2...の豆旅

2.1 関数（メソッド）

数学に**関数**は付きものである。関数は特別難しいものではない。記述の仕方に慣れさえすれば、関数はむしろ便利で使いやすいのである。

関数 $f(x) = x^2$ があるとする。これは、f という名の関数に (何がし)^2 の機能を持たせている。そのため関数 f に 5 を与えると、25 という値が返ってくるのだ。また関数 $g(x) = 2^x$ は、g という名の関数に $2^{(\text{何がし})}$ の機能を持たせている。よって関数 g に 10 を与えると、1024 という値が返ってくるのだ。

ここでちょっと変な感じを持った人がいるだろうか？　私が関数 f、g と言ったことに対してだ。だが、これでよい。$f(x)$ と書くのは、関数 f が変数 x を使うことを明らかにするためなのだ。もし皆が皆、変数には x しか使わないと取り決めていたら、$f = x^2$、$g = 2^x$ で十分である。しかし、変数には x 以外の文字を使うことはよくあるし、変数が 2 個以上ある関数だってある。さらには文字定数を利用する場合があることを思えば、関数名に続けて ( ) を付け加えることが不可欠なのである。

関数の例をあと二つ提示しよう。

一つ目は $f(x,\ y) = x^2 - 3y$ でどうだろう。関数 f は変数に x, y を使う。その結果返ってくるのが $x^2 - 3y$ で計算された値だ。だから $f(5,\ 10)$ と書けば、$x = 5, y = 10$ を代入して計算した値 -5 が返ってくることになる。一般に数学で使う関数は、実数値を受け取って実数値を返すので、$f(0.8,\ 0.1)$ でも計算可能で 0.34 が返ってくる。

二つ目の関数は $g(x) = \lfloor x \rfloor$ でどうだろう。見慣れない記号 $\lfloor\ \rfloor$ はフロア記号と呼ばれる。フロア記号を簡単に説明すると、与えられた値を超えない整数のうち、最大のものを返す機能を持っている。むむ、難しい表現だ。具体的には $g(3.14)$ と書けば 3 が返され、$g(-3.14)$ と書けば -4 が返されるのだ。これは純粋に数学の話だよ。

EX.　$g(x) = \lfloor x \rfloor$ において、$g(10)$ とするといくつが返ってくるか？　また $g(-10)$ や $g(-8.9)$ ではど

うか？

いま二つの関数を例に出してみたが、値を求めるためにする記述は、いずれも $f(5,\ 10)$ や $g(3.14)$ のように簡便だ。にもかかわらず -5 や 3 の値が返ってくるのは、裏方で f や g がせっせと $5^2 - 3 \times 10$ や $\lfloor 3.14 \rfloor$ の計算をしているからである。また、$f(3,\ 8)$ のように別の値を与えれば、それに応じた値が返ってくる。これは、関数がする仕事がきちんと決められているからである。

何のことはない。関数とは、与えられた値を別の値に加工する手続きなのだ。

この豆旅では**指数関数**を話題に取り上げる。それも、もっとも基本的な $f(x) = 2^x$ にしておこう。しかしその前に、**Java** における関数の仕組みを知るために、x の値を入力すると x^2 の値を表示するプログラムを組んでおこう。

programming list [Xsquare.java]

```
 1: import java.util.Scanner;
 2:
 3: public class Xsquare {
 4:
 5:     public static void main(String[] args) {
 6:         Scanner s = new Scanner(System.in);
 7:
 8:         System.out.print("input 'x' of x^2: ");
 9:         int n = s.nextInt();
10:
11:         System.out.println(n + "^2 = " + sqf(n));
12:     }
13:
14:     public static int sqf(int x) {
15:         return (x * x);
16:     }
17: }
```

今回のプログラムは、`main()` メソッドの他に、`sqf()` メソッドが加わっている。メソッドはいくつかの処理をひとまとめにしたものだったね。プログラムは常に `main()` メソッドから始まるので、`sqf()` メソッドの場所はどこでもよい。ただ、追加的に用意されたので後ろに付け足しただけだ。先にその部分から話しておこう。

14:–16:行目の `int sqf(int x){}` が、関数 $f(x) = x^2$ の **Java** 表現である。**Java** ではメソッドと呼ぶが結局は関数である。もう少し細かく分析すると、左辺の $f(x)$ がプログラムの `int sqf(int x)` に、右辺の x^2 がプログラムの `return (x * x);` に対応している。x の平方

(square) を求める関数 (function) の意味で、メソッド名を `sqf` としただけだから、名前に特別な意味があるわけではない。数学で使う関数なら $sqf(x)$ で済むところが、プログラムでは扱う数値の型が大きな問題となる。そこで 、メソッド `sqf(x)` が整数値を受け取り、整数値を返すことを明確にするために `int sqf(int x)` と書かなくてはならないのだ。`int x` の方が受け取る変数の型、`int sqf()` の方が返す値の型を決めている。

数学では $f(x)$ の x に値が代入されたとき、右辺の x^2 により実質的な計算が行われる。プログラムも同じだ。`sqf()` メソッドに具体的な値が代入されると、`{}` 内の `(x * x)` により実質的な計算が行われる。黙っていても値を返してくれないので、15:行目の `return;` 文を使って値を返してやらなくてはならない。これが `int sqf(int x){}` の全貌である。

このメソッドが実際に使われるのは 11:行目の `System.out.println;` 文の中である。ここで **Java** にさせたいことは、値 n を受け取り n^2 の値を出力することだけだ。n の値はいつものように `nextInt()` メソッドで受け取る。画面には "input 'x' of x^2: " と表示され、値を待ち受ける。画面に x^2 と表示させるのは難しいので、`x^2` と書いて 2 乗の表現にすることが多い。ところが皮肉なことに、 **Java** では `x^2` は 2 乗の表現ではなく、まったく別の意味の操作をすることになるのだ。詳しくはずっとあとの豆旅で出会うだろう。だから x^2 は `x * x` で計算させたのである。

以上の流れを簡潔に述べると、たとえばキーボードから 3 が入力されると、**Java** は文字列を連結しようとして "3^2 = sqf(3)" を表示するよう努めるが、`sqf(3)` は `int sqf(int 3)` により 9 になって返ってくる。よって、"3^2 = 9" が表示されるのである。

TRY! 関数 $f(x) = x^3 - 2x^2 + 3x - 4$ の値を返すようにしてみよ。

TRY! このままだと `sqf()` は、整数値だけしか扱えないメソッドである。実数値が扱えるように書き換えよ。

メソッドの使い方を見て、受け取る変数が `n` に代入されるのだから、その値を受け取るメソッドも `int sqf(int n)` とすべきではないかと感じなかっただろうか。そして返される値も `return (n * n)` で返されるべきだと。もちろんそれは正しい考えだし、そうしてもプログラムは正常に動作する。だが、一般には同じ変数名を使わないようだ。

理由の一つに、関数が再利用を目的に作られていることをあげよう。メソッドに使う変数をあるプログラムと同じにすると、別のプログラムに同じメソッドを使うときに違和感を感じるものなの

だ。今回のメソッドを例にとれば、`int sqf(int n)` と書いてしまうと、`sqf()` メソッドはどうしても整数関数の印象を受けてしまう。それは、文字 n が整数変数に使われることが多いからだ。`sqf()` メソッドは実数変数でも問題なく動作する。変数を `n` から `x` に変えているのはそのためだ。

ところで、この豆旅は数学の話題に重きを置いている。そして、この付近で目にした景色は関数もメソッドも似たようなものとして映っている。この先は数学とプログラムが入り乱れるかもしれないので、関数とメソッドを同義に述べることがあるかもしれない。でも、それは豆旅の風情と思って許してもらいたい。