覧が現れる。その中からどれかのエントリーを選ぶと，それに関する記述が右の大きなフレームに現れる。

練習問題 2.3： API 仕様のドキュメントにアクセスできる人は，実際に，ブラウザをあけ，java.awt.Color クラスの API の仕様を見てみよう。void Color(int r, int g, int b) という形のコンストラクタの説明を探してみよう。

このように，自分で一から作るのではなく，すでに用意されているクラスを用いてプログラムを組むのがオブジェクト指向プログラムの基本的な考え方である。よって，Java 言語の文法を知るだけではなく，どんなクラスライブラリが利用可能で，どのようなクラスが存在し，それらはどういうコンストラクタやメソッドをもっているのかを知っておくこと[39]は，とても重要である。本書も，後半は，Swing などの標準ライブラリ[40]の利用法の解説が中心となる。

プログラムの説明に戻る。7 行目のこのコンストラクタ呼び出しは，3 つの int 型の引数をとっている。ブラウザで見た仕様にも書いてあるように（図 2.2 参照），これらの引数は，色の r（赤），g（緑），b（青）成分の量を 0 から 255 までの数として表している。0,0,0 なら黒，255,255,255 なら白であり，255,0,0 は赤ということになる。よって，この行で作られるのは赤色に対応する Color オブジェクトであり，それが変数 c に代入される。

次の行で，c の値は m1 に対する setColor というメソッド呼び出しの引数として渡される[41]。ところで，メソッドの引数の所に書けるのは式であり，コンストラクタの呼び出しも新たに作られたオブジェクトを意味する式であった。よって，7 行目と 8 行目は，まとめて，

```
m1.setColor(new java.awt.Color(255,0,0));
```

と書いてもよいことになる。

9 行目以降は，T21.java と同様に f，m，m1 に対してメソッド呼び出しを行っている。途中，15 行目で，d = d/2; と d に d/2 代入している[42]。代入は，等号の右辺の式の値を求めて，それを変数に代入するのであった。いま，d の値は 100 なので，d/2 という式の値は 50 である。よって，この行で，d に代入されている値は 50 に変わる。これにより，16，17 行目は 11，12 行目と同じことが書かれているが，一方は 100 進み，もう一方は 50 進むという具合に動作が異なる。また，17 行目の代りに，

```
m = m1;
m.fd(d);
```

と書いたとしても同じ動作をする[43]。その際，16 行目もこの最後の行も，m の fd というメソッドを起動するという同じ形をしているが，m という式の意

[39] あるいは，自分で調べられるように，API の仕様の見方に慣れておくこと。

[40] 標準ライブラリは，すべての実行環境で利用可能である。他の自分がダウンロードしてきたライブラリは，利用者がもっている保障はなく，それを利用して作成したプログラムは，実行環境に依存したものとなる。

[41] setColor の意味は，2.5 節の Turtle の仕様を見て頂きたい。

[42] これを，d と d/2 は等しいという意味（ということは，d は 0?）に勘違いしないように。Java の = は，等号ではなく，変数への代入記号である。

[43] この代入により，m と m1 は同じタートルを意味することになる。詳しくは，3.1 節参照。

味しているものが違うので，違った動きとなる。このように，同じことを書いても，そのときの変数の値によって違った意味となることに注意されたい。

このプログラムは，x, y, d という変数を用いているが，途中の x + d などの式をすべてその値である定数[44]に置き換えても同じ動作をする。変数を用いた意義は，4 行目の x, y, d の値を変えるだけで，場所や縮尺の違った同じ図形が描けることである。もし，6 行目が new Turtle(300,300,0) と書かれていたら，なぜ 300 なのかわかりにくいし，1 辺の長さを 150 に変える必要が出ても，どことどこを書き換えたらいいかわかりにくくなる。後で読んだり書き換えたりするときのために，途中に現れる定数はできるだけ減らして，依存関係が明確になるようにプログラムは書くべきである。

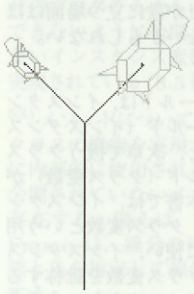
[44] 個々の値の，文字列としての表現を**リテラル**と呼ぶ。ここも，正確にはその値を示すリテラルと言ったほうが正しい。

練習問題 2.4 : T22.java をコンパイルし，実行してみよう。また，このプログラムを変更して，左上の座標が (50,100)，1 辺が 200 で同じ絵が描けるようにしてみよう。

2.7 さらにもう1つの例 — 値を返すメソッドとインスタンス変数 —

リスト 2.3 *T23.java*

```
1  import java.awt.Color;
2  public class T23 {
3    public static void main(String[] args){
4      int d = 100, x, y, a;
5      TurtleFrame f = new TurtleFrame();
6      Turtle m = new Turtle(200,300,0);
7      f.add(m);
8      m.fd(d);
9      x = m.getX();                    // m の X 座標のとり出し
10     y = m.getY();                    // m の Y 座標のとり出し
11     a = m.getAngle() − 45;           // m の角度のとり出し
12     Turtle m1 = new Turtle(x, y, a); // m1 の作成
13     f.add(m1);
14     m.kameColor = new Color(0,255,255);        // m の亀の色を水色に変える
15     m.kameScale = m.kameScale ∗ 2; // m の亀を現在の 2 倍の大きさにする
16     m.rt(45);
17     d = d / 2;
18     m.fd(d);
19     m1.fd(d);
20     }
21 }
```



もう一度，2.5 節の Turtle の仕様を見て頂きたい。各メソッド名の先頭に

は，void とか int とか書かれている。これらは，それぞれのメソッドが返す値の型を表現している。メソッドは値を返すことができる。m.getX() といったメソッド呼び出しは式であり，返された値を意味している。例えば，仕様には getX メソッドは int を返すと書かれている。よって，9 行目の m.getX() は int 型の式となり，その表す値は，m（に代入されているタートルオブジェクト）が getX の呼び出しに対して返す値である[45]。10 行目も 11 行目も同様である。式については次章で改めて扱うが，2 つの int 型の式 式1 と 式2 に対し，式1 + 式2 や 式1 − 式2 といった数式も式であり，その値は 式1 と 式2 の値の和や差である。よって，11 行目の右辺も式となる。12 行目では，これらの値を元に m1 を作成している。ここまでの式に関する説明で，9 行目から 12 行目までをまとめて，

```
Turtle m1 = new Turtle(m.getX(), m.getY(), m.getAngle() − 45);
```

と書いても同じ意味になることもおわかりだろう。

仕様の中で，void は値を返さないことを示す特別な表記である。返値の型の所に void と書かれているメソッドは値を返さないし，これらのメソッドの呼び出し式は値をもたない。getX メソッドは，オブジェクトの状態に変化を与えず値を返すだけだが，メソッドは，何らかの動作をして，しかも結果の値を返すこともできる。例えば，両方の moveTo メソッドは指定された座標までタートルを移動させるが，それと同時に，移動した距離を返すことになっている[46]。返された値は必ずしも使う必要はない。つまり，moveTo (100, 200,0); といった呼び出しをしてもよい。

さて，もう一度 Turtle の仕様を見て頂きたい。コンストラクタ，メソッドと並んで，**フィールド**という項目があり，java.awt.Color kameColor，および，double kameScale と書かれている[47]。これは，Turtle には kameColor と kameScale という名前のインスタンス変数があって，それらの型が java.awt.Color および double[48]であるということを示している[49]。**インスタンス変数**はそれぞれのオブジェクトがもつ変数であり，**インスタンスフィールド**とも呼ばれる。main の中で宣言されている変数（ローカル変数）と同じように，代入することもできるし，式として使うこともできる。違いは，これがオブジェクトの一部であり，オブジェクトの状態を表しているということである。これらのインスタンス変数の値を変化させることによって，Turtle オブジェクトの動作に影響を与えることができる。

あるオブジェクトのインスタンス変数は，

オブジェクト式.インスタンス変数名

という表現でアクセスできる。14 行目は，m のもつ kameColor インスタンス変数に代入を行い，新しく生成した色オブジェクト（水色に相当する）に

[45] どういう値を返すかは，2.5 節の Turtle の仕様を見よ。

[46] 値を返すメソッドの例がほしくてこういう仕様にしたが，実際に moveTo が値を返すことが役に立つ場面はほとんどないかもしれない。

[47] フィールドは，インスタンスフィールド（インスタンス変数）と，次章で扱うクラスフィールド（クラス変数）がある。本書では，インスタンス変数，クラス変数という用語を主に使い，インスタンス変数とクラス変数を総称するときにフィールドという言葉を使うことにする。

[48] 3.2 節に述べるが，double というのは浮動小数点数（実数の近似値）を表すプリミティブ型である。

[49] 最後にある withKameAll については 3.3 節で扱う。

設定している。また，15 行目は，m のもつ kameScale インスタンス変数の値に対してその 2 倍を計算して再び m の kameScale インスタンス変数に代入している。すなわち，現在の値（0.4 のはずである）の 2 倍に変更している。これにより，m の亀の表示の色が水色になり，大きさが 2 倍になる[50]。m の状態を変えただけで，m1 の状態は変化しないことに注意されたい。インスタンス変数は，個々のオブジェクトがもっているのであり，あるクラスのインスタンスすべてに影響を与えるものではない。

[50] 実は，このようなインスタンス変数の利用は，決して推奨されていない（6.4 節参照）。

練習問題 2.5： 2.5 節の仕様を見ながら，TurtleFrame や Turtle に用意されている，様々なコンストラクタやメソッドを用いた絵を描こう。ファイル名は何でもよい。変数の初期値を変えるだけで絵の大きさや場所が変わるように工夫しよう[51]。

[51] 複雑な絵を書くのに必要な制御構造などは次章以降で学ぶので，ここでは凝ったものを作る必要はない。

2.8 オブジェクトとは？ — 再び

ここまでの説明で，オブジェクトとはどういう “もの” か，雰囲気をつかんで頂けたと思う。まず，オブジェクトは内部に状態をもっている。仕様に書かれたインスタンス変数の値はもちろんであるが，それ以外にも，様々な状態をもっている。例えば Turtle は，現在の x, y 座標と角度，色，ペンを下ろしているか上げているか，どの TurtleFrame の上に乗っているかといった状態をもっている。また，TurtleFrame は，今どのタートルが上に乗っているかとか，どんな線が描かれているかといった状態をもっている[52]。

次に，オブジェクトはインスタンスメソッドの呼び出しを受けることができる[53]。メソッドの中でオブジェクトが行う処理は，大まかに言って，2 つである。1 つは，自分の状態を変えること。例えば，Turtle は fd というメソッドの中で，自分の x, y 座標という状態を変化させる。もう 1 つは，（自分自身を含む）他のオブジェクトのメソッドを呼び出すこと。Turtle が fd によって移動した後，その軌跡が線として画面に残るが，これは，実は，Turtle が fd メソッドの中で，自分が乗っている TurtleFrame に対して，一般には公開されていないメソッド（傍注参照）を呼び出して新しい線分を追加するように依頼することにより実現されている。

このように，**オブジェクト**は内部状態をもっており，メソッドを呼び出されることにより自分の内部状態を変化させたり，他のオブジェクトのメソッドを呼び出したりするものである[54]。このようなものの集まりとして，対象をモデル化しながらプログラムを記述するのが，オブジェクト指向のプログラムの基本的な考え方である。

この例題プログラムを見て，タートルグラフィックスのプログラムを使ってお絵描きをしただけで，プログラムをした気になれないと思う人もいるか

[52] 実は，これらの状態も，ユーザのプログラムから直接操作できないように公開されていないインスタンス変数として実現されている（8.5 節参照）。

[53] メソッドについても，そのクラスの機能を実現するために内部的に使用されるだけの公開されていないメソッドが存在する。例えば，TurtleFrame は，線分の両端の座標と色を指定して線分を登録する，addLineElement(int xx, int yy, int x, int y, Color c) というメソッドをもつ。

[54] もちろん，値を返すこともする。

もしれない。それはその通りである。ここでは，最初の例題として，すでに存在するクラスを利用するだけのプログラムを説明した。しかし，通常，Turtle や TurtleFrame のようなクラスを定義すること，つまり，自分の解きたい問題は何をオブジェクトとすれば自然に表現できるか考えて，それにどういうコンストラクタやメソッドをもたせるのが自然か考えて（できれば，Turtle や TurtleFrame のような API の仕様書を書いて），クラスを作成することが本当のプログラミングの作業となる。