計算機プログラミングI 2005　久野クラス　＃2

久野　靖*

2005.10.14

はじめに

1週間ぶりのごぶさたでしたが、演習とかやってみていかがでしたか? アンケートで色々感想を頂いていますが、まあそれなりという感じのようですので、今後とも同様に行きたいと思います。

本日の目標

1. オブジェクトとは何かについて、および値とオブジェクトの区別についてマスターする。
2. 基本的な制御構造が使えるようになる。

1　前回の演習問題の解答例 (一部)

1.1　演習 6a

まず演習 6a であるが、疑似コードは次の通り。

- 実数 x、y を入力する。
- x+y を出力する。

で、Java では次のようになる。

```
import java.io.*;

public class r1ex6a {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("x = ");
    double x = (new Double(in.readLine())).doubleValue();
    System.out.print("y = ");
    double y = (new Double(in.readLine())).doubleValue();
    double z = x + y;
    System.out.println("x + y = " + z);
  }
}
```

ところで、演習でよくある間違いに

```
    System.out.print("x = ");
    System.out.print("y = ");
    double x = (new Double(in.readLine())).doubleValue();
    double y = (new Double(in.readLine())).doubleValue();
```

*筑波大学大学院経営システム科学専攻

とやって動かして見ると

```
% java r1ex6a
x = y =
...
```

となってしまう、というのがありますが、はまった人いませんか。「x のプロンプトを出す」「x を読む」「y のプロンプトを出す」「y を読む」というふうに**きっちりと順番に**やる必要があるわけです。

1.2 演習 6b

演習 6b は 6a とおんなじようにやればいいのだが、、、

- 実数 x、y を入力する。
- x+y、x-y、x*y、x/y を出力する。

で、Java では次のようにしてみた。

```
import java.io.*;

public class r1ex6b {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("x = ");
    double x = (new Double(in.readLine())).doubleValue();
    System.out.print("y = ");
    double y = (new Double(in.readLine())).doubleValue();
    System.out.println("x + y = " + (x+y));
    System.out.println("x - y = " + (x-y));
    System.out.println("x * y = " + (x*y));
    System.out.println("x / y = " + (x/y));
  }
}
```

自分が作ったのと違う、と思いました? もちろん、それで当然であり、どっちが正解ということはない。ちょうど日本語で同じことを言うのにさまざまな言い方があるのと同じこと。ただし、スマートだとか短いとか分かりやすいとかそういった点で違いはあるかもしれない。美観の問題! だから自分の好みもとりまぜて、考えて選ぶこと。

上の解答例に戻ると、このコードでは和、差、商、積をいったん変数に入れていない。もちろん、前回の例題のように変数に入れてもよいが、上のようにいきなり出力してもよい。一般に変数が書けるところには代りに任意の計算式を書いてもよい。[1]

あと、字下げ (左側に空白を入れること) をしない人がいるけど、これは絶対やめた方がよい。たとえば、後で出て来る話だけど

```
if(...) {          if(...) {
    ....               ....
    ....               ....
}                  }
```

左のように書いてあれば、どこまでが「もし〜ならば」の範囲かすぐ分かる。しかし右のようにべったり揃えてしまうとそれが分からないし、たとえば間違って「}」をつけ忘れたり別の場所に入れてしまうともう大混乱になる。たかがスペースキーを打つ手間を惜しんで後で間違い探しに 1 時間も掛けるのは阿呆みたいでしょ? プログラムは「美しく」書こう。

[1] ところで、式を丸かっこで囲んでいるのは、「"...." + x + y」とすると足し算は行われず全部文字列として連結されてしまうから。

1.3 演習 6c+6d

演習 6c と 6d は円錐の体積と表面積…表面積はけっこう面倒ですよね。底面の半径 r、高さ h として、まず円錐の底面の面積は πr^2。体積はこれに高さを掛けて 3 で割るだけ。表面積は展開図を考えると、底面の面積はもう分かってるから側面は…側面の半径は $\sqrt{r^2+h^2}$。で、これを l と置くと、側面になる扇型の中心角は、全円周 (ラジアンで表した中心角 2π の場合に相当) の長さ $2\pi l$ と底面の円周の長さ $2\pi r$ の比率 $\frac{r}{l}$ に 2π を掛けたもの。で、扇型の面積は半径 l の円の面積の $\frac{r}{l}$ 倍となるわけだ。以上から表面積は

$$\pi r^2 + \pi l^2 \frac{r}{l} = \pi(r^2 + rl)$$

こういうのが得意な人むけにちょっと遊ぼうと思ったのだけど…大変でしたか?

```
import java.io.*;

public class r1ex6c {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("radius = ");
    double r = (new Double(in.readLine())).doubleValue();
    System.out.print("height = ");
    double h = (new Double(in.readLine())).doubleValue();
    double l = Math.sqrt(r*r + h*h);
    System.out.println("volume = " + (3.1416*r*r*h*0.333333));
    System.out.println("surface = " + (3.1416*(r*r + r*l)));
  }
}
```

1.4 計算精度と誤差

ところで、「円周率は 3.1416 じゃないねん!」と思う人もいたようだが…計算機での計算は「電卓での計算」と同じであり、有限の桁数 (double だと 64 ビット) でしか行なえないのであり、自分で必要と思う適当な桁数を決めてその範囲でやるしかない。もっとも、3.141592653589793 くらいまでは扱えるので、それをいちいち書くのは嫌だという人のために `Math.PI` と書いてもいい (ついでに e は `Math.E`)。

その他のは省略しますが、6e や 6g などで色々計算させてて「下の桁にヘンなものがついてる! おかしい!」と思った人もいますよね。実は、「0.1」というのは 2 進数で表すと無限小数になってしまうため、これを使うと必ず誤差が出る。要するに、計算機で実数計算をした場合はそれは必ず近似値である、と思った方がよいわけ。覚えておいてください。

2 値とオブジェクトとクラス

さて、Java が扱うデータには大きく分けると「値」と「オブジェクト」の 2 種類があり、その区別は次のようになっている。

- 値 — 数値 (四則演算の対象になるもの)、文字、真偽値。四則演算程度の機能だけを持つ。単純なデータ。
- オブジェクト — さまざまな「もの」を表すデータ。込み入った構造や機能を持つことができる。オブジェクトは必ずいずれかの「クラス」に属している。つまりクラスはオブジェクトの「種類」に対応している。

たとえば、「値」は前回からやっているように、定数で表したり、四則演算を施したりできる。[2]

```
double x = 1.0 + (y - 1.0); // 値の計算の例。
```

[2] 「//の右に書いてあるのは注記 (コメント) といい、プログラムの内容について分からなくならないように覚え書きしたり目印を書いたりするのに使う。

それはそれでいいのだが、プログラムでは込み入った情報も扱いたいので、そのためには「値」だけでは力不足である。なので、もっと込み入った構造のものを表わすのに「オブジェクト」が使われる。オブジェクトは new 演算によって作り出したり、メソッド (操作) を呼び出すことでさまざまな動作を行わせることができる。

```
Dog d = new Dog("Pochi");    // new で作る。Dog はクラス名。
d.bark(3); d.run(10.0);      // メソッドを呼ぶ。
```

先の例と比べると、書き方がかなり違っていることが分かる。変数 d にはポチという犬を入れたので、そのポチに対して「3 回ワンと言え」とか「時速 10 キロで走れ」とか指示するわけである。このような書き方は Java のようなオブジェクト指向言語固有のもので、これから沢山使うことになる。なお、メソッドはオブジェクト以外にクラスが持つこともある。たとえば前回課題に出て来た平方根の計算「Math.sqrt(...)」はクラス Math が持っているメソッドを呼び出している。ついでに、オブジェクトやクラスはメソッドのほかに変数を持つこともある。たとえば上述の Math.PI はクラス Math が持っている変数である。[3]

値の種類とオブジェクトの種類 (クラス) を合わせたものを「型」という。Java ではすべての変数 (値やオブジェクトを入れておくいれもの) は

```
型 変数名;
型 変数名 = 初期値;
```

の形で宣言 (「使いますよ」と予め断わること) することになっている。値としては int(整数)、long(倍精度整数)、char(文字)、float(実数)、double(倍精度 — 精度の高い — 実数)、などがある。倍精度というのは、ビット数が多い (通常は 32 ビット、倍だと 64 ビット) なので、整数ならより大きい値まで扱え、実数なら有効桁数が多い。

一方、文字列などは複雑な構造を持つ (中に文字がたくさん詰まっている) のでオブジェクトで表す。ところが困ったことに、整数や文字などを表すクラスも別途ある。これは「値」の方はデータの変換などの込み入った機能 (つまりメソッド) が持てないため、そのようなメソッドの入れ場所としてクラスが用意されているわけである。これらの値とクラスを次に示しておく。クラス名は大文字で始まることに注意。

種別	値	クラス
真偽値	boolean	Boolean
整数	int	Integer
倍精度整数	long	Long
文字	char	Character
実数	float	Float
倍精度実数	double	Double

あと、これまでのプログラムで出て来たクラスについても説明しておく。

クラス	説明
String	文字列 (文字のならび)
BufferedReader	1 行入力機能を持った入力ストリーム
InputStreamReader	1 文字入力機能を持った入力ストリーム
InputStream	バイト単位入力ストリーム
PrintStream	画面表示用ストリーム

最後の 2 つはプログラムの上でその名前が出てこなかったが、実は System.in に InputStream オブジェクト、System.out には PrintStream オブジェクトが格納されている。

これらの解説の上で、ようやく「いつもの」入力がやっていることの説明ができる。まず、

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
```

から説明しよう。まず

```
System.in
```

これは、System.in という変数に格納している InputStream オブジェクトを参照している。次に、

[3] 書き換えはできないようになっているので、π の値が途中で変わってしまう心配はない。

```
new InputStreamReader(System.in)
```

これは、その InputStream オブジェクトをパラメタとして、新しく InputStreamReader オブジェクトを作り出している。そして、

```
new BufferedReader(new InputStreamReader(System.in));
```

これでその InputStreamReader オブジェクトをパラメタとして BufferedReader オブジェクトを作り出している。最後に、

```
BufferedReader in = ...
```

これで BufferedReader 型の変数 in を用意し、そこに作り出した BufferedReader オブジェクトを入れているわけだ。

さて、では次にこの行はどうか?

```
double f = (new Double(in.readLine())).doubleValue();
```

ここでは、まず

```
in.readLine()
```

で、BufferedReader オブジェクトのメソッド readLine() を呼び出す。このメソッドは入力先 (この場合はキーボード) から 1 行ぶんを読込み、それらをまとめた String オブジェクト (文字列) を返す。次に

```
new Double(...)
```

で、読み込んだ String オブジェクトが表現しているのと同じ値を持つ Double オブジェクトを作る。次に

```
(...).doubleValue()
```

を呼ぶことで Double オブジェクトが表している倍精度実数の「値」を double(小文字!) 型の値として取り出す。そして

```
double f = ...
```

でその値を変数 f に格納するわけである。このように、Java では「オブジェクト」の生成やさまざまなメソッド呼び出しが組み合わさってプログラムが進んで行くことが多い (ほかに値の演算ももちろんあるが)。

3 枝分かれ

さて、ここまでに説明した機能だけでは、上から順に動作を実行していくようなプログラムしか作れない。それでは不便なので、枝分かれを学ぶ。たとえば、「入力 x の絶対値を計算する」例題を考えよう。しばらくの間、条件としては数値の大小比較 (> — より大、>= — 以上、< — より小、<= — 以下、== — 等しい、!= — 等しくない) だけを考える。[4]

- 数値 x を入力する。
- もし $x < 0$ ならば、
- 　　$abs \leftarrow -x$。
- そうでなければ、
- 　　$abs \leftarrow x$。
- 枝分かれ終わり。
- 数値 x とその絶対値 abs を出力する。

なお、「←」は変数に値を入れることを意味する (「〜と置く」と書くのが面倒だから)。ではこれを Java にしてみよう。

[4] Java では「!」は「否定」を表すのに使っている。階乗ではないので注意。

```
import java.io.*;

public class Sample2 {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value X = ");
    double x = (new Double(in.readLine())).doubleValue();
    double abs;
    if(x < 0.0) {
      abs = -x;
    } else {
      abs = x;
    }
    System.out.println("value: " + x + " absolute value: " + abs);
  }
}
```

変数 abs を宣言だけしているのが目立つ。これは、変数の宣言を { ... } の中に書くとその範囲内でしか有効にならないので、if の外側に書かなければいけないのでしかたがない。

ところで、同じプログラムだがこう書いたらどうだろう?

- 数値 x を入力する。
- $abs \leftarrow x$。
- もし $x < 0$ ならば、
- $\quad abs \leftarrow -x$。
- 枝分かれ終わり。
- 数値 x とその絶対値 abs を出力する。

このように、「そうでなければ」の部分で何もすることがなければ「そうでなければ」以下を書かないでよい。Java プログラムでも同様。

```
import java.io.*;

public class Sample2b {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value X = ");
    double x = (new Double(in.readLine())).doubleValue();
    double abs = x;
    if(x < 0.0) {
      abs = -x;
    }
    System.out.println("value: " + x + " absolute value: " + abs);
  }
}
```

先のプログラム例と、どちらが好みですか?

ところで、「わざわざ変数 abs を用意しなくても、変数 x の符号を反転させたら?」と思ったかも知れない。確かに、このプログラムの場合はそれでもよいけれど、場合によっては後で「元の値」と「絶対値」の両方が必要になるかも知れない。一般に、1 つの変数を複数の意味 (「x」と「x の絶対値」) で使うのは混乱する (書く人も、読む人も) からやめた方がよい。ただし、x はすぐに絶対値に直して、あとは絶対値だけしか使わない、ということなら x を直接書き換えてもよい。

要するに、プログラムの書き方は「どれが絶対正解」ということはなく、場面ごとに何がよいかが違ってくるし、人によっても基準が違うところがある。早く自分の基準を見つけよう!

演習 1 上の絶対値計算プログラムの好きな方のバージョンを打ち込んでそのまま動かせ。

演習 2 枝分かれを用いて、次の動作をする Java プログラムを作成せよ。

a. 2 つの異なる実数 a、b を入力し、より大きい方を表示する。
b. 3 つの異なる実数 a、b、c を入力し、最も大きいものを表示する。やる気があったら 4 つでやってみてもよい。
c. 実数を 1 つ入力し、それが正なら「positive」、負なら「negative」、零なら「zero」と表示する。

4 繰り返し

枝分かれができても、まだプログラムで大したことができる気がしない。計算機の特徴は、どんなつまらない単純作業でもいくらでも繰り返してくれることにあるので、繰り返しが使えるとプログラムもぐっと役に立つ。先の箱詰め問題をプログラムにしてみよう。まず疑似コードを再掲する。

- タイルの個数 n を入力する。
- 箱の一辺 l を 0 とおく。
- $l^2 < n$ が成り立つ間繰り返し:
- $l+1$ をあらたに l とおく。
- 以上を繰り返す。
- 一辺の長さ l を出力する。

これを Java にしてみよう。

```
import java.io.*;

public class Sample3 {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("value N = ");
    int n = (new Integer(in.readLine())).intValue();
    int l = 0;
    while(l*l < n) {
      l = l + 1;
    }
    System.out.println("# of balls: " + n + " edge length: " + l);
  }
}
```

書き方そのものは繰り返しの方が枝分かれより簡単なくらいだが、「繰り返す」という考え方がマスターできないとよく分からないだろう。なお、ここでは n も l も整数なので `int` 型と指定し、入力の変換にも `Integer` クラスと `intValue()` を使っている。

演習 3 次の動作を行う Java プログラムを作成せよ。

a. 正の整数 x を入力し、x が奇数か偶数かを表示する。ただし割り算演算は使わないこと。
b. 正の整数 x と y を入力し、$x \times y$ を表示する。ただし掛け算演算は使わないこと。
c. 正の整数 n を入力し、2^n を表示する。(ついでに n がいくつくらいまで計算できるか調べよう。)

5 整数と実数

先の例題で int 型が出て来たついでにもうちょっと。数学の世界では整数とは実数の部分集合だが、計算機の世界では全然違う! 整数型は次のような特徴を持っている。

- 誤差がない (扱える最大の値と最小の値はある)。
- 実数より計算の効率がよい。
- 「N 番目」を指定するような場所で使うことができる。

だから、計算するといっても「ここは整数がいいか?」「ここは実数がいいか?」を絶えず意識しておく必要がある。あと、実数型が必要なところで整数型を使うと自動的に実数に変換してくれる (整数を実数に変換しても情報が失われることはないから)。

```
int i = 12345;
double f = i; ←自動的に変換
.... i + 3.14 ... ←こういうのも自動変換
```

逆に実数を (小数点以下を切捨てて) 整数に変換するには「(int) 式」という形で指定しないといけない。これを「キャスト」(強制型変換) と呼ぶ。

```
double g = 3.1416;
int k = (int)g; ←こうしないと javac でエラーに
.... 2 * (int)(g / 7.0) ... ←こういうのもキャスト
```

覚えておいてください。

6 制御構造の組み合わせ

少し込み入ったプログラムになると、ある制御構造 (枝分かれ、繰り返し) の内側にさらに制御構造を入れることになる。たとえば、

- もし〜であれば、
- 　条件〜が成り立つ間繰り返し:
- 　　○○をする
- 　以上を繰り返し。
- 枝分かれ終わり。

だと次のようになるわけである。

```
if( ... ) {
    while( ... ) {
        ...
    }
}
```

このように規則に従って要素を組み合わせて行くことで (単に並べるのも組み合わせ方のうち)、いくらでも複雑なプログラムが作成できる。これはちょうど、簡単な規則と単語からいくらでも複雑な文章が (日本語や英語で) 作れるのと同じである。

演習 4 次の疑似コードは 2 つの数の最大公約数を求めるものである。これを Java に直して動かせ。またなぜこれで最大公約数が求まるのかを説明せよ。**(注意! 「これは〜法である」と固有名詞だけ挙げるのは固くお断りします。何の説明にもなっていない。特に「ユークリッドの互除法」は最悪。だって割り算を使っていないでしょ?)**

- 整数 x、y を入力する。
- $x \neq y$ の間繰り返し、

- 　もし $x > y$ であれば、
- 　　$x \leftarrow x - y$。
- 　そうでなければ、
- 　　$y \leftarrow y - x$。
- 　枝分かれ終わり。
- ここまで繰り返し。
- x を最大公約数として出力する。

演習 5 次の疑似コードは入力 x(ただし $x > 1$) の平方根を求めるものである。これを Java に直して動かせ。またなぜこれで平方根が求まるのかを説明せよ。

- 実数 x を入力。
- $a \leftarrow 0.0$、$b \leftarrow x$。
- $(b - a) > 0.0000001$ である間繰り返し、
- 　$c \leftarrow \frac{a+b}{2}$
- 　もし $c^2 > x$ ならば、
- 　　$b \leftarrow c$
- 　そうでなければ、
- 　　$a \leftarrow c$
- 　枝分かれ終わり。
- ここまで繰り返し。
- a の値を平方根として出力。

演習 6 「正の整数 N を読み込み、N が素数か否かを教えてくれる Java プログラム」を書け。まず疑似コードを書き、それから Java に直すこと。(ヒント: N が素数ということは、N を 2〜$N-1$ のいずれで割っても余りが出るということ。剰余は演算子「%」で計算できる。たとえば「7 % 4」は 3。)

演習 7 「正の整数 N を読み込み、N 以下の素数をすべて打ち出す Java プログラム」を書け。待ち時間 10 秒以内でいくつの N まで処理できるか調べて報告せよ。(もちろん N が大きくなるように工夫してくれるとなおよい。)

A　本日の課題 2A

今日は「演習 2」で動かしたプログラムを含む小レポートを久野まで電子メールで送ってください。具体的な内容は次の通り。

1. Subject: は「Report 2A」とする。[5]
2. 学籍番号、氏名、投稿日時を書く。
3. 「演習 2」で動かしたプログラムどれか 1 つのソース。
4. 以下のアンケートの回答 (簡単でよい)。

   Q1. プログラムを打ち込んで動かすのに慣れましたか?

   Q2. 自分にとって次の「難しいポイント」は何だと思いますか?

   Q3. 本日の全体的な感想と今後の要望をお書きください。

[5]注意! 1 バイトコード (いわゆる半角) で、大文字小文字もこの通りに、符号化なしで! プログラムで処理するので、この通りでないと間違って処理される可能性がいくらか高くなります。もちろんチェックはしますが。

B　次回までの課題 2B

次回までの課題は「演習 3(a/b/c)」〜「演習 7」までの (小) 課題からプログラムを 2 つ以上作ること。ただし「演習 4」「演習 5」「演習 6」「演習 7」のうち 1 つは必ず含まれて欲しい。

レポートは授業開始 10 分前 (10:30) までに、上記と同様に久野までメールで送付してください。具体的な内容は次の通り。

1. Subject: は「Report 2B」とする。
2. 学籍番号、氏名、投稿日時を書く。
3. 選んだプログラム 1 つのソース。
4. その簡単な説明。
5. 選んだプログラムもう 1 つのソース。
6. その簡単な説明。
7. 下記のアンケートの回答。

    Q1. 枝分かれや繰り返しの動き方が納得できましたか?
    Q2. 枝分かれと繰り返しのどっちが難しいですか? それはなぜ?
    Q3. 課題に対する感想と今後の要望をお書きください。