6.4 最小公倍数

最大公約数の次は最小公倍数の番だ。

ところで、最大公約数が分かれば最小公倍数は直ちに計算できることを知っているかね？　それはこういうことだ。

二つの数を M, N としておこう。この 2 数の最大公約数を g とすると、$M = mg, N = ng$ と書いてよいだろう。そして m, n は互いに素であることも重要だ。よって、M, N の最大公約数は mng であることが分かる。

では、**Java** が M, N の値を受け取ったとき、これだけの情報から最小公倍数を計算するにはどうすればいい？　簡単なことだ。$M \times N/GCM(M,\ N)$ でよい。そこで最小公倍数を求めるプログラムは次のようになる。

programming list [FindLCM.java]

```
import java.util.Scanner;

public class FindLCM {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        while(true) {
            System.out.print("input 2 numbers 'a b': ");
            int a = s.nextInt();
            int b = s.nextInt();

            System.out.println("LCM( " + a + ", " + b + " ) = " + lcm(a, b));
        }
    }

    public static int lcm(int m, int n) {
        return m * n / gcm(m, n);
    }

    public static int gcm(int aa, int bb) {
        if (aa % bb > 0) {
            return gcm(bb, aa % bb);
        } else {
            return bb;
        }
    }
```

```
28: }
```

いま導いた最小公倍数の式を使えば、最小公倍数を求めるプログラムは [FindGCM.java] に 17:–19:行目を付け加えるだけの話である。もっとも、`lcm()` メソッドを付け加えずとも、13:行目の `lcm(a, b)` を `a * b / gcm(a, b)` に書き換えるだけでもよい。ただ、そういうことでは芸がないので、他にちょっと変更したところがある。それに豆旅も長くなってきたので、一つのプログラム—クラスと言ってもよい—に 2 個以上のメソッドがあっても大丈夫だろうと思うからだ。

ちょっとした変更は二ヶ所である。一つは `main()` メソッドに `while(true)` を入れたこと。ここまでのプログラムは、ほとんど単発で実行が終了するので、別の値で繰り返し実行するのは手間だったろう。おそらく気が利いている君たちのことだ、`main()` メソッドに `for(;;)` を入れて繰り返しの入力ができるようにしていたのではないだろうか。ここでは 8:行目のように、`for(;;)` の代わりに `while(true)` を使っただけだ。どちらも同じことなので、好みで使ってかまわない。

さて、このクラスはメソッドが 2 個あるわけだが、13:行目で `lcm(a, b)` を計算するとき、**Java** は `lcm()` メソッドに値 `a`, `b` を渡して、何らかの値が返ってくるのを待っている。一方、値を受け取った `lcm()` メソッドは、受け取った値を `m`, `n` と解釈して `m * n / gcm(m, n)` の結果を返そうとするのだ。しかし `gcm(m, n)` もまたメソッドである。そこで `lcm()` メソッドは、いま持っている値 `m`, `n` をさらに下請けに—つまり `gcm()` メソッドに—渡して値を返してもらうわけだ。

さあ、今度は値の逆流だ。`gcm()` メソッドが返した最大公約数が `lcm()` メソッドへ渡り、`lcm()` メソッドが返す最小公倍数によって 13:行目の `System.out.println;` 文が実行されるのである。

そして、もう一つのちょっとした変更が `gcm()` メソッドだ。もちろんユークリッドの互除法で最大公約数を求めているのだが、ここでは再帰を使ってみた。再帰の方がユークリッドの互除法を直観的に表現していると思わないかい？

TRY! このプログラムでは、最小公倍数を `m * n / gcm(m, n)` で計算して返している。しかし、このようなときは `m / gcm(m, n) * n` で計算する方がよい。`m / gcm(m, n) * n` で計算する `lcm2()` メソッドを追加し、大きな 2 数の最小公倍数を求め、違いを比較してみよ。

コンピュータにはどうしても扱える数の大きさに限界がある。大きすぎる数では桁あふれの危険があるし、小さすぎる数では**桁落ち**の心配があるのだ。こういった問題は、計算順序を変えること

で回避できることがある。プログラミングには、数学では気にしなくてよいことも気にする必要がある。