連載●思考のおもちゃ箱　第4回

プログラミングで動く世界をつくる

フジタ未来経営研究所
リサーチフェロー
井庭 崇

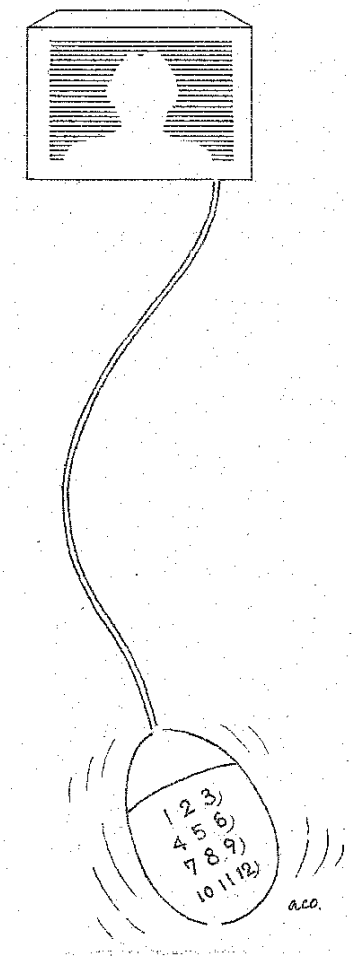


Illustration 巽 亜古

前回と前々回は（かなり間があいてしまいましたが）、オブジェクト指向によって世界を写し取るということを考えました。今回は、その写し取ったものを表現するための方法として、「プログラミング」を取り上げたいと思います。

一般的には、プログラミングは「コンピュータにさせる仕事の手順などを書くこと」だと説明されるのですが、その側面だけでは、プログラミングの面白さを十分に理解することはできません。そこで、本連載では、プログラミングを、「動く世界をつくるための方法」という視点でみていきたいと思います。そのような視点の方が、より本質を捉えていると思いますし、プログラミングが楽しくなるからです。

そのように考えると、「プログラムを書く」ということは、私たちが日常的に行なっている「文章を書く」ということに似ていることがわかります。日本語で文章を書いていろいろな「世界」を表現するのと同じように、プログラミング言語でプログラムを書けば、いろいろな「世界」を表現することができるのです。

Linuxの開発者であるリーナス・トーバルズは、プログラミングは「外から見るかぎりは、この世で一番退屈なものと映る」けれども、「プログラミングをやっている者にとって、それはこの世で一番面白いことだ」と語っています。それは、「自分が作った世界を経験させてくれ、どんなことができるのか教えてくれる」からだといいます。

以下では、技術的な作業として片付けられがちなプログラミングという行為について、その本質に迫ってみたいと思います。

動く世界をつくるための手順

コンピュータ上で動く世界をつくるためには、まずその世界がどのようなものかというモデルを作成する必要があります。表現したいものと関係のない部分は省略して、関係のある部分だけを写し取ることが重要です。

次に、この設計をもとに、プログラムをテキスト形式で記述していきます。このテキスト形式のプログラムを「ソースコード」といいます。そして、プログラムを書くという行為を「プログラミング」といい、記述に使う言語を総称し

て「プログラミング言語」といいます。プログラミングをするときに、私たちが普段使っている言語（例えば日本語）で書くことができればよいのですが、コンピュータは人間の言葉を理解することができないので、コンピュータと人間の両方が理解できる専用の言語を使って、世界を記述していくことになります。

ソースコードが一通り書き終えたら、そのソースコードを、コンピュータが直接理解できる言葉（機械語）に変換します（図1）。この変換したものが「実行可能プログラム」です。私たちが普段使っているワープロソフトや表計算ソフト、ブラウザなどは、どれもこの実行可能プログラムです。また、ゲームやシミュレーションというものも、実行可能プログラムということになります。

オブジェクト指向の世界をプログラミングしてみよう

前回と前々回は、世界をオブジェクト指向で写し取るということを考えました。今回は、このオブジェクト指向の世界モデルを、プログラムとして記述し、コンピュータ上で動かしたいと思います。オブジェクト指向では、世界の構成要素を「オブジェクト」として写し取り、それらオブジェクトの共通の属性や振る舞いを定義したものを「クラス」として定義する、ということを思い出してください。オブジェクト指向のプログラミングでは、このクラスをプログラムとして記述していきます。

今回は、「フジタ ミライ君という人間が、自分のテレビを見る」という例を通じて、世界モデルをプログラムとして記述することについて考えたいと思います。ここでは、ミライ君とテレビの属性と振る舞いは、図2のように定義されているとしましょう。以下では、この世界に登場する「テレビ」を最初にプログラミングし、次に「ミライ君」をプログラミングすることにします。

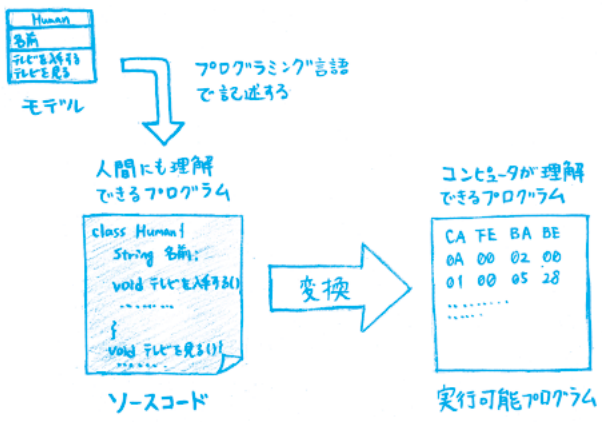
「テレビ」をプログラムとして記述する

まず、テレビのプログラムを書いてみたいと思います。ソースコード①を見てください。たったこれだけで、クラスをつくることができます。もちろんこれだけでは中身が何もありませんから、{}の中に、そのクラスの属性や振る舞いについて記述する必要があります。

Televisionクラスがどのような特性のクラスであるかは、すでに図2のところで考えてあります。Televisionクラスは、属性として『電源状態』と『チャンネル番号』をもっています。このことをプログラムとして表現すると、ソースコード②のようになります。

次に、Televisionクラスの振る舞いである『電源を入れる』、『電源を切る』、『チャンネルを変える』を、プログラムとして記述したいと思います。先ほどのプログラムに、これらの振

図1　ソースコードから実行可能ファイルへの変換



■ソースコード① Television.java

```
class Television {

}
```

※「Television」という名前のクラスを作るには、クラスを定義することを示すキーワード（class）の後に、定義したいクラス名「Television」を書きます。

■ソースコード② Television.java

```
class Television {
    String 電源状態="切";
    int チャンネル番号=1;
}
```

属性の定義

※ Stringというのは、その属性が「文字列」(string) で表現されていることを表し、intというのは「整数」(integer) で表現されていることを意味しています。

※製造された時点では、『電源状態』は「切」になっていて、『チャンネル番号』は「1」チャンネルになっていることにしたので、そのように設定するように書かれています。

る舞いを追加したのが、ソースコード③です。

以上で、Televisionクラスの定義を書き終えました。図2とソースコード③を比較すると、きちんと対応していることがわかると思います。クラス図として記述したモデルを、プログラムに変換しているのですから、当然といえば当然ですね。

「人間」をプログラムとして記述する

さて今度は、Humanクラスを定義しましょう。Humanクラスの『名前』と『持っているテレビ』という属性と、『テレビを入手する』と『テレビを見る』という振る舞いを書き込んだものが、ソースコード④です。ただし、『テレビを見る』の部分には、どのように見るかということは、まだ書いてありません。この部分は、後ほど書き込むことにしましょう。

世界の設定をプログラムとして記述する

以上で、この世界に登場するテレビと人間のクラスを記述することができました。しかし、これだけでは、登場人物の特徴を記述しただけで、この世界に何を登場させるのかということは定義していません。そこで、世界の設定もプログラミングする必要があります。

ここでは、世界の設定をObjectWorldという名前で定義することにします。このクラスは、最初に実行されるメインプログラムを含んでおり、その部分に、何を登場させるのかを記述します。ここでは、新しいテレビとミライ君を登場させるということと、ミライ君に新しいテレビを与え、『テレビを見る』ということを書いていきます。以上を記述したものが、ソースコード⑤です。

さあ、実行してみよう！

これまでに書いたプログラムを実行してみましょう。ただ、このままでは、何が起こっているのかは眼に見えるかたちではわかりません。プログラムの実行中にどのようなことが行なわれているのかを示すために、進行状況を表示する部分を追加することにしましょう。それが、ソースコード⑥です。

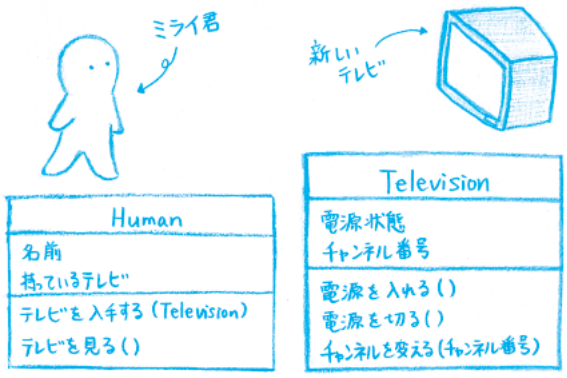
それでは、プログラムを実行してみることにしましょう。今度は、進行状況が表示されるので、プログラムが動いていることがよくわかります。実行の結果、次のように表示されます。

```
フジタ ミライの登場です！
フジタ ミライは、テレビを手に入れた。
フジタ ミライは、テレビを見ています…
```

さらに世界をつくり込んでいくと……

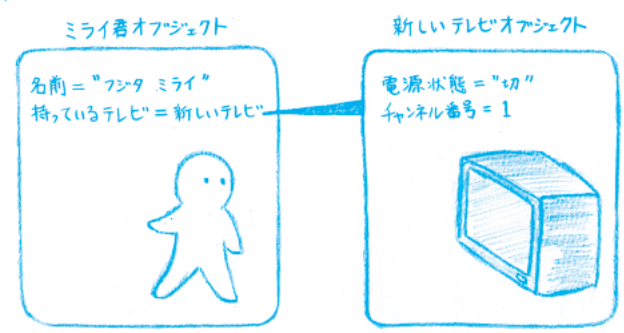
実行結果をみると、ミライ君オブジェクトとテレビオブジェクトが生成されて、無事動いているようです。だけど、ちょっと待って。どこ

図2　TelevisionクラスとHumanクラス



テレビ（Television）には、その属性として、現在の『電源状態』や『チャンネル番号』の状態をもっています。そして、テレビは、『電源を入れる』、『電源を切る』、『チャンネルを変える』という振る舞いが可能です。このテレビは、出荷時は『電源状態』が「切」になっていて、『チャンネル番号』が「1」チャンネルになっているとしましょう。人間（Human）には、『名前』があります。また、どのテレビを持っているのかを記憶するために『持っているテレビ』という属性も用意しておきます。そして、人間は『テレビを入手する』や『テレビを見る』という振る舞いが可能です。ここでは、「フジタ ミライ」という『名前』の人間をつくります。

図3　実行結果のイメージ



■ソースコード⑤ ObjectWorld.java

```
class ObjectWorld {

    public static void main(String[] args) {

        Television 新しいテレビ;
        Human ミライ君;

        新しいテレビ=new Television();
        ミライ君=new Human("フジタ　ミライ");

        ミライ君.テレビを入手する(新しいテレビ);
        ミライ君.テレビを見る();
    }
}
```

実行されたときに、ここが実行される

クラスの宣言

オブジェクトの生成

オブジェクトに振る舞うようにメッセージを送る

※プログラムが実行されたときに、まず実行されるのが、「public static void main (String[] args)」の部分です。なにやら難しそうに見えますが、細かいことは気にしないでよいでしょう。とにかく、プログラムを実行したときにここが実行される、とだけ覚えておけば十分です。
※まず最初の方で登場するオブジェクトを宣言します。そのとき、新しいテレビはTelevisionクラスのオブジェクトであり、ミライ君はHumanクラスのオブジェクトであることを明記します。
※実際にこの世界に登場させる（オブジェクトを生成する）には、「オブジェクト名＝new クラス名()」と書きます。つまり、「新しいテレビ」オブジェクトを生成するには、「新しいテレビ＝new Television()」と書きます。そして、「ミライ君」オブジェクトを生成するには、人間クラスでは名前を指定する必要があるので、括弧内に名前を書きます（Humanクラスのコンストラクタの部分で、そういうふうに決めたからです）。
※ミライ君に新しいテレビを与えるのは、「ミライ君.テレビを入手する（新しいテレビ）」で実現できます。このように、他のオブジェクトに何かをしてほしい場合には、「オブジェクト名.メソッド名（引数）」というふうに書きます。同様に、ミライ君に『テレビを見る』ことをしてもらうために、「ミライ君.テレビを見る()」を書きました。

■ソースコード⑥ Human.java

```
class Human {

  String 名前;
  Television 持っているテレビ;


  void テレビを入手する(Television 手に入れたテレビ){
      持っているテレビ=手に入れたテレビ;
      System.out.println(名前+"は、テレビを手に入れた。");
  }

  void テレビを見る(){
      System.out.println(名前+"は、テレビを見ています…");
  }


  Human(String 指定名前){
      名前＝指定名前;
      System.out.println(名前+"の登場です！");
  }
}
```

※System.out.println( )という部分は、そこに書いてある文章を、私たちに見えるかたちで表示するという命令文です。

■ソースコード③ Television.java

```
class Television {
    String 電源状態="切";
    int チャンネル番号=1;

    void 電源を入れる(){
      電源状態="入";
    }

    void 電源を切る(){
      電源状態="切";
    }

    void チャンネルを変える(int 指定チャンネル){
      チャンネル番号=指定チャンネル;
    }
}
```

属性の定義

振る舞いの定義

※『電源を入れる』というところには、電源を入れると『電源状態』が「入」になる、ということが書かれています。『電源を切る』の部分は、電源を切ると『電源状態』が「切」になる、ということが書かれています。『チャンネルを変える』のところでは、どのチャンネルに変えるのかを指定する必要があるので、括弧内で「int 指定チャンネル」という指定ができるようになっています。
※voidというのは、この振る舞いをした後に、戻り値がない（void：空）ということを意味しています。

■ソースコード④ Human.java

```
class Human {
    String 名前;
    Television 持っているテレビ;

    void テレビを入手する(Television 手に入れたテレビ){
        持っているテレビ=手に入れたテレビ;
    }

    void テレビを見る(){
    }

    Human(String 指定名前){
        名前=指定名前;
    }
}
```

属性の定義

振る舞いの定義

コンストラクタ（オブジェクトを生成するためのもの）

※このプログラムでは、『持っているテレビ』や「手に入れたテレビ」は、Televisionクラスのオブジェクトなので、Televisionという型で指定されています。
※『テレビを入手する』という振る舞いでは、「手に入れたテレビ」を『持っているテレビ』として記憶します。
※Human（String 指定名前）の部分は、コンストラクタといって、オブジェクトを生成するときに、どのようにオブジェクトを初期設定するのかを記述する部分です。このプログラムでは、オブジェクトを作成するときに名前を指定すれば、その名前をもつオブジェクトをつくってくれる、ということが書かれています。

か変だと思いませんか？　プログラムは正しく動いているのですが、その実行された世界の状況が少し奇妙です。というのは、テレビは、最初『電源状態』が「切」の状態なので、このプログラムのままでは、ミライ君は、何も映っていないテレビを見ていることになってしまうのです。

　この奇妙な状況を解消するには、『テレビを見る』時は必ず電源を入れることにすればよいでしょう。その部分を、Humanクラスのソースコードに書き足すことにします。ついでに、チャンネルも変えて、他のチャンネルを見るようにしたいと思います。これらを追加したものが、ソースコード⑦です。また、テレビの方も、現在の状況を表示するようにしたいので、ソースコード⑧のように変更します。実行結果は、次のようになります。

フジタ ミライの登場です！
フジタ ミライは、テレビを手に入れた。
フジタ ミライは、テレビをつけます。ポチっ。
テレビがつきました。
いま、1チャンネルが流れています。
フジタ ミライが、テレビのチャンネルを変えます。ポチっ。
テレビのチャンネルが変わりました。
いま、8チャンネルが流れています。
フジタ ミライは、テレビを見ています…

　コンピュータ上で、「フジタ ミライ」というヴァーチャルな人間が、ヴァーチャルに自分のテレビのスイッチを入れ、チャンネルを変えるということが実現しました。これが、コンピュータ上

■ソースコード⑦ Human.java

```
class Human {

    String 名前;
    Television 持っているテレビ;


    void テレビを入手する(Television 手に入れたテレビ){
        持っているテレビ=手に入れたテレビ;
        System.out.println(名前+"は、テレビを手に入れた。");
    }

    void テレビを見る(){
        System.out.println(名前+"は、テレビをつけます。ポチっ。");
        持っているテレビ.電源を入れる();

        System.out.println(名前+"が、テレビのチャンネルを変えます。ポチっ。");
        持っているテレビ.チャンネルを変える(8);

        System.out.println(名前+"は、テレビを見ています…");
    }


    Human(String 指定名前){
      名前=指定名前;
      System.out.println(名前+"の登場です！");
    }
}
```

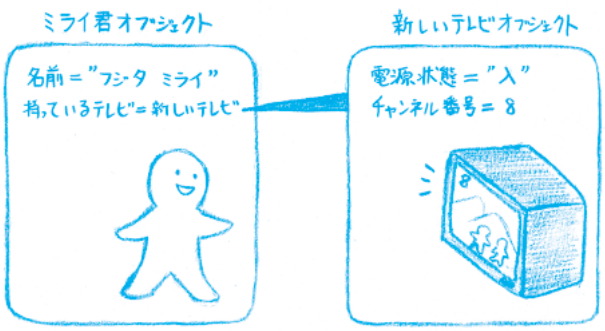
追加分

※テレビの電源を入れるということは、「持っているテレビ.電源を入れる()」で実現できます。

■ソースコード⑧ Television.java

```
class Television {

    String 電源状態="切";
    int チャンネル番号=1;

    void 電源を入れる(){
        電源状態＝"入";
        System.out.println("テレビがつきました。");
        System.out.println("いま、"+チャンネル番号+"チャンネルが流れています。");
    }

    void 電源を切る(){
        電源状態＝"切";
        System.out.println("テレビが消えました。");
    }

    void チャンネルを変える(int 指定チャンネル){
        チャンネル番号=指定チャンネル;
        System.out.println("テレビのチャンネルが変わりました。");
        System.out.println("いま、"+チャンネル番号+"チャンネルが流れています。");
    }
}
```

図4　実行結果のイメージ



に動く世界をつくる、ということなのです。

さらにどんどん作りこんでいくことによって、もっと複雑なものをつくることも可能です。例えば、テレビクラスに『音量』という属性をつくって、音量を大きくしたり小さくしたりできるようにしたり、チャンネルや音量を、具体的な数値を指定するのではなく、＋1、－1というように調整できるようにすることもできるでしょう。ミライ君の方も、曜日や時間帯に応じて自分の観たいチャンネルを選ぶ、というのもよいかもしれません。

今回の例では、「テレビを見る」ということを表現しましたが、社会や経済に関するモデルのプログラムをつくりこんでいけば、現実の社会・経済に近い世界をコンピュータ上に再現することができるでしょう。そう、それこそが、コンピュータ・シミュレーションによって社会・経済を表現するということなのです。

プログラムという知的構築物

今回は、プログラミングによって「動く世界をつくる」ということをみてきました。コンピュータ上に世界をつくるということの一番わかりやすい例は、コンピュータ・ゲームでしょう。そこでは、ゲーム・プレイヤーが主人公となって旅する架空世界や、サッカーや野球の試合が行なわれるヴァーチャルな世界などがつくられます。

しかし、コンピュータ上に世界をつくるというのは、ゲームに限られたことではありません。どんなソフトウェアでも、プログラミングによってそのソフトウェアごとの「世界」がつくられているのです。普段使っているワープロソフトも表計算ソフトも、もっと言えば、コンピュータのOS（オペレーティング・システム）も、すべてこのようにプログラミングされた知的構築物としての「世界」だということができます。

最初にも紹介したリーナス・トーバルズは、プログラミングは「自分で世界を作る」ことだといい、次のように語っています。「コンピュータという世界の中では、君は造物主だ。君はその中で起こるすべてのことを支配する。君がそれなりに有能なら、神になることも可能だ―ちっぽけな世界の神様だけど。」

プログラミングについての話題では、それによって生み出されるソフトウェアの便利さや有効性が強調されることが多いのですが、その行為自体の楽しさも、注目に値します。エリック・レイモンドの『ハッカーズ大辞典』では、プログラミングは「服を着ているときに味わえる最高の楽しみ（ただし衣服は必須条件ではない）」と紹介されているくらいです。

このように、動く世界を記述し、表現できるプログラミングは、私たちが本連載で模索している「未来をデザインし伝達するための道具」として、かなり有効なものだと言えるでしょう。プログラミングについての入門書やツールを入手して、実際にプログラミングの楽しみを味わってみてはいかがでしょうか。

References

■Javaの入門書としては、以下のものがおすすめです。

●高橋麻奈、『やさしいJava』、第2版、ソフトバンクパブリッシング、2002

■Java言語による統合プログラミング環境としては、最近は「eclipse」がおすすめです。eclipseは、インターネット上で無料で入手できます。

●http://www.eclipse.org/

■引用

●リーナス・トーバルズ、デイビット・ダイヤモンド、『それがぼくには楽しかったから』、風見潤 訳、中島洋 監修、小学館プロダクション、2001

●Eric S. Raymond、『ハッカーズ大辞典』、改訂新版、ASCII、2002

いば・たかし

1974年神奈川県生まれ。現在、千葉商科大学 政策情報学部 専任教員（助手）。著書に『複雑系入門』（共著、NTT出版）など。

E-mail: iba@sfc.keio.ac.jp

［ひとこと］翻訳を手がけた本が出版されました。書店で見かけたら、ぜひ手に取ってみてください。『社会シミュレーションの技法』（ナイジェル・ギルバート、クラウス・G・トロイチュ著、日本評論社、2003年）。詳しくは、本号の巻末にある「FIFだより」をご覧ください。