成し、媒体に転写すれば同じ形をした製品というオブジェクトがどんどん生まれてきます（図 2）。プログラムでクラスが先にあって、そこから同じ形のインスタンスが生成されてゆくのと似ています。

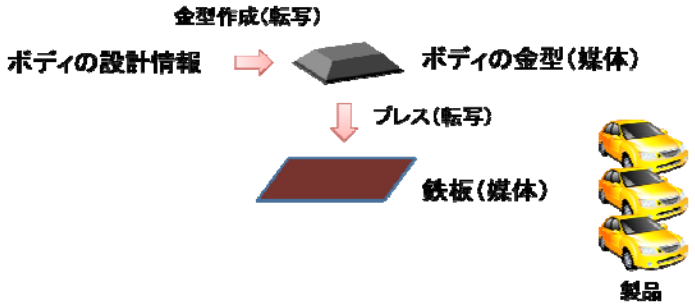

図 2　（連載第 28 回 図 1 再掲）

●リバースエンジニアリング

リバースエンジニアリングにより製品から設計情報をある程度取り出すことができます。これは製品というオブジェクトから設計情報というクラスを抽出することです。これは概念レベルで人が普通に行っている具体概念から抽象概念を導くという行為と方向は同じです。しかし当然ながら製品の設計情報は単純な概念ではありません。詳細で複雑なものであり方向は同じですが同列に考えられません。

ところでプラトンなら何と言うでしょう？人工的なものは設計情報の入った鋳型からできるが、この世界に存在する人工物でないすべてのものはどうやってできたのか？抽象概念は人が便宜上後から作ったものに過ぎないと言っているが、それもじつはこの世界のリバースエンジニアリングに過ぎないのではないか。不完全で未熟であり、本来のこの世界の設計情報は人の理解を越えたものとして厳然として実在しているのだ。それをイデアと呼ぶのだ。

以下、次回。

【参考】

[1]佐々木譲、ストックホルムの密使、新潮文庫、1997
[2]当連載第 28 回 メディアを形相と質料で考える