第2章

オブジェクトの作成とメソッド呼び出し

本章では，オブジェクトと呼ばれる“もの”を作成することと，“もの”に対してメソッドを呼び出すことという，オブジェクト指向の基本について学ぶ。

2.1 オブジェクトとクラス

オブジェクト指向プログラミングは，オブジェクトと呼ばれる“もの”を作成すること（new 演算子）と，“もの”に依頼を行うこと（メソッド呼び出し）を基本としたプログラミングの方法である。

オブジェクトを作成するには，それがどんなオブジェクトなのか定義したものが必要である。そのような定義は，**クラス**[1]と呼ばれている。クラスは，システムに最初から用意されているものもあるし，自分で定義するものもある。1.4 節で述べたように，クラスの内容は，“クラス名.class”という名前のファイル（**クラスファイル**）に格納されている。つまり，1.4 節で作成した 4 つのファイルは，それぞれ TurtleFrame，Turtle，(TurtleFrame$)Line，TurtlePanel というクラスを定義したクラスファイルである。Turtle は画面上を動き回るタートル（亀），TurtleFrame はタートルグラフィックスの表示画面，Line はタートルの動作によって描かれる線分，TurtlePanel は TurtleFrame の画面の中のタートルが表示される白い矩形部分という“もの”をそれぞれ意味している。このうち Line クラスは TurtleFrame クラスの内部クラス[2]で，ユーザ[3]からは利用できない。また，TurtlePanel クラスも本書前半ではユーザプログラムから直接扱うことはない。TurtleFrame クラスと Turtle クラスを用いたプログラムを作成しながら，Java 言語の説明を始めよう。

[1] このクラスの説明は本質的ではあるが，単純化しすぎている。詳細は，3 章，6 章で説明する。

[2] 内部クラスは，あるクラスの機能を実現するためにそのクラス内に定義されるクラス（6.8 節参照）。A というクラスの内部クラス B は，A$B.class というファイル名となる。

[3] クラスを利用したプログラムの作成者を，クラスのユーザという。

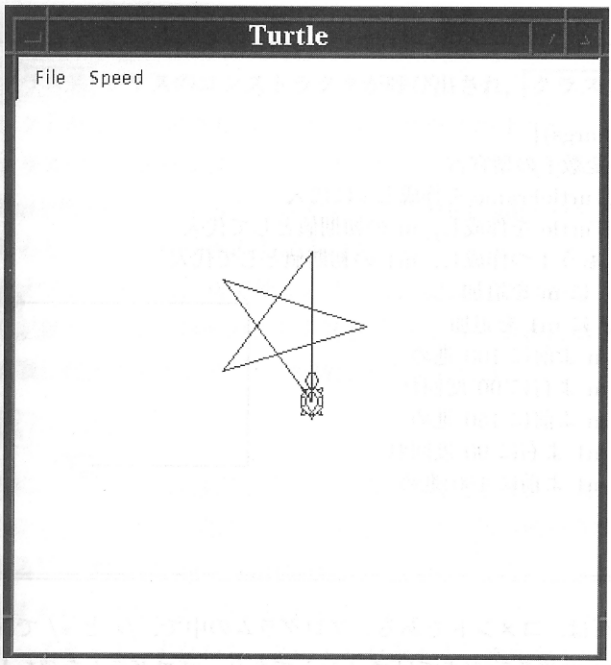


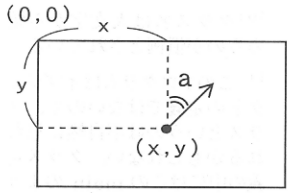
図 2.1 タートルグラフィックで描いた星型

2.2 タートルグラフィックス

タートルグラフィックスは，画面上に置かれたタートルに対して「前に 100 進め」とか「右に 60 度回れ」といった命令を送ることによりタートルを動かし，その軌跡として線を描く，プログラムによって線画を描く方法である。タートルの最初の位置や向きを指定するのに画面の座標系が必要である。ここでは，左上を原点として，右に X 軸，下に Y 軸をとることにする。また，角度は，上向きを 0 度として右回りに度数で表すことにする[4)5)]。

図 2.1 は，400×400 の大きさのフレーム（ウィンドウ）を開き，(200,200) に 0 度の方向に配置されたタートルに対して “前に 100 進め” と “左に 144 度回れ” という命令を 5 回繰り返すことによって描いた絵である。

4)



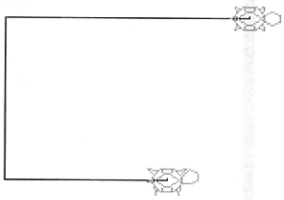
5) このタートルグラフィックスでは，座標や角度は整数値のみを考えることにする。

2.3 最初の例題 — オブジェクトの生成とメソッド呼び出し —

リスト 2.1 は，TurtleFrame と Turtle を用いて，絵を描くプログラムであり，T21.java という名前のファイルにおさめられている。

リスト 2.1 *T21.java*

```
1  /** 最初のプログラムの例 */
2  public class T21 {
3    public static void main(String[] args){
4      TurtleFrame f;              // 変数 f の型宣言
5      f =  new TurtleFrame();     // TurtleFrame を作成し f に代入
6      Turtle m = new Turtle();    // Turtle を作成し，m の初期値として代入
7      Turtle m1 = new Turtle();   // もう 1 つ作成し，m1 の初期値として代入
8      f.add(m);                   // f に m を追加
9      f.add(m1);                  // f に m1 を追加
10     m.fd(100);                  // m よ前に 100 進め
11     m.rt(90);                   // m よ右に 90 度回れ
12     m.fd(150);                  // m よ前に 150 進め
13     m1.rt(90);                  // m1 よ右に 90 度回れ
14     m1.fd(100);                 // m1 よ前に 150 進め
15   }
16 }
```



最初の行は，**コメント**である。プログラムの中で，/* と */ で囲まれた部分[6]や，// の後ろ行末まではコメントであり，プログラムを読む人のために書かれた説明文である。また，3 行目以降は，行の先頭をインデント（字下げ）するために空白を入れているが，これも，プログラムを読む人が見やすくするための工夫である。Java では，プログラム中の改行，空白は，区切りであって，どれだけ入っていても意味は変わらない[7]。インデントやコメントをどう書くかは自由だが，プログラムは動けばいいというだけではない。デバッグ[8]や保守[9]のことも考えて，読みやすいプログラムも心掛けてほしい。

2 行目は，行末の { からそれに対応する 16 行目の } までが T21 という名前[10]のクラスの定義だと宣言している[11]。T21 という文字列は，このプログラムを作成している人が自由に選んでつけた名前なのに対し，public や class という文字列は，Java 言語の一部をなしている。このような文字列のことを，**予約語**という[12]。3 行目の main は，このプログラムの実行が開始されると，3 行目の最後の { と 15 行目の } で挟まれた部分に書かれた内容が順に実行されることを示している。2，3 行目の詳細は 6 章で説明する。

5 行目はオブジェクトを作成している。オブジェクトを生成するための手続きのことを，**コンストラクタ**と呼ぶ。

[6] /** で始まるコメントは，6.2 節で述べるように，特別な意味をもつ。

[7] 例えば，2 行目は，
␣␣␣␣public␣␣class
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣T21
{
と 4 行に分けられていても同じ意味である（␣ は空白）。

[8] バグを修正すること。

[9] 5.1 節の傍注参照。

[10] クラス名は大文字から始めるのが慣例とされている。

[11] このプログラムはオブジェクトの定義ではないので，クラスというのは不自然に思われるかもしれない。クラス定義の中にはこの main のように，オブジェクト定義以外のプログラムも含めることができ，このクラスは，それだけからなっている。詳細は，6.6 節で述べる。

[12] 本書では，読みやすさを尊重して，プログラムリスト中の予約語はボールド体で示した。また，多少見にくいが，3 行目の [] は [ ] の 2 文字である。

new クラス名()

により，クラス名 クラスのコンストラクタが呼び出され，クラス名 クラスのオブジェクトが 1 つ生成される[13][14]。あるクラスのオブジェクトのことを，そのクラスの**インスタンス**という。よって，ここで生成されたオブジェクトは，TurtleFrame のインスタンスである。TurtleFrame のインスタンスが作成されると，すぐに枠が画面上に表示される[15]。

生成したオブジェクトは，後で使えるように，記録しておかなくてはならない。値を記録するための場所のことを**変数**[16]という。変数に値を記録することを，変数に**代入**するという[17]。変数に代入するには，

変数名 = 式;

という具合に，= の左辺に変数を，右辺に代入する値を表す式を書いて，最後にセミコロン (;) をつける。**式**については後述するが，何か値を意味する表現のことである[18]。例えば，new TurtleFrame() は作成したオブジェクトという値を意味する式である。よって，以上をまとめると，5 行目は TurtleFrame クラスのインスタンス，すなわち，タートルを表示する画面を 1 つ作成して，f という変数に記録することになる。最後のセミコロン (;) は，ここまででコンピュータに対する 1 つの命令になっていることを示している。このような命令の単位を**文**という。5 行目から 14 行目のそれぞれの行は，文となっている。

変数[19]は，プログラム中に最初に現れる前に，そこに記録される値の種類を指定しておく必要がある。そのような指定を，変数の**型宣言**という。型宣言は，

型名 変数名;

という形で行われる。**型**についてもあとで述べる。ここでは，とりあえずクラスなどのことと思っておいてほしい。この例では，4 行目で f に格納できるのは TurtleFrame 型のオブジェクトだと宣言している。

変数の型宣言のときには，

型名 変数名 = 初期値を表す式;

という形で，その変数の初期値を同時に代入することもできる。よって，この 2 行は，

```
TurtleFrame f = new TurtleFrame();
```

と 1 行で書いてもよい。6 行目は初期値つきの変数宣言で，タートルを作成して m という変数に代入している[20]。7 行目も，タートルをもう 1 つ作って m1 という変数に代入している。型宣言は，同じ型の変数がたくさんある

[13] このコンストラクタの説明で，new や () はプログラムの中にそのまま書かれるものである。それに対して，クラス名 は，特定の 1 つの文字列を指しているのではなく，クラス名となっている文字列なら何でもここにこれることを示している。そのようなものを，文字列の上を動く変数という意味で，メタ変数という。この本では，メタ変数は四角で囲んで示すことにする。

[14] この new のことを，new 演算子という。

[15] そのように，ここで呼び出されるコンストラクタは作られている。

[16] 変数名には，数字以外の文字から始まる文字列を用いることができる。ただし，予約語はだめである。また，変数名は，小文字から始めるのが慣例とされている。

[17] 変数への代入は 3.1 節で詳しく扱う。

[18] 値についても後述する。

[19] 3.3 節で述べるように，変数にはいろいろな種類がある。ここでいう変数は，正確には，**ローカル変数**と呼ばれるものである。

[20] Turtle は，8 行目で TurtleFrame に貼り付けられるまで表示されない。

ときは，

```
Turtle m, m1;
```

と一度に行うこともできるし，

```
Turtle m = ..., m1 = ...;
```

と，一度に初期値つきで行うこともできる。

8行目からは，**インスタンスメソッド**[21]の呼び出しである。8行目は，f（に代入されていたTurtleFrame）に対して，addというインスタンスメソッドを，m（に代入されている，最初に作ったTurtleオブジェクト）を引数として呼び出している。**引数**[22]とは，コンストラクタやメソッドを呼び出すときに一緒に与える値のことである。オブジェクトに対するインスタンスメソッドの呼び出し[23]の一般形は，以下のようになる。

オブジェクト式.インスタンスメソッド名(引数式1,...,引数式n)

引数が2つ以上あるときには，このようにコンマ(,)で区切って並べられる。オブジェクト式や引数式の所には式を書くことができ，それらの式の意味している値がメソッド呼び出しの対象となるオブジェクトや，メソッドに引数として渡される値となる。変数も式であり，その変数に代入されている値が式の意味する値となる[24]。

インスタンスメソッド呼び出しが行われると，呼び出されたオブジェクトは，そのメソッドに応じた処理を行う。TurtleFrameクラスのオブジェクトのaddメソッドの場合には，引数として与えられたTurtleを自分の画面上に載せる動作をする。よって，この1行を実行することによって，5行目で作成した画面上に亀の絵が現れる。9行目も同じである。タートルには，それぞれ座標，向き，色などの状態がある。mとm1は，標準の初期値である，画面中央の(200,200)という座標と0度の角度，黒色という状態で作成されている。

10行目は，今度はmに対してfdというメソッドを呼び出して[25]，mに前に100だけ進むように指令している。これにより，mは移動し，動いた跡が画面に線として表示される。11行目では，rtというメソッドで，mが右に90度向きを変えるように指令している。これにより，mは角度を90度増やし，それに応じて画面表示も変化する。以下同様である。これによって，リスト2.1の右に示した絵が描ける。

2.4　コンパイルと実行

このプログラムをコンパイルして実行してみよう。Javaのソースプログラムのファイル名は，クラス名と一致させて，“クラス名.java”という名前にす

21) メソッドにはインスタンスメソッドとクラスメソッドがある。クラスメソッドについては，次章で扱う。文脈から明らかなときには，インスタンスメソッドのことを単にメソッドという。

22) ヒキスウと読む。

23) オブジェクトのインスタンスメソッドの呼び出しのことを，オブジェクトに**メッセージ**を送るという言い方もする。

24) 変数に代入されている値を使うことを**変数**を参照するという。変数への代入や参照のことを変数への**アクセス**という。

25) このようなときに，「mにfdというメッセージを送る」という表現は，mに処理を依頼しているイメージが強くて，オブジェクト指向の考え方をよく表現している。