第3章 処理の流れ

この章では，プログラムのある部分を繰返し実行したり，ある条件が満たされるときだけあることを行ったりといった，処理の流れを制御する方法について学びます。

3.1 for 文による繰返し

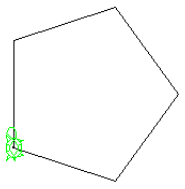
3.1.1 同じことを繰り返す

タートルに対して “100 前に進め” というメソッド呼出しと “72 度回転しろ” というメソッド呼出しを 5 回繰り返せば，1 辺 100 の正五角形を描くことができます。これは，文を繰返し実行させる **for 文**という構文を用いて，次のように簡潔に書くことができます [1)]。

[1)] Eclipse 版では，章に応じたパッケージ宣言 (この章なら package chap03;) が各ファイルの最初に必要です。これ以降のサンプルプログラムでは省略します。また，Turtle を用いたプログラムは必ず import tg.*; が最初に必要です (**??**節)。この行も，基本的に省略します。

リスト 3.1 for 文による繰り返し (`import` 宣言は省略，`T31.java`)

```
 1  public class T31 {
 2      public static void main(String[] args){
 3          TurtleFrame f = new TurtleFrame();
 4          Turtle m = new Turtle();
 5          f.add(m);
 6          int i;
 7          for(i = 0; i < 5; i++){
 8              m.fd(100);
 9              m.rt(360.0/5);
10          }
11      }
12  }
```



7〜10 行目が for 文で，これで，「{ と } で囲まれた部分を 5 回繰り返せ」というコンピュータに対する命令，すなわち文になります [2)]。for 文は次の形をしています。

for (初期化式 ; 繰返し条件式 ; ループの更新式) 繰り返す文

この例題の for 文では，初期化式 は i = 0，繰返し条件式 は i < 5，ループの更新式 は i++ です。これらの場所には式を書きます [3)]。式は，オ

[2)] **??**節**??**。

[3)] 初期化式 および ループの更新式 には，複数の式を “,” で区切って書くことができます。また，後述するように，初期化式 に変数宣言を書くこともできます。これら三つの式の部分は，空白にすることもできます。

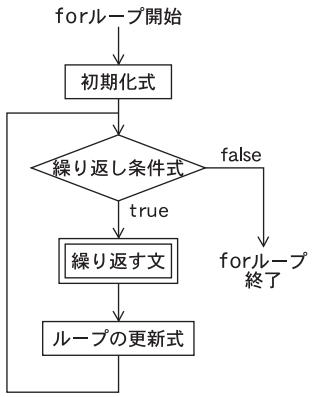


図 3.1 for 文の制御の流れ

ブジェクトや数といった値を意味する表現のことだと 2 章で述べましたが[4)]，式の値を計算する途中で，変数に値を代入するまどの状態の変化を起こすことができます[5)]。初期化式 と ループの更新式 には変数の値の変化を起こす式を書きます[6)]。i = 0 は変数 i に 0 を代入する代入式です[7)]。また，i++ は，変数 i の値を 1 増やす式です[8)]。繰返し条件式 には，boolean 型の式が書けます。boolean は真偽値，つまり true と false の 2 つの値をとるプリミティブ型です。詳細は 3.3 節で述べます。i < 5 は boolean 型の式で，i が 5 より小さいときには true, 5 以上のときには false を意味します。

そして，繰り返す文 が[9)]

```
{ m.fd(100); m.rt(72); }
```

です。これは，**ブロック**と呼ばれる文です[10)]。ブロックは，

{ 文 1 文 2 ... 文 n }

という形をしており，その中の文を順番に実行せよという命令を意味します[11)]。ブロックの中には，文に混じってローカル変数の宣言も書くことができます。その場合，その変数はブロック内のみ有効となります[12)]。

for 文の実行は，次のように行われます。まず，初期化式を実行します。それから 繰返し条件式 が成り立つか調べます。もし 繰返し条件式の値が false なら for 文の実行を終了します。true なら 繰り返す文 を実行して，それから ループの更新式を実行します。そして 繰返し条件式 が成り立つか調べます。これを，繰返し条件式 が成り立たなくなるまで繰り返します。この実行の流れは，図 3.1 のように図示できます[13)]。

リスト 3.1 では，初期化式 が i = 0 なので，まず変数 i に 0 が代入されます。それから i < 5 かどうかを確認し，これは真なので，{,} で囲まれた部分を実行し，i++ により i の値を 1 増やし（つまり，i は 1），再び i < 5 かどうか確認します。このように，繰返しを行うたびに i の値が 0, 1, 2, 3, 4

[4)] **??**節**??**。

[5)] 詳細は**??**章で述べます。

[6)] 式の値は用いません。

[7)] 代入式の最後に ; をつけたものが代入文です。

[8)] i の値を 1 増やすという点では，
i = i + 1
と同じです。実際ここに，そのように書いてもよいです。

[9)] 上の例では，for のある行の最後の { から 10 行目の } までが一つのブロックです。改行や空白は区切りであり，どれだけ入っていても意味が変わらないことを思い出してください。

[10)] 文はコンピュータに対する命令でした。

[11)] もちろん，繰り返す内容が文一つなら括弧で囲んでブロックにする必要はありません。

[12)] いままでのプログラムの main の中身（たとえば，リスト 3.1 の 2 行目 の { から 11 行目 の } まで）も一つのブロックです。

[13)] 繰返し条件式 が空白のときには，つねに true を意味します。よって，
for(i = 0;;i++)...
とか
for(;;) ...
と書くと，無限に繰り返すことになります。

と増えます。そして，i が 5 になったとき，i < 5 の値は false となり， for 文の実行を終了します。つまり，{,} で囲まれた部分は 5 回実行されます。

初期化式 には，式の代わりに変数宣言を書くこともできます。よって，6, 7 行目をまとめて

for(**int** i = 0; i < 5; i++) ...

と書くこともできます [14]。このように，あることを n 回繰り返したいときに，繰返し回数をカウントするための変数（ここでは i とする）を用意して

for(**int** i = 0; i < n; i++) { 繰り返したい式の列 }

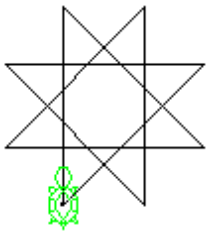
と書くのが典型的な for 文の使い方です。n のところに式が書けることに注意してください [15]。for 文自体は上に述べたように柔軟性にとんだ形をしており，これ以外にも様々なループが実現可能です [16]。

正五角形なので 9 行目は rt(72) と書くこともできますが， n 角形に拡張することも考えて 360.0/5 としています。正五角形の時なら，ここを 360/5 と書いても同じですが，正七角形の時には 360/7 と書いたのでは正しい描画になりません。 int と int の割り算は切り捨てを行うので，結果が整数値となります。よって，m.rt(360/7); と書いたのでは 51 度しか回転しません。それを防ぐには，割り算を 360.0 / 7 と書く必要があります。割り算は，どちらかの引数が double ならもう片方も double の値に変換して double の間の割り算が実行されて double の値が得られます。よって，これにより， 360 / 7 = 51.428... という値 [17] が m.rt に渡されます。

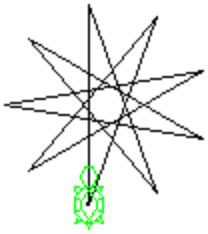
練習問題 3.1： リスト 3.1 のプログラムを変更して，double 型の変数 s と int 型の変数 n に対して，1 辺の長さ s で n 角形を描くようにしよう (P31.java)。これを用いて，1 辺の長さ 1 で 360 角形を描いてみよう。

練習問題 3.2： 練習問題 **??** および練習問題 **??** と同じプログラムを，for 文を用いて作ろう (P30.java, Mouse31.java)。

練習問題 3.3：* 練習問題 **??** で 5 角形の星型を描いたが，これは，5 角形の頂点を一つ飛ばしに結んでできている。n と k を int 型の変数とし，n 角形の頂点を k−1 個飛ばしに結んでできる図形を描くプログラムを作成しよう (Star31.java) [18]。



n=8, k=3



n=9, k=4

[14] i はこの for 文の中でのみ有効な変数となります。

[15] ただ単に 5 回同じことを書くのより簡潔に書けるというのではありません。

[16] 8 ページの傍注 **39** 参照。

[17] 正確には 360 / 7 という実数ではなく，360 / 7 に一番近い倍精度浮動小数点値が渡されるので，誤差が生じます（**??**節）。

[18] k が 1 のときには，n 角形が描けるはずです。ここでは，n と k が互いに素（共通の約数がない）のときのみ考えています。タートルは，絵を描く間に図形の中心の周りを k 回，回ります。よって，その間に，合計 k * 360 度向きを変えなくてはなりません。そのことから，それぞれの頂点で，何度向きを変えればよいか考えましょう。

練習問題 3.4： ?? 節で説明するように，Math.random() [19] を呼び出すことにより 0 以上 1 未満の double の乱数を得ることができる [20]。これを用いて，0 以上 360 未満の乱数 a [21] を発生させて，10 前に進むことと a 度回転することを 1000 回繰り返させてみよう (Random31.java) [22]。進む距離も、0 以上 10 未満の乱数にしてみよう (Random32.java)。

[19] Math クラスのクラス・メソッドの呼び出しです

[20] つまり，double 型の変数 r に対し，
r = Math.random();
と書くと、実行のたびにランダムな値が r に代入されます。

[21] 360 倍することにより、すなわち、
a=360*Math.random();
により得られます。

[22] 適宜 TurtleFrame の Speed メニューを利用しましょう。

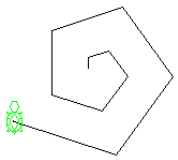
3.1.2 繰返しをカウントする変数値の利用

繰返しの回数を数えている変数を繰り返す文 の中で用いることができます。それにより，繰返しのたびに実行する内容に変化をもたせることが可能となります。次のプログラムは，繰り返すたびに進む長さが $10, 20, \ldots, 100$ と増えるようにリスト 3.1 を変更し，渦巻きを描画するようにしたものです [23]。

[23] ここで，i <= 100 は，i $\leq$ 100 なら true，そうでないなら false です（3.3 節）。

リスト 3.2 繰り返しをカウントする変数の利用（package，import 宣言は省略，T32.java）

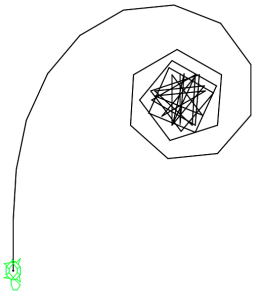
```
1  public class T32 {
2      public static void main(String[] args){
3          TurtleFrame f = new TurtleFrame();
4          Turtle m = new Turtle();
5          f.add(m);
6          int i;
7          for(i = 1; i <= 10; i++){
8              m.fd(i * 10);
9              m.rt(72);
10         }
11     }
12 }
```



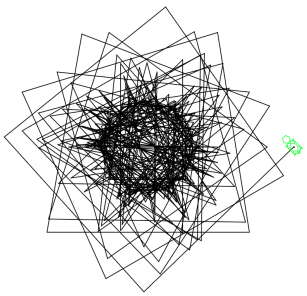
練習問題 3.5： 50 進み 2 度向きを変える，50 進み 4 度向きを変える，...，50 進み 180 度向きを変えるということを実行させてみよう。繰り返しをカウントする変数を 2, 4, ..., 180 と変化させて書いてみよう [24](R30.java) 。

[24] ループの更新式を i=i+2 にすればよいです（??節 ?? 参照)。スタートする場所を工夫して，全体が収まるようにしてください。

練習問題 3.6： 1 進み 1 度向きを変える， 2 進み 2 度向きを変える，... ，270 進み 270 度向きを変えるということを実行させてみよう (R31.java) 。



練習問題 3.5



練習問題 3.6

繰返しの中での変数に代入を行うことにより，変数の値を次々と変化させていくことができます。次のプログラムは，1〜10 の和を計算して表示するものです [25][26]。sum += i; は sum = sum + i; と同じ意味です（?? 節）。繰返しの中で sum に i を足しこむことにより，繰り返す文の実行が終わった時点では，常に，sum にそのときの i の値までの和が代入されています。System.out.println(変数); によってその変数のその時の値が画面に表示されます [27]。右に，変数 i と sum の値のプログラムの実行による変化を表にしておきます。

[25] 繰り返す内容が 1 文だけなのでブロックにしていません。

[26] このプログラムは，ウィンドウを用いたプログラムと異なり，main メソッドの終了とともに終了します（?? 節）。

[27] println 文については?? 節で詳しく解説します。

リスト 3.3　1 から 10 までの和（Sum31.java）

```
1  public class Sum31 {
2      public static void main(String[] args){
3          int sum = 0;
4          for(int i = 1; i <= 10; i++)
5              sum += i;
6          System.out.println(sum);
7      }
8  }
```

i	1	2	3	4	5	6	7	8	9	10
sum	1	3	6	10	15	21	28	36	45	55

練習問題 3.7： $1 + 3 + 5 + 7 + \cdots + 21$ の和を表示するプログラムを作成しよう (Sum32.java)[28]。途中の和も表示するようにしよう (Sum33.java)。

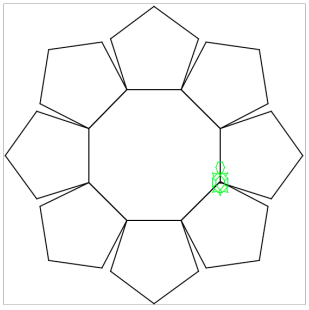
[28] 21 のところは変数にしよう。

3.1.3　繰返しのネスト

for 文は内部に他の文を含む複雑な構造をしていますが，これ全体も一つ

リスト 3.4　繰り返しのネスト（import 宣言は省略，T33.java）

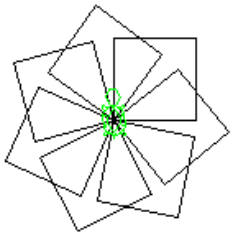
```
1  public class T33 {
2      public static void main(String[] args){
3          TurtleFrame f = new TurtleFrame();
4          Turtle m = new Turtle();
5          f.add(m);
6          for(int j = 0; j < 8; j++){
7              for(int i = 0; i < 5; i++){
8                  m.fd(50);
9                  m.rt(72);
10             }
11             m.fd(50);
12             m.lt(45);
13         }
14     }
15 }
```



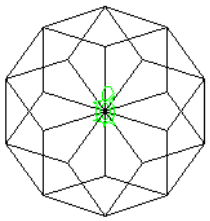
の文です。よって，これをさらに他の for 文の 繰り返す文 の中に入れることにより，繰り返し実行することができます。このように，繰返しの中で繰返しを行うことを，繰返しのネストと言います。リスト **??**は，1 辺 50 の五角形を右回りに描いて 50 進んで左に 45 度回転することを，8 回繰り返しています。

練習問題 3.8： リスト 3.4 の 8, 5 のところを変数にし，n 角形の各辺の回りに k 角形が存在する絵を描画するプログラムに書き換えよう (P35.java)。

練習問題 3.9： 図のように，n 角形を r 枚，回転しながら描画するプログラムを作成しよう (P36.java)。r=2 から始めて，描画が終わったら画面を全部消して（TurtleFrame の clear メソッドを使用）r を 1 増やしてまた描画するということを，無限に繰り返させてみよう (P37.java) [29]。



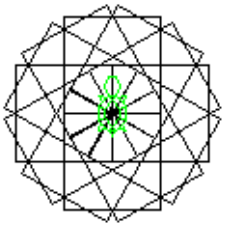
n=4, r=7

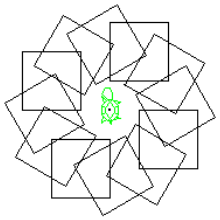


n=5, r=10

[29] 無限に繰り返すことは，for(r=2;;r++) 文 によりできます。これにより，for 文が 3 重にネストすることになります。ネストが深くなると，プログラムの理解が難しくなります。ネストを深くせずに同様のことを記述する方法を，**??**節で学びます。

練習問題 3.10：* 練習問題 3.9 において，回転の中心を，頂点以外の場所に置くことを考えよう (P38.java)。左図は，辺の中点に中心をおき，四角形を 12 枚回転しながら描画したもの，右図は，辺の長さの半分だけ辺を延長した点に中心をおき，四角形を 12 枚回転しながら描画したものです [30][31]。



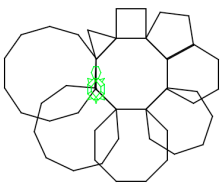


[30] up, down, bk などのメソッドを利用しよう。

[31] 回転の中心を表現するための変数を考えて，変数の値を変えるだけで，両方の絵が描けるようにしよう。

練習問題 3.11：傍注の絵 [32]（八角形の各辺の回りに 三角形から十角形が配置されている）を描くプログラムを作成しよう (P39.java)。

[32]



練習問題 3.12：次のように，Java と 1 行に 10 個，10 列書くプログラムを作成しよう (Q31.java)。System.out.print(”Java ”); で，”Java ” と画面に表示できます。また，System.out.println(””); により，改行をすることができます [33]。次に，1 行目に 1 個，2 行目に 2 個，…，10 行目に 10 個書くプログラムを作成しよう (Q32.java)。次に，1 行目に 10 個，2 行目に 9

[33] print 文, および println 文については**??** 節を参照。

個，…，10 行目に 1 個書くプログラムを作成しよう (Q33.java)。また，前に空白をあけて右揃えにしよう (Q34.java) [34]。

34) 変数 n に対し，n 行の同様な文字列を出力する様に作成しよう。

```
Java Java ... Java
Java Java ... Java
        ...
Java Java ... Java
```

Q31.java

```
Java
Java Java
....    ...
Java Java ... Java
```

Q32.java

```
Java Java ... Java
     Java ... Java
          ...
               Java
```

Q34.java

3.2　while 文による繰返し

繰返しの回数が最初から分かっているときには，for 文は使いやすいです。しかし，回数が決まっておらず，ある条件が成り立っている間だけ繰り返すという繰返しも存在します。そのときには，**while 文**という書式を用いたほうが，自然に表現できます。while 文は，次の形をしています。

while(繰返し条件式) 繰り返す文

この実行の流れは，図 3.1 において，初期化式 および ループの更新式 を省いたものとなります。

35) リスト 3.2 で，for の繰返し条件を i <= 10 から m.getY() > 0 に変更すれば，リスト 3.5 と同等のプログラムになります。このように，while を使ったプログラムは for で置き換えられますし，逆に，for 文を用いた繰返しを while 文で書き直すこともできます。どちらを使うかは，書きやすさや読んだ時の分かりやすさで決めましょう。

リスト 3.5 は，リスト 3.2 の 7 〜10 行目を，タートルの y 座標が 正の間だけ繰り返すように変更したものです [35][36]。

36) 変更する部分だけを載せています。

リスト 3.5　while 文による繰り返し（一部，T34.java）

```
1    int i = 1;
2    while(m.getY() > 0){
3        m.fd(i * 10);
4        m.rt(72);
5        i++;
6    }
```

練習問題 3.13： $1^2 + 2^2 + 3^2 + \cdots$ の和が 1000 を超えるのは，何の 2 乗までを足したときか求めるプログラムを作成しよう (Sum34.java)。

for, while の他に，

do 繰り返す文 **while**(繰返し条件式);

という形の **do 文**という繰返し構文も存在します。while 文と同様ですが，繰り返す文 を実行してから 繰返し条件式 の値を調べるため，最初から条件が不成立のときでも 1 回は繰り返す文 が実行されます [37]。

37) for や while では，1 回も 繰り返す文 が実行されないこともあることに注意してください。

3.3 論理演算子

boolean 型は，true（真）, false（偽） という二つの値からなるプリミティブ型です [38]。リスト 3.1 では，boolean 型の繰返しの条件式として i < 5 を用いました。この < は，数値どうしの大小比較を行い，不等式が成り立つかどうかを boolean 型の値として返す演算子 [39] です。同様の演算子として，>, <, >=, <=, ==, !=があります。下の表で，書かれていないときの値は false です。

式	値
a > b	a > b のとき true。
a < b	a < b のとき true。
a >= b	a ≥ b のとき true。
a <= b	a ≤ b のとき true。
a == b	a と b が等しいとき true。
a != b	a と b が等しくないとき true。

== と != は，引数がクラス型などの参照型のときにも使用できます。参照型どうしの比較は，両辺が同一のオブジェクトかどうかで比較します [40]。

条件を boolean 型の式で表すときには，二つの条件 A, B に対し，「A と B が両方成り立つとき」とか，「A と B のどちらかが成り立つとき」あるいは，「A が成り立たないとき」といった条件を書きたいこともあります。これに対応して，二つ（あるいは一つ）の boolean 型の値から，これらの条件を表す boolean 型の値を求める演算子が存在します [41]。

A && B	A と B の両方が true のとき true（論理積）。
A || B	A と B のどちらかが true のとき true（論理和）。
A ^ B	A と B の一方だけが true のとき true（排他的論理和）。
!A	A が false のとき true（否定）。

例をあげます。

x が 100 と等しい。	x == 100
x^2 が 100 より大きい。	x * x > 100
x か y のどちらかが 0。	x == 0 || y == 0
$0 \leq x < 100$。	0 <= x && x < 100
$0 \leq x < 100$ または $0 \leq y < 100$ 。	(0 <= x && x < 100) || (0 <= y && y < 100)[42]
x は 2 以外の偶数。	x % 2 == 0 && x != 2

ここで % は，+ や * などと同様な，数を二つもらい数を返す演算子で，x % y は x を y で割った余りを意味します (?? 節)。

[38] ?? 節の一覧表参照。

[39] 足し算 (+) は，数を二つもらい数を返す演算です。これと同様に，< を，数を二つもらい true か false という値を返す演算と考えます。このような演算を表す記号を**演算子**と言います（第??章)。

[40] ??節参照。

[41] この表で，それぞれこれ以外の場合の値は false です。これらの式の評価方法については，6.4 節で述べます。

[42] ??節で説明する演算子の強さ関係から，この括弧は省略可能です。

練習問題 3.14： 次の条件を，boolean 型の式として表してみよう [43]。

1. x 月は小の月 (2,4,6,9,11) (Q35.java)。
2. 西暦 y 年はうるう年（つまり，4 の倍数であって 100 の倍数ではないか，400 の倍数）(Q36.java)。
3. 点 (x, y) は，(0,0), (100,100) を左上と右下の頂点とする正方形の内部にある (Q37.java)。

[43] 次節の if 文を用いて変数 x，y が条件を満たすときに Yes，満たさないときに No と表示するプログラムを作成しよう。

練習問題 3.15：練習問題 3.4 のプログラムを修正して，(0,0), (400,400) を左上と右下の頂点とする画面から外に出るまで 10 前に進むことと a 回転することを繰り返すようにしよう (Random33.java) [44]。

[44] 外に出る直前ではなく，出たところで終了すればよい。

3.4　if 文

変数などの状況に応じて，実行する内容を変更したいことがあります。そのとき，**if 文**を用います。次は，if 文の最も基本的な形です [45]。

if(条件式) 条件が成り立つときだけ行う文

[45] 条件が成り立つときに行う文が複数の文からなるときには，{, } で囲んでブロックにします。

リスト 3.6　if 文（一部，T35.java）

```
1    for(int i = 0; i < 5; i++){
2        if(i == 3){
3            m.setColor(Color.RED);
4        }
5        m.fd(100);
6        m.rt(72);
7    }
```

リスト 3.6 は，for 文によって 5 回繰り返しますが，その中で i が 3 のときだけ，3 行目を実行します [46]。Color.RED というのは **??** 節で説明しますが，赤色の色オブジェクトが最初から代入されているクラス変数です [47]。よって，i が 3 の時に setColor が行われ，それ以降の辺が赤色になります。

ある条件を満たしていればあることを，それ以外のときには他のことをさせる場合には，次の形を用います。

if(条件式) 条件が成り立つとき行う文
else 条件が成り立たないとき行う文

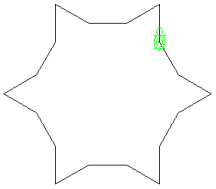
次のプログラムは，i が 3 の倍数のときには左に 120 度，それ以外のときには右に 30 度回転するのを 18 回繰り返します。

[46] 最初にインポート文 import javafx.scene.paint.Color; が必要です (**??**節)。

[47] リスト 3.8，3.11 に現れる Color.GREEN, Color.YELLOW も同様です。

リスト 3.7 if - else 文（一部，T36.java）

```
1    for(int i = 0; i < 18; i++){
2        m.fd(50);
3        if(i % 3 == 0) m.lt(120);
4        else m.rt(30);
5    }
```



練習問題 3.16： 練習問題 3.4 のプログラムを修正して，(0,0) から (400,400) の画面から外に出たら，それまでの線を消して，ペンをあげて中心に戻ることを無限に繰り返すようにしよう (Random34.java)[48]。

[48] ペンを上げて移動するには，up, down メソッドを用いればよいです。

練習問題 3.17： * Math.random() を用いて 2 匹のタートルの競走をさせよう。2 匹のタートル m1, m2 を (0,100), (0,300) に右向きに配置し，Math.random() の値が 0.5 未満なら m1 を，そうではないなら m2 を 10 だけ前に進めることを，どちらかが 30 進むまで繰り返そう (Race31.java)。それぞれの進んだ数を数えて，10 回，20 回進んだところで色が変わるように，28 進んだところでタートルが大きくなるようにしよう (Race32.java) [49]。

[49] 大きさや色を変更してから反映させるには，タートルを動かす必要があります (**??**節)。

条件式 A が成り立てば 文 a を，そうではなくて 条件式 B が成り立てば 文 b を，…，すべてが成り立たないときには 文 n を行う，という場合には，次の形を用います [50]。

[50] すべて成り立たないときに何もさせないのなら，最後の else はなくてもよいです。

if(条件式 A) 文 a
else if(条件式 B) 文 b
...
else 文 n

以下は，赤，緑，黄色を交互に用いて十二角形を描くプログラムです [51]。

[51] これも，最初に
import javafx.scene.paint.Color;
が必要です。リスト 3.11 も同様です。

リスト 3.8 if - else if 文（一部，T37.java）

```
1    for(int i = 0; i < 12; i++){
2        if(i % 3 == 0){                  // i が 3 の倍数のとき
3            m.setColor(Color.RED);
4        }else if (i % 3 == 1) {          // i が 3 で割って余り 1 のとき
5            m.setColor(Color.GREEN);
6        }else{                           // それ以外（3 で割って余り 2）のとき
7            m.setColor(Color.YELLOW);
8        }
9        m.lt(30);
10       m.fd(50);
11   }
```

[52] 各繰り返しで 2 回 Math.random() を呼び出さないように注意しよう。
double c=Math.random();
と変数に代入しておき,
if(c < 1.0/3){...}
else if(c < 2.0/3){...}
とすればよい。

練習問題 3.18：練習問題 3.17 のプログラムを，3 匹のタートルが競争するように修正しよう (Race33.java)。乱数の値が 1/3 より小さいかどうか， 2/3 より小さいかどうかを，順に比較すればよい。1/3, 2/3 は，それぞれ 1.0/3, 2.0/3 とプログラム中で表記する（**??** 節）[52]。

3.5 break 文 と continue 文

for，while などの繰返し文では，繰返しの先頭でいつ繰返しを終了するのか判断していました。プログラムによっては，繰り返す文 の中で，繰返しを終了するための条件判断を行いたいこともあります。そういうときに break; という **break 文**を用います。break 文の実行により，その文が含まれる最も狭い範囲の繰返し文（for 文，while 文，do 文，switch 文（後述））の実行が終了します。

[53] $x^2 - 145x + 3616 = (x - 32)(x - 113)$ です。

次のプログラムは，1〜100 の範囲で $x^2 - 145x + 3616 = 0$ [53] の整数解が存在するかどうか調べるものです。

リスト 3.9 break 文 (Hoteisiki31.java)

```
1  public class Hoteisiki31{
2      public static void main(String[] args){
3          int x;
4          for (x = 0; x < 100; x++){
5              if(x * x - 145 * x + 3616 == 0)break; //この break 文が実行されると
6          }                                          // 4〜6 行目の for 文が終了する
7          if (x < 100) System.out.println(x);
8          else System.out.println("Not exist");
9      }
10 }
```

for 文の繰返しを終了し 7 行目にくるのは，x < 100 が成り立たなくなったときと，break 文が実行されたときです。前者のときには x の値は 100 になっているので，8 行目の else のほうが実行されます。後者のときには x の値は break を実行したときのままなので，7 行目の if のほうが実行されます。

練習問題 3.19：練習問題 3.17 のタートルのレースで，30 回繰り返す間に 2 匹のタートルの差が 50 ついたらレースを終了して後ろのタートルを画面から消すようにしよう (Race34.java) 。

練習問題 3.20： 練習問題 3.2 のプログラムを修正して，マウスで指定された点を線でつないでいき，コントロールキーを押しながらマウスを押されたらタートルを消して終了することにより折れ線を描画するようにしよう (Mouse32.java)。それを繰り返すことにより，折れ線の集まりとして絵を描

画できるようにしよう (Mouse33.java)。タートルの消えた状態でシフトを押しながらマウスを押すとプログラムが終了するようにしよう。

ループが 2 重になっており，外側のループから一気に脱出したいときには，このような break 文では難しいです。break ラベル名: という，ラベル付きの break 文を用いればその break 文を内部に含み，同じラベル [54] をもつ文から脱出できます。リスト 3.10 は，$x^3 + 2xy - 10x - y^2 = 39$ の整数解を，$(0 \leq x < 100, 0 \leq y < 100)$ の範囲でしらみつぶしに調べるプログラムです [55]。6 行目の if 文が成り立ったら break out; が実行され，out: というラベルの付けられた 4〜8 行目の for 文の実行が終了し，9 行目にきます [56]。

[54] 文の先頭にラベル名: と書くことにより，文にラベルを付けることができます。

[55] 一つの解は，$x = 4, y = 5$ です。

[56] もし，6 行目がラベルなしの break 文だったら，5〜7 行目の for 文からぬけるだけです。

リスト 3.10　大域脱出 break 文 (Hoteisiki32.java)

```
1  public class Hoteisiki32 {
2      public static void main(String[] args) {
3          int x=0, y=0;
4  out:    for(x = 0; x < 100; x++){
5              for(y = 0; y < 100; y++){
6                  if(x * x * x + 2 * x * y - 10 * x - y * y == 39)break out;
7              }
8          }
9          if (x < 100) System.out.println("x = " + x + ",y = " + y);
10         else System.out.println("Not exist");
11     }
12 }
```

練習問題 3.21：* 練習問題 **??** の Mouse33.java を修正して，タートルの消えてない状況でもシフトを押しながらマウスを押すとプログラムが終了するようにしよう (Mouse34.java)。

また，ある条件を満たしていたら，それ以降の部分を省略して次の繰返しの先頭に移りたいこともあります。**continue 文**は，そのような役割を果たします。たとえば，リスト 3.9 のプログラムで，5 行目の前に，

if(x % 2 == 0) **continue**;

を挿入すると，0〜99 の範囲の奇数だけを調べることになります。

練習問題 3.22：そのような変更を Hoteisiki41.java に加えよう (Hoteisiki33.java)。

3.6 switch 文

リスト 3.8 のプログラムは，i % 3 の値によって行うことを分けていました。このように，ある式の値に応じて実行する内容を分けることは，**switch**

文を用いれば自然に書けます。switch 文は，以下の形式をとります [57)][58)]。

switch (式) {
case 定数式 1:
　そのとき実行させたい式の列 1
　break;
case 定数式 2:
　そのとき実行させたい式の列 2
　break;
...
default:
　どれにもマッチしなかったとき実行させたいこと
}

リスト 3.8 のプログラムは次のようにも書けます [59)]。

リスト 3.11　switch 文（一部，T38.java）

```
1     for(int i = 0; i < 12; i++){
2        switch (i % 3) {
3        case 0:
4           m.setColor(java.awt.Color.red);
5           break;
6        case 1:
7           m.setColor(java.awt.Color.green);
8           break;
9        case 2:
10          m.setColor(java.awt.Color.yellow);
11          break;
12       }
13       m.lt(30);  m.fd(50);
14    }
```

練習問題 3.23：T36.java を switch 文を用いて書き換えよう (Q38.java)。

57) 式は, char, byte, short, int のどれかの型でなければならなりません。定数式とは，プログラム実行前から値が定まっている式で，リテラルや，定数に初期化された final 変数 (3.5.1 項) や，それらから演算で作られたものです。

58) 最後の default: はなくてもよいです。途中の break 文もなくてもよいです。そのときには，次の case の中が続けて実行されます。switch 文の実行の手順は,「式 の値と同じ case ラベルをもつ文から実行が始まり, switch 文の終わりにくるか，break 文を実行するまで順に実行する。」と言い換えられます。

59) こちらでは i % 3 は 1 回しか評価されないことに注意してください。