るのが普通である[26]。よって，上記のプログラムを T21.java というファイル名で作成したとしよう。

T21.java をコンパイルするには，以下のように javac コマンドを用いる。

```
% javac T21.java
```

javac コマンドにより，引数で与えられたソースプログラムの中に書かれているクラスごとに，クラスファイルが作成される。この場合，T21.class が作成される[27]。このクラスファイルを実行するには，java コマンドを用いる。

```
% java T21
```

java コマンドは，引数で与えられた文字列をクラス名と考え，それを格納してあるクラスファイル（この場合は T21.class）を探し，その main の中身を実行する。その途中で，TurtleFrame と Turtle のクラスの定義が必要となる。そのたびに，java コマンドはこれらのクラスファイルを探して読み込んでくれる。

TurtleFrame が表示するウィンドウは，上部にメニューバーがついている。左の File メニューの中から Quit を選ぶことによりプログラムを終了することができる。また，Speed メニューの中から選択することにより，描画のスピードを変えられる。一番上の no kame を選べば，亀を用いない，高速な描画が可能となる。

練習問題 2.1：図 2.1 の星の絵を描くプログラム T20.java を作成しよう。

2.5 TurtleFrame と Turtle の仕様

このようなプログラムを書くためには，ここで利用されているクラスの仕様（すなわち，どういうコンストラクタがあり，オブジェクトはどういうメソッドを受けつけるのかといったこと）を知る必要がある。このような，利用者のために提供されたクラスに関する情報を，API (Application Programmer Interface) という。以下が，TurtleFrame と Turtle の API の仕様である。

TurtleFrame
コンストラクタ:
TurtleFrame() TurtleFrame を，デフォルトの大きさ (400 × 400) で作成する。
TurtleFrame(int width, int height) TurtleFrame を width × height の大きさで作成する。
メソッド:
void add(Turtle t) t をこのフレームに追加する。
void remove(Turtle t) t をこのフレームから削除する。
void clear() 今までに描かれたすべての線を消す。
void addMesh() 方眼紙のような桝目を表示する。

[26] この例のように，public という修飾子（8.5 節参照）をつけて定義されたクラスの場合は，ファイル名とクラス名が必ず一致している必要がある。

[27] ここでエラーが表示されたら，T21.class は作成されない。「TurtleFrame が見つかりません」というエラーの場合には，サポートページのソフトウェアの展開がうまくいっていなかった場合がある。TurtleFrame.class が同一ディレクトリにあるか確認せよ。1.4 節参照。

Turtle
コンストラクタ:
Turtle() (200,200) という座標に 0 度の角度で Turtle を作成。
Turtle(int x, int y, int angle) (x, y) という座標に angle 度の角度で Turtle を作成。
メソッド:
void fd(int n) n だけ前に進む。
void bk(int n) n だけ後ろに進む。
void rt(int n) n 度だけ右に回る。
void lt(int n) n 度だけ左に回る。
void up() ペンを上げる。
void down() ペンを下ろす。ペンを下ろした状態で進むと，その軌跡が画面に線として描画される。
boolean isDown() ペンを上げた状態なら false，下げた状態なら true を返す。
int moveTo(int x,int y, int angle) (x, y) という座標まで進み，angle の方向を向く。移動した距離を返す。
int moveTo(Turtle t) t と同じ座標まで進み，t と同じ方向を向く。移動した距離を返す。
void setColor(java.awt.Color nc) ペンの色を nc に変更する。
int getX() 現在の座標の X 成分を返す。
int getY() 現在の座標の Y 成分を返す。
int getAngle() 現在の角度を返す。
void speed(int x) タートルの動きの速さを x に設定する。x = 20 がデフォルトである。数字が小さいほど速い。
static void speedAll(int x) 亀全体の速さを 1 から 3 で指定。1 が高速，3 が低速。
フィールド:
java.awt.Color kameColor 亀の色。初期値は緑色。
double kameScale 亀の大きさを表す実数。初期値は 0.4 。
static boolean withKameAll もし，false ならすべてのタートルが亀を表示しないで瞬時に描画を行う。true なら通常の描画を行う。

このうちで，T21.java では，TurtleFrame と Turtle の最初のコンストラクタと，TurtleFrame の add メソッド，それに，Turtle の fd と rt メソッドを用いた。このような仕様があれば，それを見ながら，これらのクラスを用いたプログラムを作ることができる。まだ説明していない部分も多いが，大まかな雰囲気はつかんで頂けると思う。

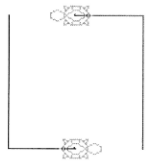
練習問題 2.2 : T21.java の 12 行目の前に，f（に代入された TurtleFrame ）に対する addMesh メソッドの呼び出しを挿入してみよう。14 行目の後に，f に対する clear メソッドの呼び出しを挿入してみよう。

2.6 もう少し複雑な例題 — 標準クラスライブラリの利用 —

もう少し複雑な例を考えてみよう。

リスト 2.2 *T22.java*

```
1  public class T22 {
2    public static void main(String[] args){
3      TurtleFrame f = new TurtleFrame();
4      int x = 200, y = 200, d = 100;
5      Turtle m = new Turtle(x,y,180);  //引数のあるコンストラクタ呼び出し
6      Turtle m1 = new Turtle(x+ d,y+ d,0);
7      java.awt.Color c = new java.awt.Color(255,0,0); //赤色オブジェクトを作成
8      m1.setColor(c);                              //m1 の色を赤色に指定
9      f.add(m);
10     f.add(m1);
11     m.fd(d);
12     m1.fd(d);
13     m.lt(90);
14     m1.lt(90);
15     d = d / 2;    //d の値を d/2 に変更
16     m.fd(d);
17     m1.fd(d);
18     }
19 }
```



3 行目までは，T21.java と同じである。4 行目では，3 つの変数を型宣言して初期化している。今度の型は，整数を表すプリミティブ型である int である[28]。これら 3 つの変数の値は，これから描画する図形の左上の座標と 1 辺の長さを表している。

[28] プリミティブ型については後述する。

5 行目と 6 行目では，Turtle を作成して変数に代入している。今度は，引数を 3 つ与えてコンストラクタを呼び出している。引数も含んだ**コンストラクタ呼び出し式**の一般形は，以下のようになる。

new クラス名(引数式 1, ..., 引数式 n)

また，前節の Turtle クラスのコンストラクタの仕様を見て頂きたい。この説明は，Turtle クラスには 2 つのコンストラクタが存在し，1 つは引数がないもので，もう 1 つは 3 つの int 型の引数をとり，それらが順に Turtle の x, y 座標と向きを指定していることを示している。このように，引数が n 個のコンストラクタの API の仕様は，

クラス名(引数 1 の型 仮引数 1, ... , 引数 n の型 仮引数 n)

という形で書かれている[29][30]。一般に，1 つのクラスは，引数の数や型の違う，いくつものコンストラクタをもつことができる。このことを，コンストラクタの**多重定義**（あるいは**オーバーロード**）という。多重定義されたコンストラクタの呼び出しが実行されると，引数の式の数と型に応じて 1 つのコンストラクタが選ばれて，それが起動される。5 行目の場合，引数の式は，x,

[29] 仮引数は，一種の変数で，ここでは，API の説明文の中でその引数に与えられた値を参照するのに用いている。

[30] なぜコンストラクタの API の仕様をこのように書くかは，6.5 節を見れば明らかとなろう。

y, 180 である。変数 x, y, d はそれぞれ int と宣言されているので，引数の数と型が合致した後者のコンストラクタがこの呼び出しで起動される。6 行目も同様である[31]。

いま，x, y, d には，それぞれ 200, 200, 100 が代入されているので，引数の式の値は，5 行目では 200, 200, 180 で，6 行目では 300, 300, 0 である。よって，これらの値がコンストラクタに渡される。そして，仕様に従い，(200,200) という座標と 180 度の角度をもった Turtle と，(300,300) という座標と 0 度の角度をもった Turtle が生成されて，それぞれ変数 m と m1 に代入される。

メソッドも同様に多重定義が可能で，同じ名前で引数の数や型の違ったメソッドを複数用意することができる。メソッドの多重定義の例として，Turtle の moveTo メソッドの仕様を見て頂きたい。

7 行目も同様に，オブジェクトを作成している。java.awt.Color がクラス名である。これは，画面に表示する色を意味する，Java のシステムと一緒に配布されているクラスである。このように，Java のシステムには最初からたくさんのクラスが付属しており，プログラム中で利用することができる[32]。一般に，様々なプログラムから利用されることを目的に作成されたクラスの集まりは**クラスライブラリ**と呼ばれている。プログラム作成時に利用できるクラスライブラリは，この java.awt.Color などの属する標準のクラスライブラリだけではない。インターネット上の多くのサイトで，誰でもダウンロードして使える形で様々なクラスライブラリが配布されている。このように，Java では，いろいろな人が作成したクラスを利用してプログラムを組むのが普通である。それを可能にするためには，クラス名の衝突[33]を防がなくてはならない。また，個々のクラスがどのクラスライブラリに属しているか，どのクラスがまとまって 1 つの機能を実現しているかといった視点でクラスが整理されていることも重要だろう。それを提供するのが**パッケージ**（8.2 節参照）である[34]。java.awt.Color は，java というパッケージ（Java のシステムに付随した標準クラスライブラリのパッケージ）のサブパッケージである java.awt（グラフィカルユーザインターフェース作成に用いるクラス群を格納したパッケージ）に属する Color というクラスという意味である。

このように，パッケージに属するクラス[35]は，パッケージ名.クラス名 という形で指定する。この記法を，そのクラスの**完全限定名**という。しかし，毎回パッケージ名をつけるのはあまりに不便なので，それを省略する方法がある。

import パッケージ名.クラス名;

とファイルの先頭[36]で宣言しておくと，そのクラスは，クラス名 だけで指定できるようになる。あるいは，

[31] x と d が int 型なので，x+d は x と d の和を意味する int 型の式となる。このように，型は変数だけではなく式に対して与えられる。詳しくは 3.7，3.8 節参照。

[32] java.awt.Color のクラスファイルのある場所については 8.2 節で述べる。

[33] すなわち，同じクラス名のクラスが複数存在し，プログラム中の名前の指定がどちらを指すのかわからないこと。

[34] パッケージの趣旨からすれば，Turtle や TurtleFrame も，利用者に配布するときにはパッケージとして提供すべきだろう。8.4 節参照。

[35] より正確にいうと，自分が今書いているプログラムの属するパッケージ（T21.java や T22.java のように何も指定していないと，名前のないデフォルトのパッケージに属することになる。）と異なるパッケージに属するクラス。8.2 節参照。

[36] すなわち，1 行目より前。

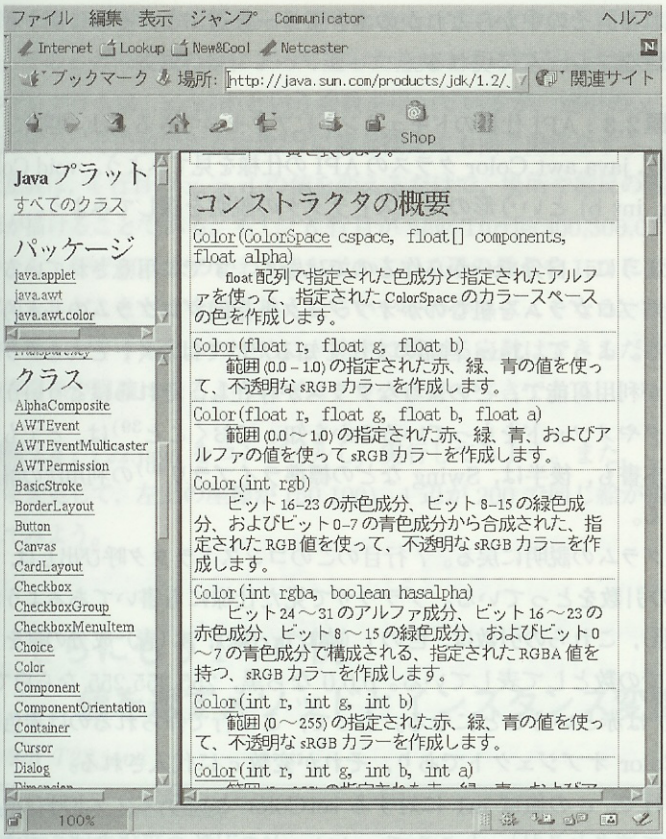


図 2.2 java.awt.Color の API の仕様を Web ブラウザで見ているところ

import パッケージ名.*;

と宣言すると，そのパッケージに含まれるすべてのクラスが，パッケージ名なしで参照できるようになる。よって，リスト 2.2 のプログラムの先頭に，

import java.awt.Color;

と[37]挿入しておくことにより，7 行目は，

Color c = **new** Color(255,0,0);

と書けることになる。

このようなプログラムを書くためには，java.awt.Color の仕様も必要である。それは，ブラウザの中で見ることができる[38]。図 2.2 は，ブラウザで，java.awt.Color クラスの API の仕様を見ているところである。

画面は 3 つのフレームに分かれており，左上のフレームにはパッケージの一覧が表示されている。この中から 1 つのパッケージを選択すると，左下のフレームにそのパッケージに属するクラスとインターフェース（後述）の一

[37] あるいは，
import java.awt.*;

[38] 「Java2 SDK ドキュメント」の中ほどにある，「Java プラットフォーム 1.3 API 仕様」というリンクをたどれば，出すことができる。