計算機プログラミングI 2005　久野クラス　＃1

久野　靖*

2005.10.7

はじめに

こんにちは、久野です。今週から「計算機プログラミングI(久野クラス) (金曜2限)」を開始します。「計算機プログラミングI」は、「すでに計算機の利用方法を知っている学生を対象に、プログラミング入門教育を行う」ことを目標とした科目です。簡単に言えば:-)、プログラミングができるようになるための科目、ということですね。

本クラスの基本方針と目標水準

久野の基本的な信念は、「プログラミングができるようになるためには、コードを自分で書いて動かす経験を一定量積むことがどうしても必要である。だからプログラミング入門科目はある程度までは『トレーニング』の科目となる」というものです。本クラスもこの方針に沿って行い、授業時間のほかに時間外に (次の授業までの間に) やって頂く課題を出しますので、『トレーニング』を行うのが嫌だと思う人 (努力するのが嫌である人、または授業時間外に時間を費したくない人) はこのクラスは選択しないでください。

次に、本クラスでどの程度までの水準を目指すかということですが、次のようなレベルを考えています。

- 簡単な (自分で計算手順が思い付く程度の) 計算であればその計算を実行する Java プログラムを自分で書いて動かせるようになる。
- 上記の程度のプログラムに GUI 部品を使ったインタフェースを持たせることができるようになる。
- 簡単な (自分で描き方が思い付く程度の) 絵であればその絵を表示する Java プログラムを自分で書いて動かせるようになる。
- 上記の程度の絵を時間とともに変化させられる (アニメーション)。
- Java 言語およびオブジェクト指向の基本概念、基本用語を一通りマスターする。

過去の例では、この水準が「高すぎて最後までついて来られない」人の割合は最大 2〜3 割くらいですが、その 100%が能力の問題ではなく「トレーニングに時間を割かなかったから」であると判断しています。

平均的な学生さんの場合で、1 週間あたり授業時間以外に授業時間の 3 倍程度、すなわち 4 時間くらいは自分の時間をつかって課題をやる必要があると思います。ただしこの時間自体は、それぞれの学生さんの「個性」(決して時間が掛かるのが劣っているということはなく、時間は掛かるけどすばらしいプログラムを書いてくれる人も沢山います) によって大幅に変動しますので、そのことも承知しておいてください。もちろん、たまたま忙しい期間があってその週だけ手抜きすることは仕方ありませんが、毎週手抜きした場合はこのクラスを取る意味がなくなります。

皆様の多くが気にされる単位についてですが、過去において「努力する気はないけれど単位が取れそうだからこのクラスを選択してしまう」人が結構いました。それらの人にとってはそれは結局悪い選択だったと思います (ついていくのが大変で、挫折して、嫌な想い出だけが残るという…)。ですから、しつこいようですが、あくまでも努力する気がある人だけ選択してください。実際に努力をしていただければ、それでもなお単位取得基準に達しないような学生さんはいないと考えています。

*筑波大学大学院経営システム科学専攻

警告! 本クラスの対象学生について

上記のように、本科目は初心者を対象としてプログラミングを最初から学ぶものですが、例年、既にある程度プログラミングができる人も本科目を選択されています。それは構いませんが、特に本クラスの場合、「既に知っていることの有利」は期待しないでください。各回の課題についても、原則としてその回までに講義で取り上げた範囲の機能を用いて工夫するようお願いします (あくまでお願いですが)。

また、課題の評価に際しては「色々考えて工夫して頂いているか」も大きなポイントですが、過去の経験から言ってこの面においては初心者の方が「初心で臨める」ため全般に有利だと言えます。経験者が頑張ると長くて複雑なプログラムができてしまうことが多いのですが、評価時に長く複雑なプログラムを熟読する時間はあまりないので、「そこそこ」に読んで採点するしかないことが多く、この面でも経験者の方が不利かと思います。経験者の方がこのクラスを選ぶ場合は予め了承しておいてください。

警告! 履修人数調整について

「計算機プログラミング I」は多くのクラスが設置されており、そのどれを履修するかについては定員の調整等が必要になる可能性があります。詳しくは (科目の目的等も含めて)

`http://www.edu.c.u-tokyo.ac.jp/edu/cp1.html`

に案内がありますのでよく読んで登録してください。調整システムについて知らないで不利を被っても非常勤である久野には何ともできない場合があります。大学生としての自己責任でお願いします。

警告! 本クラスで使用する数式について

本クラスでは例題等において、次の程度の数学的・物理学的知識、および対応する数式を使用します。この科目は数学や物理ではないので、その内容を理解していなくても「公式をそのまま適用」できれば大丈夫ですが、数式アレルギーのある方はご注意ください (つまり嫌なら別のクラスに行きましょうということ)。

- 距離の公式 — 平面上の点 (x_1, y_1) から (x_2, y_2) までの距離は $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ で計算できる。
- ラジアン — 角度の単位として、度数 (直角が 90 度、全周が 360 度) で表す方法のほかに、ラジアンという単位もある。360 度は 2 π ラジアンである。以下では角度をラジアンで表す。
- 三角関数 — 長さ r の線分を図 1 のように、X 軸から反時計周りに θ 度回転させ、一端を原点に置き、他端の座標を (x, y) とすると、$x = rcos\theta$、$y = rsin\theta$ である。この関係は θ がいくつでも (負でも、0〜2 π でも、2 π を超えても) 成り立つ。また、原点を (x_0, y_0) に平行移動して $x = x_0 + rcos\theta$、$y = y_0 + rsin\theta$ という形で使うこともある。
- 逆三角関数 — 図 1 を逆に使って、(x, y) から θ を求めたければ、$\theta = atan\frac{y}{x}$ を使う。この場合 (x, y) は任意の点でよく (ただし x は 0 でないこと)、θ は 0〜2 π の範囲で求まる。これも原点を (x_0, y_0) に平行移動して $\theta = atan\frac{y-y_0}{x-x_0}$ という形で使うこともある。
- 等速度運動 — 点が時刻 0 に (x_0, y_0) の位置にあり、X 軸方向の速度が v_x、Y 軸方向の速度が v_y で等速度運動しているとすると、時刻 t における位置 (x, y) は $x = x_0 + v_x t$、$y = y_0 + v_y t$ で表わされる (または等速度でなくても、ごく短い時間 dt の間についてはこの式で近似できる)。
- 等加速度運動 — 加速度 (速度を変えるような力) が X 軸方向に α_x、Y 軸方向に α_y の一定値で働いているとすると、時刻 0 に速度が v_{x0}、v_{y0} だった場合、時刻 t においては X 軸方向の速度は $v_x = v_{x0} + \alpha_x t$、Y 軸方向の速度は $v_y = v_{y0} + \alpha_y t$ となる。

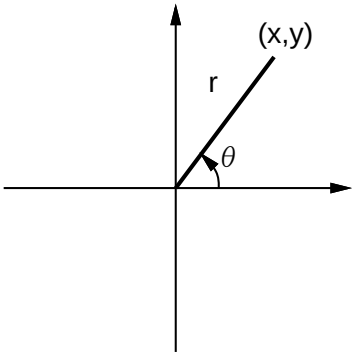

図 1: 三角関数

Web

このクラスの Web サイトは

`http://lecture.ecc.u-tokyo.ac.jp/~kuno/cp05/`

です。**ここに掲示等を出しますからこまめにチェックしてください**。出席/レポート課題等は久野宛のメールで提出してもらいますが、**それらのメールは原則としてここで公開**します。予め了承してください (公開されて困る内容は出席/レポート課題としては送らないこと)。なお、出席/レポートメールについては「`@@@`」(アットマーク 3 つ) で始まる行は削除してから掲載しますので、自分の名前を書く行にはこのおまじないをつけることを推奨します。たとえば次のような感じです。**アットマークは必ず 8 ビット文字 (いわゆる半角) を使うこと。**

```
@@@ 氏名: 久野　靖
えーと、このレポートは…（以下略）
```

また、皆様からの率直なご意見ご感想を伺いたいので、相談サーバにある掲示板を積極的に活用してください。質問 (Q&A) 用と雑談用があります。ハンドルでの書き込みも構わないそうですが、荒らし的行為 (他人のハンドルで書いたり毎回ハンドルを変える等) はやめて欲しいそうです。上記のページからもリンクが張ってあります。

```
http://www.sodan.ecc.u-tokyo.ac.jp/cgi-bin/qbbs/view.cgi : Q&A
http://www.sodan.ecc.u-tokyo.ac.jp/cgi-bin/sbbs/view.cgi : 雑談
```

参考書

このクラスでは毎回資料のみを使用しますが、参考書が欲しいという場合は次の本をおすすめします。

久野 禎子, 久野　靖, Java によるプログラミング入門, 共立, 2001. ISBN4-320-02968-2, 2800yen.

授業の内容とまったく同じではありませんが、ある程度共通部分がありますし、本になっているということで丁寧な説明が書かれているつもりです。

科目の運営方針

このクラスはこれまでにプログラミング経験がない人を前提として進めますが、プログラミングを身につけるのに時間内の演習だけでは絶対的に時間が不足です (人によっては…経験者とか、この手の才能のある人は…大丈夫かも知れませんが)。このため、次の方針を取ります。

- 講義時間の外に自発的に演習を行って頂くことを前提とします (宿題を出します)。
- 毎週講義のときに出席点兼用の小レポートを出し、それとは別に次回までの宿題の小レポートを出します。ただし授業の具合いによっては出さないこともあります。
- さらに、冬休みに大レポート課題を課します。
- 試験は行わず、点数は大小レポートのみによってつけます。

利用環境ですが、基本的に Java のコンパイラ等はここの Mac OS-X システム上の javac、ブラウザは同じく Mac OS-X 上の Safari ないし Mozilla を前提とします。ただし、Java やブラウザはさまざまなマシンで動かすことができますから、どこかよそでやっても、自宅でやってもそれは自由です。なお、Java にはいくつかのバージョンがありますが、この講義では JDK 1.4 以降を前提とすることにします (昨年度は 1.1 以降としていたので、一部の内容が変化すると思います)。

さて、具体的な講義内容ですが、今年はおおよそ次のようなロードマップでやりたいと考えています (年内ぶん)

- プログラムの基本構造 — 変数、演算、代入、選択、反復
- オブジェクトの利用
- アルゴリズムとデータ構造
- アプレットプログラミング
- クラスとメソッド
- インタフェースとクラス階層
- GUI と GUI 部品
- アニメーション

上記のは概要で、細かいところはやりながら決めていく予定ですが、ともかくアプレットで絵を作って動かせることは押さえるつもりです (それを冬休み課題に予定しています)。では半年間、よろしくお願いいたします_o_。

1 プログラムとアルゴリズム

今までは皆様は、計算機である処理をしたければ、それを行うためのコマンドが既に存在していて、その名前を言うだけでいい、という形で計算機を使って来たはずである。しかし現実には、自分のやりたいことすべてが予めコマンドとして用意されている、などということはもちろん不可能である。

ではどうするか? それには、「どのような処理をしたいか」という情報を、計算機に与えればよい。この情報のことを (散々聞いたことがあると思うが)「プログラム」という。実は「既にあるコマンド」というのは、「あらかじめ計算機に入力されて蓄えられているプログラム」に他ならない。

言い替えれば、計算機というのは実はプログラムに従って情報を処理する装置なわけである。そして、どのような処理であっても (たとえその計算機が製造された時には夢想だにされなかったようなものでも)、その処理方法をプログラムとして与えさえすれば処理できるようになる、というところに計算機の特徴があるわけである。

なお、ここで「プログラム」と「アルゴリズム」ないし「手順」の違いを整理しておく。「アルゴリズム」ないし「手順」といった場合、それは加工の方法自体をいう。例えば駒場地区正門から渋谷駅まで歩いて行く行き方、といったものである。一方、「プログラム」といった場合はアルゴリズムを計算機に与えられるような形に表現したものをいう。例えば道筋を他人に教えるためには、行き方を記した地図やメモといった具体的な形に表現する必要がある。

演習 1 「アルゴリズム」と「プログラム」、「行き方」と「行き方を示した地図」のような関係にあるものの例を他にも挙げてみよ。

2　計算機のためのアルゴリズム

計算機で使う「アルゴリズム」ないし「手順」は、何かを求めるための具体的な計算 (ないし情報の加工) 方法でなければならない。たとえば、海外のニュースなどを見ていると気温が華氏で表示されているので、それは摂氏では何度かな、と知りたくなる。それを求めるには、華氏の温度を f として、

$$c = \frac{5}{9}(f - 32)$$

により値 c を求めればよい。これも立派なアルゴリズムである。

ところで、何をもって「具体的」というかは実は簡単ではない。たとえば、n 個の正方形のタイルを (すき間があってもいいから) 正方形の箱 (ただし 1 辺の長さはタイルの 1 辺の整数倍) に平らに並べて入れたければ、その箱の 1 辺の長さはタイルの 1 辺の

$$l = \lfloor \sqrt{n} \rfloor$$

倍であればいい。ただしこれは、「切り上げ」とか「平方根」とかの計算手順が具体的にわかっていれば、である。もし加減乗除だけしか使えないのなら、例えば

$$l^2 \geq n$$

なる最初の l が見つかるまで $l = 0, 1, 2, 3, 4, \ldots$ を順に試していく、という手順を使うことになろう。

実は、四則演算も「具体的」かどうかは議論があってよい。たとえば小学生はどうやって四則演算をやるかの手順を一生懸命憶えさせられるわけである。しかし幸いなことに、計算機には四則演算のための機能がもともと備わっているから、それにお願いすることにして、まずは四則演算は「具体的」だということにする。

演習 2 仮に、かけ算と割り算は「具体的でない」(つまり使えない) ものとする。足し算と引き算と数の大小比較は使ってよいとして、次のことをするアルゴリズムを考えよ (ただし x、y は正の整数とする)。

a. 数 x が奇数か偶数かを判定する。
b. 数 x と y の積を求める。
c. 数 x と y の最大公約数を求める。

3　アルゴリズムと疑似コード

前節まででは、計算されるべきデータはなぜか x だの f だのという変数に入っていたかのように書いていたが、ご存じの通り、計算機ではキーボードからデータを打ち込まなければプログラムに情報が伝わらないし、プログラムの中でいくら計算が終わっても画面に表示されなければ何が何だか分からない。すなわち、入力 (人間から計算機にデータを渡すこと) と出力 (計算機から人間にデータを渡すこと) も明確に示さなければならない。

しかも、何と何を入力する、というだけではだめで、まずこれを入力し、次にこれを入力し、というふうに順番まで明確に示さなければ何をどの順で入力してよいかも分からなくなってしまう。出力も同様である。

このため、計算機のためにアルゴリズムを記述するときは、(1) 各段階ごとに、(2) 入力、出力、計算の動作などを明確に、記述する必要がある。たとえば華氏摂氏変換を考えてみよう。

- 華氏の温度 f を入力する。
- $c = \frac{5}{9}(f - 32)$ により値 c を求める。
- 摂氏の温度 c を出力する。

このような段階になる。こういう書き方のことを「疑似コード」と呼ぶ (「コード」というのは「プログラムの断片」を意味しているが、これはあくまでも日本語だからプログラムではなく、「のようなもの」なので、「疑似コード」)。

華氏摂氏変換では処理が 1 直線だったが、箱詰め問題の場合には「繰り返し」が必要になる。

- 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。

ここで注目して欲しいのは、「$l+1$ をあらたに l とおく」という部分で、つまり l というのは「変数」とは言っても方程式のように 1 つの値を持っているというより、計算の途中で自由に値を書き換えられる「入れもの」みたいなものだという点である。

もう 1 つのポイントは変数と繰り返しの関わり方で、$l^2 < n$ が成り立つ間繰り返し l を増やして行くのだから、繰り返しが終わったときは $l^2 < n$ が成り立たなくなって、ということは $l^2 \geq n$ が成り立っていて、しかも l はこの条件が成り立つ最小の整数になっている、ということである (だって 1 つずつ増やして試して行くのだから)。この辺の感覚が分かることが、プログラミングのコツだと言ってよいだろう。

演習 3 演習 2 で考えたアルゴリズムを、入力や繰り返しなどを含んだ計算機用のアルゴリズム (疑似コード) に書き直してみよ。

4 アルゴリズムとプログラミング言語

「プログラム」とは、アルゴリズムを実際に計算機与えられる形で表現したものであり、その具体的な「書き表し方」がプログラミング言語である。「言語」という名前はついているが、プログラミング言語は「日本語」や「英語」などの「自然言語」とは違って、あくまで計算機に読み込ませて処理できることが前提の「人工言語」であり、そのため書き方も杓子定規なのがふつうである。

プログラミング言語も、すでにご存じと思うが、さまざまな特徴を持つさまざまなものが使われている。ここでは Java と呼ばれる言語を用いる。Java は、C や C++といった言語に比べると「型とか例外とかについてきっちり書かせることで、お行儀のよいプログラミングを行う」という特徴があるので、その意味では入門教育にも向いていると考え、ここで採用するものである。

では、先の華氏と摂氏の変換プログラムを Java で記述してみる。

```
import java.io.*;

public class Sample1 {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("degree F = ");
    double f = (new Double(in.readLine())).doubleValue();
    double c = (5.0 * (f - 32.0)) / 9.0;
    System.out.println("degree C = " + c);
  }
}
```

先の疑似コードと比べてみると、いろいろ細かい記述が増えているので、順次説明しよう。

1. Java のプログラムは「クラス」と呼ばれる単位の集まりである。もっとも単純なプログラムはクラス 1 つだけだから、次の形をしている。

   ```
   public class クラス名 {
     中身...
   }
   ```

2. プログラムとして実行するクラスは、main というメソッド (手順ないしアルゴリズムを記述する単位) を持たなければいけない。そしてそれは次の形をしている。

```
public class クラス名 {
  public static void main(String[] args) throws Exception {
    手順の中身...
  }
}
```

3. キーボードからの入力ではだいたい「1 行読む」という機能を使うことが多いのだが、なぜか Java の標準の入力機能ではこれがついていないので、以下の例題ではすべて BufferedReader というクラスの 1 行入力機能を使う。このクラスは java.io パッケージに入っているので、先頭に「import java.io.*;」という指定 (おまじない) が必要である。また、まず BufferedReader を in という変数に入れて置くことにするので、併せて次のようになる。

```
import java.io.*;

public class クラス名 {
  public static void main(String[] args) throws Exception {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    本来の中身...
  }
}
```

BufferedReader は 1 文字単位の入力を行うクラス InputStreamReader を利用し、InputStreamReader は System.in に入っている InputStream を利用することになる。

4. ここからいよいよ手順の中身になる。まず華氏の温度 F を入力してもらうのだが、何を入力するのか分からないと利用者に不親切なので「これこれを入力してね」というメッセージを最初に表示する。

```
System.out.print("degree F = ");
```

System.out.print(...) というのは文字列を行かえなしで画面に表示する。個々の動作は最後に ; を入れて区切る。動作は 1 つずつ順に実行されていく。

5. 次に 1 行ぶんの文字列をキーボードから読み込み、それを実数値に変換して変数 f に入れる (詳しい説明はちょっと後まわし)。

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

6. f の値を元に摂氏の温度を計算し、実数変数 c に入れる。

```
double c = (5.0 * (f - 32.0)) / 9.0;
```

なお、Java など普通のプログラミング言語では数式は 1 行に書かないといけないため割り算には「/」をつかい、適宜かっこでくくる。また変数名も 1 文字とは限らないので掛け算も「*」で表す。ついでながら、「%」という演算子もあり、これは「剰余」(割った余り) を計算する。

7. 結果を表示する。System.out.println(...) は前述の System.out.print(...) とほぼ同じだが、行替えをおこなう。

```
System.out.println("degree C = " + c);
```

なお、「+」は両辺のどちらかが文字列 ("...") のときは足し算ではなく文字列の連結演算になり、他方も文字列に変換される。

以上で手順の中身はおしまいである。

要は、プログラミング言語というのは計算機に対して実際にアルゴリズムを実行する際のありとあらゆる細かい所まで指示できるように決めた形式であり、だからプログラムのどこか少しでも変更すると計算機の動作もそれに相応して変わるか、(もっとよくあることだが) そういう風には変えられないよ、と怒られることになっている。いくら怒られても偉いのは人間であって計算機ではないのだから、そういうものだと思って許してやって頂きたい。

5 動かしてみよう!

では実際にこれを計算機で動かそう。まず、Terminal アプリケーションを起動してコマンドが打ち込める窓を出す。以下、コンパイル等はすべてコマンドを打ち込んで実行させることを前提とする。

まず、エディタ (Emacs でもテキストエディットでもそれ以外のものでも、好きなものでよい) で上と同じ内容を「Sample1.java」というファイルに打ち込む。Java プログラムを入れるファイルはかならずクラスの名前と大文字小文字まで含めて同じにして、その後に「.java」をつけた名前にしなければならない。

次にこのプログラムを実行に適した形に変換する。これをコンパイルする、という。なお、コンパイルする前の (人間が読める形の) プログラムを「ソースプログラム」と呼ぶ。Java のコンパイルには javac コマンドを使う (システムによってはちょっと時間が掛るかも)。

```
% javac Sample1.java
%
```

なお、「%」はコマンドプロンプト (次のコマンドを受け付けますよ、という印のつもり。コンパイルする時に、プログラムの形に整っていない点やおかしい点があればメッセージがでる。何もメッセージがでなければ成功で、Sample1.class という出力ファイルができている。

次に、このプログラムを実行するには java というコマンドを使う。なお、java コマンドではクラス名の部分だけを指定すること。実行が始まるとすぐ最初のメッセージがでて来るので、華氏の温度を入力してあげよう。

```
% java Sample1
degree F = 100    ← 「100[RET]」と入力した
degree C = 37.777777777778
%
```

苦労のわりにはあんまり大したことはない感じだが、まあ初心者の第 1 歩ということで、そうがっかりしないで戴きたい。

演習 4 例題の華氏摂氏変換プログラムをそのまま、打ち込んで実行させてみよ。入力に数字でないものを入れるとどうなるかも試せ。

演習 5 動いたら、"..."の中身を別のものに変更したり、System.out.println を 2 回行うなど、プログラムを直して再度実行し、変更の結果が反映されることを確認せよ。

演習 6 次のような動作をするプログラムを書いて動かせ。

a. 2 つの実数を入力し、その和を出力する。
b. 2 つの実数を入力し、その和、差、積、商を出力する (さまざまな値について計算し、何か気がついたことがあれば述べよ)。
c. 円錐の底面の半径と高さを入力し、体積を出力する。
d. 円錐の底面の半径と高さを入力し表面積を出力する。

e. 実数 x を入力し、x を 10 で割った結果、および x の 0.1 倍を出力する (もし何か気がついたことがあれば述べよ)。
f. 摂氏の温度 c を入力し、華氏の温度を出力する。
g. 実数 x を入力し、x の平方根を出力する (さまざまな値について計算し、何か気がついたことがあれば述べよ)。
h. 実数 x、y を入力し、剰余演算子 `x % y` の結果を出力する (さまざまな入力について試し、剰余演算子がどのような結果を返すかまとめよ)。
i. その他、自分が面白いと思う計算を行うプログラムを作って動かしてみよ。

なお、x の平方根は `Math.sqrt(`x`)` で計算できます。

6 うんちく: 値とオブジェクト

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) と、文字。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。クラスによって定義され (てい) る。言い替えればクラスとはオブジェクトの種類である。

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」といい、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;
```

の形で宣言することになっている。

値としては `int`(整数)、`char`(文字)、`float`(実数)、`double`(倍精度 — 精度の高い — 実数)、などがある。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オブジェクトも別途ある。これは「値」の方はデータの変換などの込み入った機能が入れられないため。そのような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`
文字	`char`	`Character`
実数	`float`	`Float`
倍精度実数	`double`	`Double`

あと、先のプログラムでで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` は `InputStream` オブジェクト、`System.out` は `PrintStream` オブジェクトを表している。

オブジェクトを作るには特別な演算 `new` を使って

```
new クラス名(...)
```

とする。かっこの中は空っぽのこともあるし、作るに当って必要なデータを渡すこともある。

オブジェクトは値にない特徴として、「メソッド」を持てる。たとえば BufferedReader オブジェクトの readLine() メソッドを使うと、入力元から 1 行を読み込んでその内容を文字列オブジェクト (String オブジェクト) として返してもらうことができる。メソッドを呼ぶときは

```
オブジェクト.メソッド名(パラメタ…)
```

という形を使う。パラメタは空っぽでもよい。ついでながら、クラスに対してもメソッドを持たせることができる。こちらは

```
クラス名.メソッド名(パラメタ…)
```

で呼び出す。これらの解説の上で、ようやく残っていた説明ができる。

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

ここでは、まず

```
in.readLine()
```

で、変数 in に格納されている 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 行ぶんの文字を読み、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、プログラミング言語では計算機に対するさまざまな/多様な指令をけっこう「ぎゅっと」詰まった形で指定することができる。

A　本日の課題 1A

今日は「演習 5」または「演習 6」で動かしたプログラム (どれか 1 つでよい) 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。メールアドレスは

```
kuno    (ECC のシステムから)
kuno@mail.ecc.u-tokyo.ac.jp  (外部から、「その他」の注意参照)
```

です。具体的な内容は次の通り。

1. Subject: は ASCII(いわゆる半角) 文字で「Report 1A」とする。
2. 学籍番号、氏名、投稿日時を書く。
3. 「演習 5」「演習 6」で動かしたプログラムどれか 1 つのソース。コピー&ペーストなどで挿入すること (エンコードされた添付ファイルはいちいち解読する手間が掛けられないので避けてください)。
4. 以下のアンケートの回答 (簡単でよい)

   Q1. プログラム、って恐そうですか? 第 2 外国語と比べてどう?
   Q2. Java 言語のプログラムを打ち込んで実行してみて、どのような感想を持ちましたか?
   Q3. 本日の全体的な感想と今後の要望をお書きください。

B　次回までの課題 1B

次回までの課題は「演習 6」の (小) 課題 (ただし 1A で提出したものは除外) からプログラムを 2 つ以上作ること。

レポートは授業開始時までに、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 1B」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. プログラムを作るという課題はどれくらい大変でしたか?

   Q2. 疑似コードを書くのと、Java に直すのと、打ち込んで動かすのとで掛かった手間の比率はどうですか?

   Q3. 課題に対する感想と今後の要望をお書きください。

C　その他

注意! 自動集計する都合上、レポートのメールはすべて東大 ECC のアカウントから出してください。自宅等、別のアカウントから来たものは「保留」します (後日東大 ECC から同じものを送ってもらった時点で受理します)。よろしく。

レポートは× (提出なし)、△ (遅刻、保留、ないし内容に問題)、○ (普通)、◎ (特に買うべき点がある)、の 4 段階で評価します。昨年の状況から言って、「◎」がある程度ないと成績「A」をつけるのは難しそうです (「A」の比率に制限があります)。

「◎」の基準ですが、久野から見て「これは工夫されている/買える/よいアイデアがある」と判断したものに差し上げます。プログラムが高度ならいいとかいうものではなく、その人のレベルから見て工夫があれば買いますから、初心者の人もにも十分チャンスがあります。頑張ってください。

なお、レポートにアンケートの回答が付随していなかったり、回答として内容のないもの (例: 全項目に「よくわかりません」「?」「むずい」等の記入しかないもの) は△になると思ってください。アンケートは授業内容に関する重要なフィードバック材料ですので、簡単でいいですからちゃんと記入してください。遅刻の「△」は差し替えませんが、アンケート等の内容不完全で「△」のものは後日適切なものを再提出して頂ければ「○」に差し替えます。

その他、個人的な質問等があればいつでも、メールで久野 (上記メールアドレスです) あてお知らせください。ただしレポートと混同するような Subject:は避けてくださいね^_^;。課題の分からないところ等、全般的な質問であれば掲示板の方がよいと思います。

次回から資料は自分で打ち出してきてください。本クラスの Web ページの「資料」ページに PS(PostScript) 版と PDF 版へのリンクを起きますから (内容は同一)、どちらでも都合のいい方を打ち出してください。授業時に資料を持って来てないと時間を無駄にしますから、必ず予め打ち出し、できるだけ目を通して来てください。印刷がもったいないからと画面で見るだけで済ませる人がいますが、久野個人としては「紙に打ち出して繰り返し資料を読む」ことが上達の早道だと考えています。いくら紙が節約できてもプログラミングで挫折したら大損でしょう?