Ⅱ オブジェクト指向の基本用語

学習目標
　オブジェクト指向の基本的な用語を理解させる。
　　情報の抽象化
　　継承
　　メッセージ
　　多様性
　　クラス
　　オブジェクト

ここでは、Ⅰ章で説明したオブジェクト指向のもつ基本的な仕掛けや新しい考え方を、オブジェクト指向の専門用語を使って言い直すとどのようになるかについて説明する。これから紹介する専門用語は、知ってさえいれば何も恐れることはないが、もし知らなければオブジェクト指向の本を読んでも、少しも理解できないであろう。

1 情報の抽象化

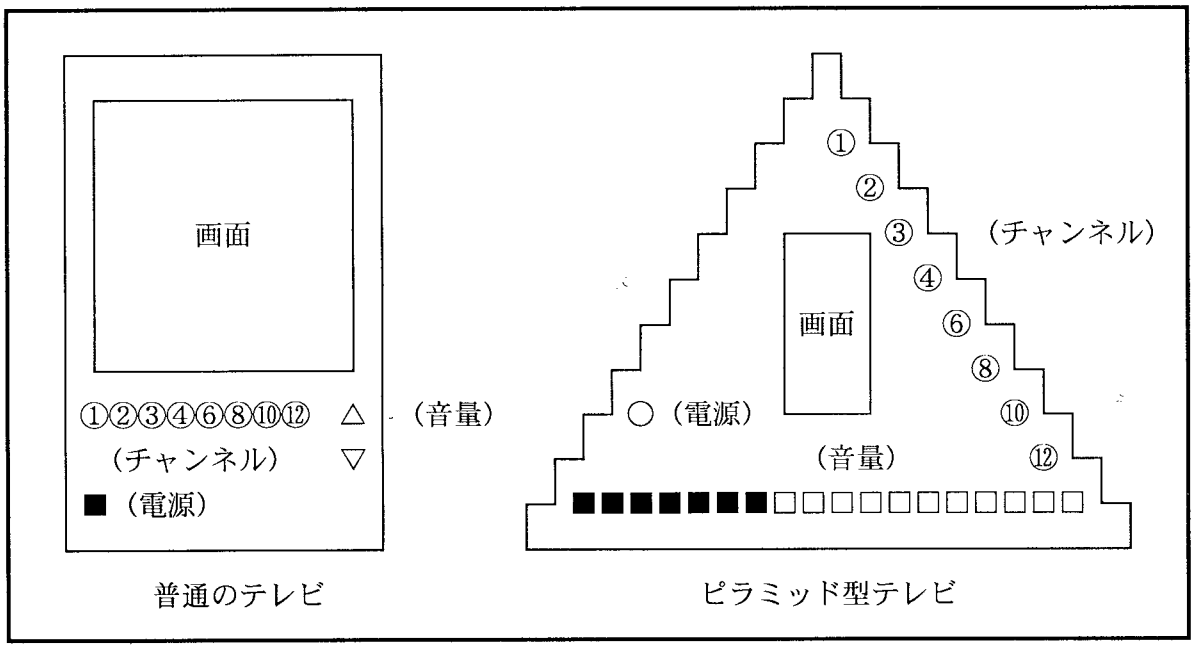
「情報の抽象化」とは、オブジェクト指向の「ある人が作った部分を他人に使わせる場合、具体的なプログラミングの詳細は教えず、使い方だけを教えるという仕掛け」のことである。これは、また、「情報のカプセル化」、「情報隠蔽」と呼ばれることもある。

情報の抽象化を具体的な例で考えてみる。 例えば、私たちはテレビを見るとき、電源スイッチを入れたり切ったり、チャンネルを切り換えたり、音量ボリュームを大きくしたり小さくしたりする。そこで、図Ⅱ－１に示した２種類の異なった形をもつテレビを例に考えてみる。

まず、左側に普通のテレビがある。もし、このテレビで６チャンネルの番組を楽しもうとする場合、普通の人なら電源スイッチを入れ、その後６チャンネルを選択するであろう。また、もし音量が小さかったら、音量のボリュームを変え音を大きくするであろう。

次に、右側にピラミッド型のテレビがある。このテレビは、普通のテレビと形はかなり異なるが、電源スイッチ、チャンネル、音量ボリュームがあるところは同じである。そこで、もし、このテレビで６チャンネルの番組を楽しもうとする場合、やはり、電源スイッチを入れ、その後６チャンネルを選択するであろう。また、もし音量が小さかったら、音量のボリュームを変え音を大きくするであろう。

このように図Ⅱ－１に示した２種類のテレビは、それぞれ形は異なるが、私たちは同じ使い方で利用できる。



図Ⅱ－１　異なる形のテレビ

テレビは、その使い方さえ知っていれば、内部でどのような方式によって画像が画面に表示されるのかといった技術的な知識はなくても、その上で番組を楽しむことができる。このような場合、テレビは「ブラックボックス」であると考えることができる。ブラックボックスとは、箱の内容を具体的に知らなくとも、使い方さえわかればそれで用が済むとする考え方である。

もし、外側から、テレビの内部に直接手を触れられるようになっているテレビがあるとしたらどうであろうか。テレビの中には、高電圧の部分もある。そのため、感電事故が多発するであろう。ブラックボックスの内部には、外部から直接触れないようにすることが必要である。どうしても触らないといけない、チャンネルの切り替えや音量の調節などの機能は、それら専用のつまみをテレビの外側に取り付け、そこを経由して操作するようにする。

これと同様のことが、ソフトウェアの開発についてもいえる。オブジェクト指向において、情報の抽象化、情報のカプセル化、情報隠蔽とは、プログラムの各部分をブラックボックス化し、その使い方だけを他人に教えることを意味する。それによって、他人はプログラムの各部分をテレビのように手軽に利用できるようになり、その結果ソフトウェア開発の生産性が向上する。

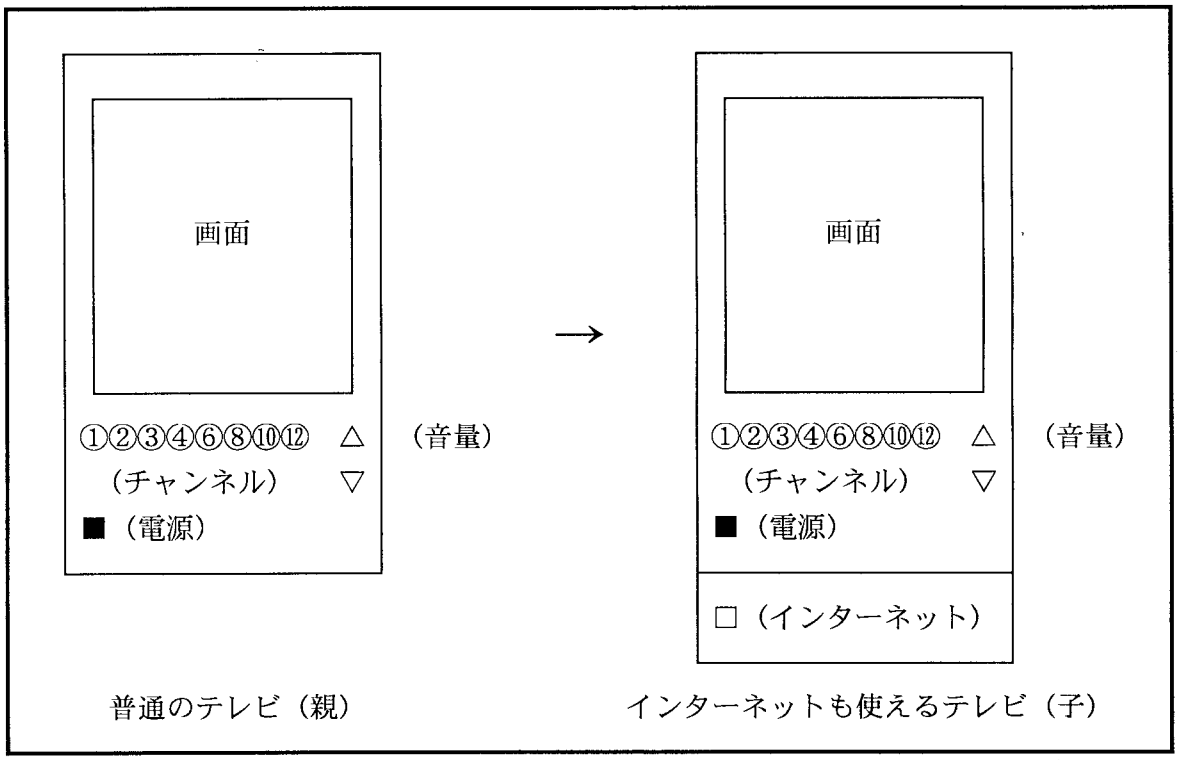
また、プログラムの各部分の内部に他人が直接触れられないようにすることによって、内部データを他人が誤って操作する危険も避けられ、ソフトウェアの信頼性が向上する。また、どうしても触らないといけない機能は、必ず外部と内部を仲介する仲介者を経由して操作するようにする。

2 継　承

「継承」とは、オブジェクト指向の「ある人が作った部分は、なるべく流用し、もう一度作り直すことはしないという仕掛け」のことである。ある人が作った部分が親に相当し、流

用した部分が子に相当する。継承により子を作るとき、親の性質に変更を加え、子に固有の性質を持つことができる。

継承を具体的な例で考えてみる。例えば、私たちが電気メーカのテレビ開発部門に勤めており、今度新しくインターネットも使えるテレビの開発をすることになったとする。このような場合、既に普通のテレビは開発済みなので、その部分はそのままにして、新しくインターネットを使う部分だけ開発すればよい。普通のテレビが親だとすれば、そのほとんどを継承した子として、インターネットも使えるテレビを開発することになる。両者の関係を図Ⅱ－２に示す。



図Ⅱ－２　普通のテレビとインターネットも使えるテレビ

これまで普通のテレビを使ったことのあるような人なら、インターネットも使えるテレビも、違和感なく使えるはずである。それは、インターネット用のボタンが１個増えただけで、後の電源スイッチ、チャンネル、音量ボリュームは、普通のテレビと全く同じであるからである。

このように、既にあるものはなるべく流用し、必要なところだけを付け加えて、新しいものを作るという開発手法をソフトウェアの開発に適用したものが、オブジェクト指向でいう継承である。継承により親から子を作るとき、親のもっている性質を変更し、子に固有の性質を持たせることができる。

また、継承は何代にもわたって行うことができる。例えば、インターネットも使えるテレビを親としてその性質を継承し、インターネットも使えビデオも使えるテレビを子として作り出すようなことも、ソフトウェア開発において実現することができる。

しかし、あまり継承を使いすぎると、親子の関係が複雑になりすぎて、プログラムが理解しにくくなるので、やたらに継承を何代にもわたって続けることは、あまり良いことではな

い。

ところで、これまで説明してきたのは、一つの親クラスから属性を受け継いで、そこから子クラスを作成するものであった。そのため、これは単一継承（Single Inheritance）と呼ばれる。一方、オブジェクト指向では、二つ以上の親クラスから属性を受け継いで、そこから子クラスを作成する継承も考えることができる。これは、多重継承（Multiple Inheritance）と呼ばれる。

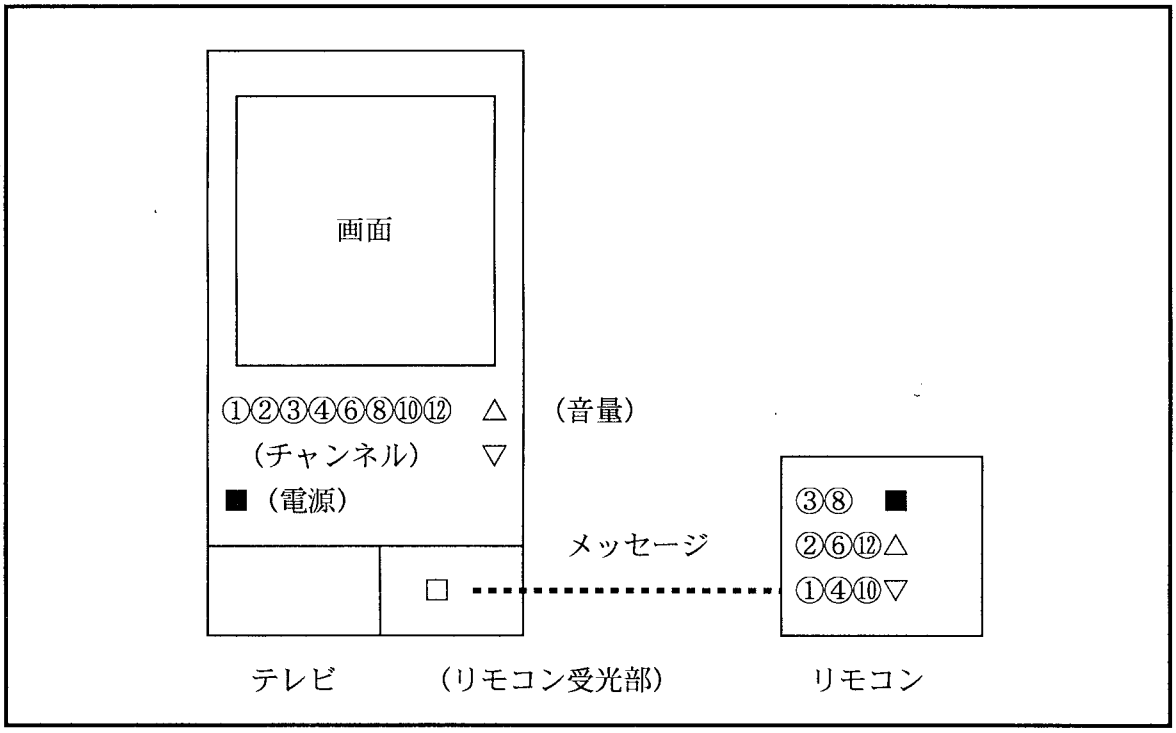
多重継承の機能があると、より簡単にクラスを作成できるという考え方もある。しかし、親クラス間の名前が衝突したり、かえって親子関係が複雑になる可能性があり、特に多重継承がなくても、単一継承だけで困らないという考え方もある。実際、Ｃ＋＋言語は多重継承をサポートしているが、Ｊａｖａ言語のように、多重継承をサポートしていないオブジェクト指向プログラミング言語も存在する。

3 メッセージ

メッセージとは、オブジェクト指向の「自分で何でもやってしまうのではなく、他の人にお願いをして処理をしてもらうという仕掛け」のことである。

情報の抽象化で説明したように、内部データに対する操作は、必ず仲介者を経由して行うようにする。決して、オブジェクトの内部データに直接触れるような操作をしてはならない。その仲介者に対し、操作を依頼する処理がメッセージである。

メッセージを具体的な例で考えてみる。例えば、リモコン付きのテレビを考えてみる。リモコンを使うと、私たちは直接テレビのスイッチなどを操作しなくとも、遠く離れた場所からテレビを操作することができる。その様子を図Ⅱ－３に示す。



図Ⅱ－３　リモコンによるテレビの操作

電源スイッチを入れたり、チャンネルを選択したり、音量ボリュームを変えたりする操作は、テレビから離れたところにあるリモコンから、テレビのリモコン受光部に光を送ることにより行うことができる。このとき送られる光は、リモコンからテレビに送られたメッセージであると考えることができる。

この考え方をソフトウェアの開発に適用したのが、オブジェクト指向でいうメッセージである。オブジェクト指向では自分で直接内部データを操作せず仲介者に対して、メッセージを送り操作を依頼する。これによって、内部データを他人が誤って操作する危険も避けられ、ソフトウェアの信頼性が向上する。

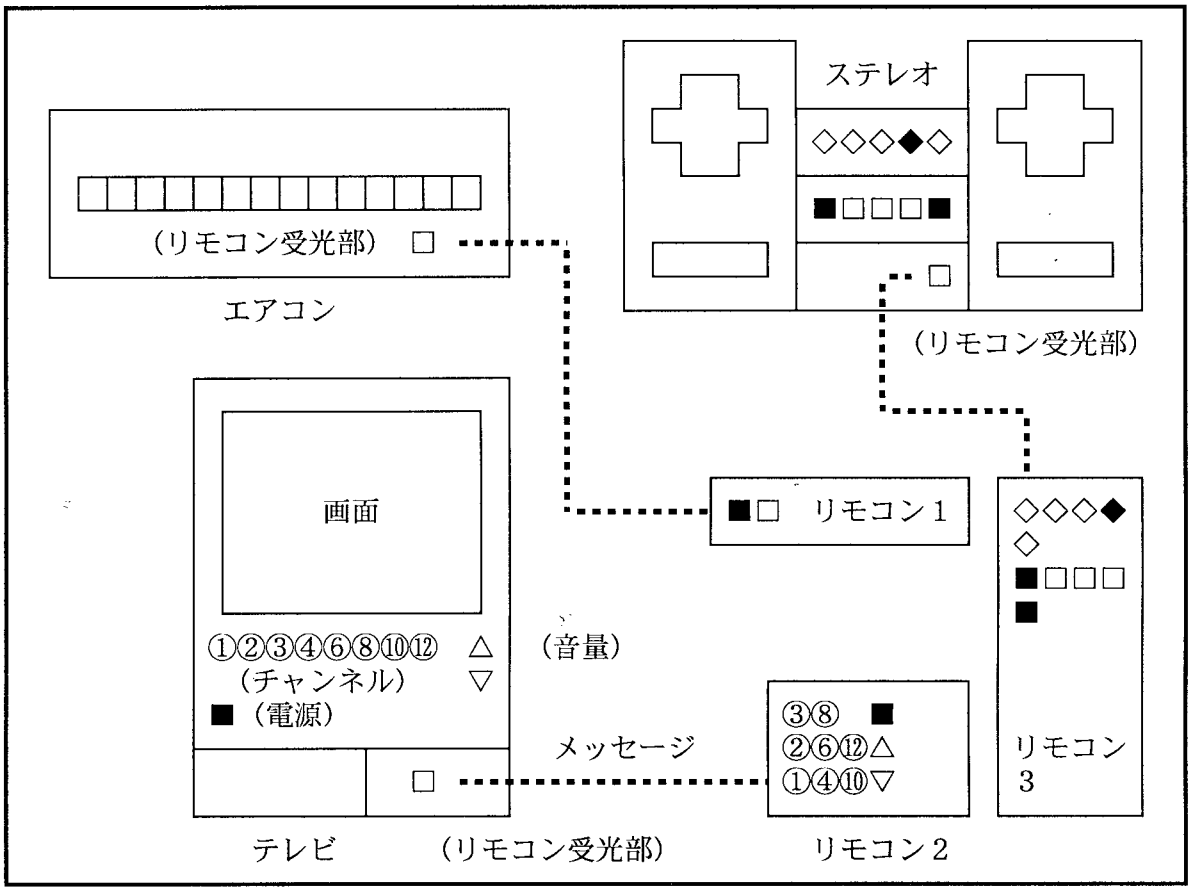
メッセージによって処理を行うようにプログラミングすることを、メッセージ駆動プログラミングと呼ぶ。これは、メッセージがプログラムを動かしているという意味であり、従来の手続き型プログラミングとは一線を画する考え方である。

ただし、これまでのプログラミングの主流であった手続き呼び出しの方法に比べて、メッセージを送る方法は、一般に余計なオーバヘッドがかかる。そのため、メッセージ駆動プログラミングの方が、手続き型プログラミングよりも多少処理が遅くなることが多い。

4 多様性

多様性とは、オブジェクト指向の「同じ名前をつけた部分がプログラムの中にいくつもあってもよく、しかも、それぞれ違う動作をするという仕掛け」のことである。

多様性を具体的な例で考えてみる。



図Ⅱ－4　リモコンによるエアコン、テレビ、ステレオの操作

例えば、室内にエアコンとテレビとステレオがあり、すべてリモコンで電源のオン・オフが操作できるものとする。この様子を、図Ⅱ－４に示す。

この場合エアコン用、テレビ用、ステレオ用にリモコンが３台ある。そして、全てのリモコンは形もボタンの数も異なる。しかし、３台とも、電源スイッチのオン・オフという機能は同じである。「電源スイッチをオンにせよ」又は「電源スイッチをオフにせよ」というメッセージは、それぞれのリモコンから、光というメッセージとなって、エアコン、テレビ、ステレオのリモコン受光部に送られる。

この考え方をソフトウェアの開発に適用したのが、オブジェクト指向でいう多様性である。これまでのプログラミング手法では、プログラムの中に大まかな働きは似ているが、細かい部分は異なるという機能が何カ所もある場合、それらにプログラムの中でぶつからないような、それぞれ異なる名前を付けなければならなかった。例えば、「エアコン電源オン」、「テレビ電源オン」、「ステレオ電源オン」といった名前である。

これに対して、オブジェクト指向では、プログラムの中で同じような機能があれば、同じ名前をつけることができる。そして、機能の区別は、その機能の対象を指定することによって行う。

例えば、エアコンに対する「電源オン」、テレビに対する「電源オン」、ステレオに対する「電源オン」という具合である。

このように、メッセージを送る対象がエアコン、テレビ、ステレオと異なる場合、同じ名前のメッセージを送っても多様な動作が行われる。このような性質を多様性と呼ぶわけである。これによって、似たような動作をする部分にいちいち違う名前を割り当てる必要がなくなり、プログラミングが簡単になるわけである。

5 クラスとオブジェクト

オブジェクトとは、オブジェクト指向の「プログラミングの対象となる世界に存在するもの」のことである。そしてクラスとは、オブジェクト（もの）を分類したものである。

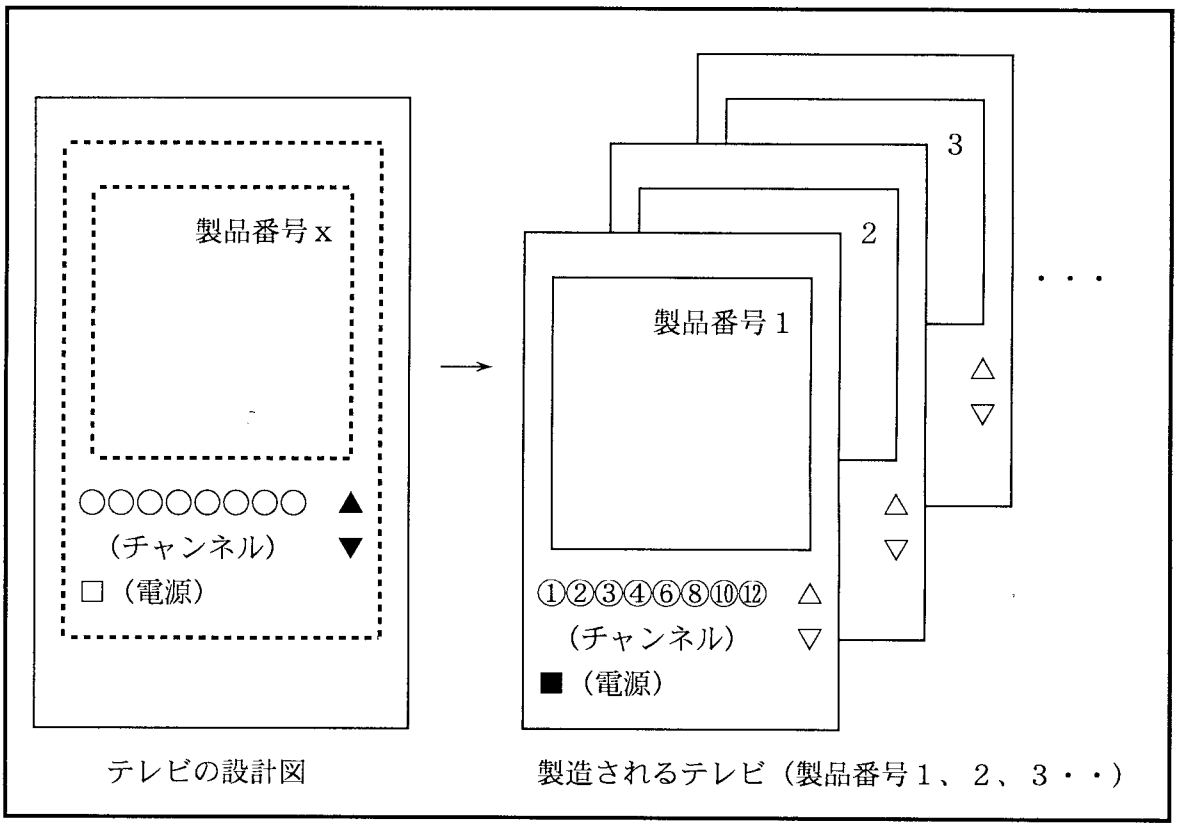
クラスとオブジェクトの関係は、クラスがオブジェクトの設計図、あるいは製造装置と考えることもできる。

クラスとオブジェクトを具体的な例で考えてみる。例えば、電気メーカは、同じ型式のテレビを何台も製造する。そして、ある型式のテレビを大量に製造するためには、必ず、その型式のテレビの設計図と製造装置をもっている。この様子を、図Ⅱ－５に示す。

この場合、テレビの設計図はクラス、それに従って製造されるテレビ１台１台は、オブジェクトと考えることができる。製造されるテレビは、製品番号を除いて全く同じものである。

この考え方をソフトウェアの開発に適用したのが、オブジェクト指向でいうクラスとオブジェクトである。オブジェクト指向においては、あらかじめクラスを設計し、そのクラスをもとに、プログラムの中でオブジェクトを製造し、思い通りの処理をさせるという手法が用いられる。もちろん、必要に応じてオブジェクトは複数個作られる。

クラスとオブジェクトは、オブジェクト指向の考え方のまさに真髄ともいうべき部分であり、これまで説明してきた「情報の抽象化」、「継承」、「メッセージ」、「多様性」といった基本的な仕掛けも、すべて、クラスとオブジェクトを対象にしている。



図Ⅱ－５　テレビの設計図と製造されるテレビ

ここで、クラスを設計するときに注意すべき点を考えてみる。それは、クラスを設計するときに、データそのものとデータを操作するための手続きを独立させることと、クラスの外部から不用意にクラス内部のデータを参照したり変更したりできないようにすることである。これは、情報の抽象化という意味から非常に重要である。

これまでのプログラミング手法では、内部データが密接に関係していたり、グローバル変数が存在するために、データ管理が困難であった。そのため、プログラム内部でのつながりが強く、プログラムの一部を取り出して再利用をすることが大変難かしかった。

オブジェクト指向プログラミング手法では、クラスの外部からは、クラスに対しデータ操作を依頼するためのメッセージを送る仕組みになっている。すなわち、外部からの内部データの直接参照や変更を許さないことにより、データ管理を強力に行うことができる。これにより、クラス間のつながりを断ち切ることが容易になるため、保守性が高く、かつ誤りも少ないソフトウェアを開発することができる。