0.3 有理数への継承

さて、整数を使えばどんな大きな数でも、またどんな小さな負の数でも表すことができる。ただし、それは数学の中での話であって、コンピュータでは数の大きさに制限がついてまわるのである。ところで整数だけでは端数を表すことができない。端数を表すためにわれわれは**分数**や**小数**を使っているのだ。身近に分数や小数を使っていると、それらの数は自然発生的に生まれたように感じてしまうものだ。しかし、分数や小数はわれわれ人間が作り出した数なのである。

分数や小数はどちらも "数" と名乗っているものの、性格には大きな違いがある。いまでは 1/8 と 0.125 は、どちらも同じ量を表す数の扱いを受けている。けれども 1/8 は、8 の量に対する 1 の量という比を意味し 1 : 8 と表す。すなわち $1/8 = 1 : 8$ だ。一方 0.125 は、1 ほどに大きくない量の 0.1 に、0.1 ほどに大きくない 0.02 を加え、さらに 0.01 ほどに大きくない 0.005 を加え合わせた量である。どっちも同じじゃないかって？　そう、量という点ではね。

結局のところ、分数でも小数でも細かい量—これは精密な量と言ってもかまわない—を表せることに変わりがない。たとえば 10/3 や 3 という値は、**円周率**を大まかに表している。もちろん円周率というのは、直径に対する円周の比のことだ。でも、これよりも 22/7 や 3.14 とすることで、さらに 355/113 や 3.141592 とすることで、円周率をより精密に表せる。ここで注目してもらいたいのが、分数の作り方である。分数はあくまでも自然数の比で表している。一方、小数は数字を増やすことで対処しているが、これは小数に小数を足していく操作になっている。つまり、分数が自然数を継承して作られる数であるのに対し、小数は自分自身を継承している点に注意してほしいのだ。ここに分数と小数の性格の違いを見ることができるわけだ。

今日では、われわれは $a : b$ の比は a/b という分数で表すのが一般的になっている。このように整数 a, b (ただし $b \neq 0$) を用いて a/b で表すことができるすべての数を**有理数**と呼ぶことにする。有理数の呼称は慣れないと戸惑うかもしれない。a/b で表した数は rational numbers (比の数) と呼ぶが、日本語訳が有理数となっているだけのことなのだ。すると、この約束に従えば -5 は $-5/1$、0 は 0/1 と表すことができる数なので、整数も有理数であると言えるのである。

先ほど、分数と小数は同じ量を表す点で同じであると言った。しかし、分数の集合と小数の集合は同じ濃度ではない。このことはあとで示すが、その前に **Java** でも小数を扱えるようにしておきたい。

プログラム [Parrot.java] によって、キーボードからの数値入力を **Java** に受け付けてもらえるようになった。しかしそのままでは整数値しか受け付けないので、うっかり小数を入力すると期待を裏切られる。[Parrot.java] は変数 n に整数を代入するようにできているからだ。そのため、小数を扱いたいのならプログラムに若干の修正をしなくてはならない。

programming list [Numeric.java]

```
 1: import java.util.Scanner;
 2:
 3: public class Numeric {
 4:
 5:     public static void main(String[] args) {
 6:         Scanner s = new Scanner(System.in);
 7:
 8:         System.out.print("input a number: ");
 9:         double n = s.nextDouble();
10:         // float n = s.nextFloat();
11:
12:         System.out.println("an input number is " + n);
13:     }
14: }
```

解決方法は 9:行目のように nextDouble() メソッドを使う。変更はこれだけだが、小数が入力されるので n も double 型で宣言しないといけない。int n のままだと、せっかくの小数入力が整数値になってしまう。

ちなみに **Java** では、いくつかの処理をひとまとめにしたものをメソッドと呼ぶ。だから、このプログラムの 5:–13:行目のまとまりは main() メソッドと呼んでよい[1)]。nextDouble() がメソッドと呼ばれるのも、キー入力の文字を小数値として取り込むための処理がまとめられているからである。そして、それは s が持つメソッドの一つであるから、s.nextDouble() と書いて s から呼び出しているのである。え？　いつの間に s はそんな機能を持ったのかって？　それは 6:行目だ。このとき s は Scanner クラスの機能を含んで初期化されたのである。あまり正確な説明ではないけれど、豆旅は始まったばかりである。差し当たっては動くプログラムを書くことに注力して、厳密な意味は徐々に身に付けていけばよいだろう。

TRY!　与える数をいろいろ変えて実行してみよ。どのように表示されるか？　また、正の最大数・負の最小数はどのぐらいか？

1)　"main メソッド" と呼べばよいのだが、豆旅では以後、メソッドは () 付きで呼ぶことにする。

コンピュータで扱う小数は**浮動小数点数**と呼ばれる。ここで用いた `nextDouble()` メソッドは**倍精度浮動小数点**で処理するためのものである。浮動小数点数を扱う場合、もう一つ**単精度浮動小数点**で処理する `nextFloat()` メソッドがある。とくにこだわることがなければ倍精度浮動小数点を使えばよい。その名のとおり精度が高いからだ。

TRY! ちなみに **Java** のプログラムでは、`//` から行末までは**コメント**扱いとなりプログラムの実行に影響しない。`[Numeric.java]` の 9:行目と 10:行目を差し替えて実行してみよ。精度の差が分かるだろうか。