付録 Java による非常に簡単な科学技術プログラミング
ー実行環境の設定、実行方法、およびプログラムのダウンロードについてー

小山敏幸
名古屋工業大学 大学院工学研究科(しくみ領域)
物質工学専攻(物性分野)
工学部 環境材料工学科(材料機能系プログラム)
〒466-8555 名古屋市昭和区御器所町
TEL:052-735-5124, E-mail:koyama.toshiyuki@nitech.ac.jp
URL: http://tkoyama.web.nitech.ac.jp/Phase-Field_Modeling.htm

１．はじめに

ここでは Java[1]を用いた非常に簡単なプログラミング（科学技術計算）について説明する。ただし内容については、計算に必要な最小限度の内容にとどめている。

さて、従来、Java 関連の書籍では、文字列を"Hello World"と表示するところから始まり、クラスの概念、インスタンスの生成、継承、・・・等、オブジェクト指向[1]のプログラミングの考え方が展開される場合が多い。Java を用いてソフトウェア（ごく簡単なものも含めて）を作成することを目的とするならば、このような教科書は有用である。しかし、技術者・研究者で、ソフトウェア作成ではなく、単純に数値計算がしたいだけであるならば、上記のテキストの大部分は必要のない知識である。なぜならば、通常、技術者・研究者が計算において必要とする操作の大部分は、以下の三種類のみであるからである。

（１）手続き的な数値の計算
（２）結果の可視化（グラフや画像）
（３）数値データの保存と読み出し

きちんとしたソフトウェア作成自体を目的としなければ、プログラムにおいて、ボタン等のコンポーネントの作成や過度の例外処理などは必要ない。単にソースプログラムが書けて、それをコンパイルし、上記三つの操作に対して、プログラムを実行して結果さえ得られれば良い（計算結果の数値データさえ保存できれば、それを綺麗にグラフ化・可視化するソフトは巷にあふれている）。プログラムに入力値が必要な場合には、極論すれば、ソースコードに毎回直接書き込み、その都度コンパイルして実行しても、ほとんどの場合、日常必要とする数値計算に支障は起きない。荒っぽい言い方であるかもしれないが、プログラムも、実行するたびにその都度強制終了するもの(終了操作の記述自体を含まないプログラム)であってもかまわないのである[2]。

Java に限った話ではないが、基本的に通常の Windows ソフトウェアを作成することを目的としてしまうと、イベントドリブン型の部分、すなわち、ボタンやツールーバー、テキストボックス、クリック、ドラッグ、・・・など、本来の数値計算部分とは全く無関係な部分を作りこまなくてはならなくなるために、Windows プログラミングは非常に複雑になる（コンポーネントウェアの概念にてプログラミングの手間はかなり削減されてはいるが、根本的に数値計算以外の部分が多く煩雑である点にかわりはない）。数値計算を行いたいだけであるならば、ボタンやツールーバー、クリック、ドラッグなどのイベントドリブン型の部分の知識は全く必要ないので、この部分の学習は後回しでかまわない（後々、きちんとした Windows ソフトウェアを作成したくなった時にあらためて勉強すれば良い）。

もう一つ最近のパソコンの高性能化をあらためて考えてみていただきたい。現在のパソコンは、十年前の通常の大型計算機並みの能力がある。十年前、数千万円単位であったマシンが、現在は数万円で個人的で購入できるのである（実験機器でたとえるならば、十年前の最新鋭の電子顕微鏡が、現在、個人で買えるようになったようなものである）。これを研究・教育に使わないのは、明らかにもったいない。もちろん、既存のソフトウェアは豊富であり、文書作成や研究発表などでパソコンは大いに活用されている。しかし、みずからソースコードを書いて数値計算する用途には、あまり使用されていないのではないだろうか。本当は誰でも簡単にプログラミングして数値計算できるのである。

そこで、以下では、上記の三つの操作に限定して、非常に簡単に Java アプリケーションを作成する手法を解説する。なお Java アプレット（ネットワークを通して Web ブラウザに読み込まれ実行される Java のアプリ

ケーションの一種）[1]については言及しない。Java アプレットには、インターネット上で稼動するソフトウェアを簡単に作成できる利点があるが、セキュリティーの関係で、データのディスクへの保存機能がない。これでは数値計算してもデータを保存できないので、上記（３）の操作を実行できず、技術者・研究者における日常の数値計算の目的からはふさわしくないからである。ちなみに、Java アプリケーションを Java アプレットに書き直すことは、それほど難しくはなく、また Java アプレットに関しては多くの入門書が出版されているので、興味のある読者はトライされることを薦める。言語として Java を選んだ理由は、Java は OS に非依存であるので、MS-Windows でも MAC でも Linux でも、同一のソースプログラムにて計算できるからである。特にグラフィックスに関するライブラリーが OS に依存しない点も重要である。さらにインターネットを通じて無料でプログラミング環境が入手できる点も利点であろう。

２． Java 本体の入手先およびソースコードの実行方法

Java 本体のインストールの詳細については、ホームページ（http://java.sun.com/javase/ja/6/download.html）等を参照していただきたい（**jre ではなく jdk をインストールする点に注意**）。また以下、OS は MS-WIndows XP 以降とする。Java のバージョンに関しては、本計算では JDK 5 以降とし（執筆時の最新バージョンは JDK6 Update 23）、Java のインストール先については、上記 URL から Java 本体をダウンロードして、C ドライブの c:¥jdk6_tmp にインストールされているものとする（ディレクトリは新規に作成する）。

さて、ソースコードのコンパイルおよび実行方法について説明しよう。ソースコードの名前をたとえば、"F_curve_FePt.java"とする。F_curve_FePt.java のファイル形式は通常のテキストファイルである（MS-Windows の"メモ帳"などで読み書きできる形式）。ソースコードをコンパイルするために、バッチファイル(MS-DOS、OS/2、Windows でのコマンドプロンプトに行わせたい命令列をテキストファイルに記述したもので、UNIX 系オペレーティングシステムのシェルスクリプトに相当する)を作成しておくと良い。著者は、図 1 のようなバッチファイルを使用している。ファイル名は"vjc.bat"で（ファイル形式はテキストファイル）、内容を図 1 に示す（Java 本体は c:¥jdk6_tmp にインストールされている場合を想定している）。このバッチファイル vjc.bat とソースコード F_curve_FePt.java は、C:¥tmp_program にあるものとする。コマンドプロンプト（通常、MS-Windows のアクセサリー内にある）を起動し、カレントディレクトリを C:¥tmp_program に移動して、C:¥tmp_program>vjc F_curve_FePt.java　と入力しリターンすることによって、F_curve_FePt.java がコンパイルされる。エラーがなければクラスファイル F_curve_FePt.class が生成される。エラーがある場合には、エラーメッセージとそのエラーが存在するソースコード上の行番号がコマンドプロンプト画面に示されるので、エラーメッセージに従いソースコードを修正する。

```
set JAVADir=c:¥jdk6_tmp
set path_1=%path%
path %JAVADir%¥bin;%JAVADir%¥include;%JAVADir%¥lib;
javac -deprecation %1 %2 %3 %4 %5 %6 %7 %8 %9
path ;
path %path_1%
```

図 1 vjc.bat の内容

プログラムの実行（得られた F_curve_FePt.class の実行）には、図 2 のバッチファイルを使用すると良い（ファイル名を"vj.bat"とする）。具体的には、C:¥tmp_program>vj F_curve_FePt　と入力し（拡張子の class は必要ない）リターンすることによって、プログラム（F_curve_FePt.class）を実行することがで

```
set JAVADir=c:¥jdk6_tmp
set path_1=%path%
path %JAVADir%¥bin;%JAVADir%¥include;
java %1 %2 %3 %4 %5 %6 %7 %8 %9
path ;
path %path_1%
```

図２　vj.bat の内容

きる（なおプログラムを停止するには、画面右上の×をクリックして、強制終了する)。なお上記のバッチファイルでは、毎回、現行の path を書き換えて、処理を行った後、再び path を元に戻す操作を行っており、あまりスマートではない。OS の path 設定にあらかじめ Java 関連の path を追加しておけば、この部分の操作は不要となる。

３．プログラムの説明

以下では、実際の研究に役に立つ代表的な例題プログラム F_curve_FePt.java について説明する。F_curve_FePt.java は、以下の一連の操作を行っている。

(1) 規則度 *s* の自由エネルギー関数 *G*(*s*)を計算し、*s* と *G*(*s*)の離散数値データを配列に入力
(2) *s* と *G*(*s*)の配列を描画（自由エネルギー曲線）
(3) *s* と *G*(*s*)の配列をディスクに保存
(4) (3)で保存された *s* と *G*(*s*)の配列を読み出し、別の配列に入力
(5) (4)の配列データを再描画

したがって、データの数値計算、グラフ化、保存、および読出しの一連の操作が、このコードに含まれている。通常の技術系の計算では、以上が出来れば、ほとんどの作業が可能となる。なお自由エネルギーの数値データを計算したいだけならば、上記の(4)と(5)は必要ない。(4)と(5)はデータの読み出し部分の説明のために加えたものである。また上記の関数 G は、本書の FePt の変位型変態の計算に用いた化学的自由エネルギー曲線である。

```
import文
public class F_curve_FePt extends Frame{
   グローバル変数
   public F_curve_FePt(){}                    //コンストラクタ
   public static void main(String[] args){}   //メインプログラム
   double G(double s, double AA1){}           //自由エネルギー関数
   public void paint(Graphics g){}            //自由エネルギーのグラフ描画
   private void datsave(){}                   //データの保存
   private void datin(){}                     //データの読み出し
}
```

図 3 F_curve_FePt.java の構成

さて、プログラム F_curve_FePt.java の全体構成は、図 3 のようになっている。Java のプログラムの基本はクラスであり、このプログラムは、F_curve_FePt という名前のクラスである（このプログラムには、グラフ表記が含まれるので、Frame クラスを継承して定義している)。図 3 の最初の import 文は C 言語[3]における include 文に相当する。続いてクラスの中身について説明する。クラス全体で共通に使用する変数（や配列）が、「グローバル変数」である。コンストラクタ[1]は、このクラスの実体（この場合は画面）を作成する部分である（厳密ではないが、グラフを書くための下地の Window の設定と考えても良い)。メインプログラムが実際の計算処理および描画を行っているプログラム部分である。その後の４行がサブルーチンで、それぞれ、自由エネルギーの値を計算する部分、自由エネルギーのグラフの描画、数値データのハードディスクへの保存、およびハードディスクからの数値データの読み出しに対応している（プログラムの実行は、コマンドプロンプトから行うので、標準入出力画面はコマンドプロンプト画面となる)。

さて F_curve_FePt.java のプログラムソースは著者のホームページ（後述）よりダウンロードできる。Java では、ソースプログラムにおいて、ダブルスラッシュ"//"以降の１行がコメント文として扱われるので（C++でも同様)、プログラムの内容説明は、ソースプログラム内にコメント文として、直接書き込んである。

図 4 はプログラム：F_curve_FePt.java を実行した時の自由エネルギー曲線である。縦軸が自由エネルギーで、横軸が組成、始めに図 A4 (a)の曲線が表示され、５秒ほどしてから、図 4 (b)のように曲線が上書きされる。ここでのグラフ表示は、計算時の結果確認を意図しており、綺麗なグラフ作成を目的としていない。したがって、単に曲線の描画のみにとどめ、縦軸や横軸の説明などの表示は全て省略している。曲線の数値データはハードディスクに保存されているので､たとえば論文用などの図を描きたい時には、この数値データを各種のグラフ作成ソフト等に読み込んで綺麗に清書すればよい（ちなみに本書図 7.2 の自由エネルギー曲線は、この数値データを読み込んでグラフ作成ソフトにて描いたものである)。最近では､非常に優れたグラフ作成ソフトや可視化ソフトが簡単に入手できるようになり、また多くのグラフ作成ソフトには、関数を定義して曲線を描く機能は内蔵されている。関数が初等的で簡単な式ならば問題はないが、数値積分や収束計算などが含まれる場合には役にたたない。ここで説明した手法ならば、もともとソースコードを自分自身で書いているのであるから、いかなる計算にも対処でき、ソフトの制約を受けない。さらに作成したプログラムは、自分が一生使える資産となって積み上がっていく点も強調しておきたい。

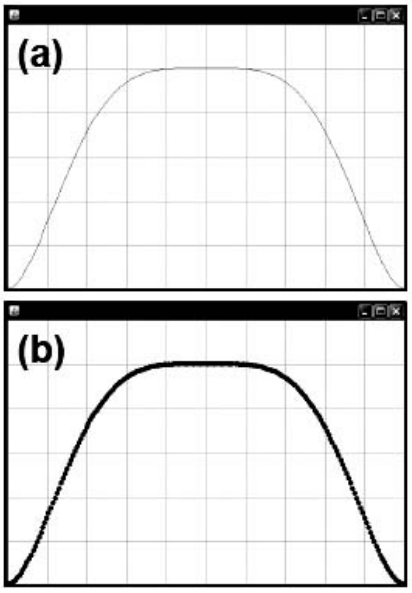

図 4 自由エネルギー曲線の計算結果
(a) 1 回目の描画, (b) 2 回目の描画

４．各種ソースコードについて

各種のソースコードは¥Java 内において、以下のように分類されている（コンパイルおよび実行のためのバッチファイルは全てのディレクトリに含まれている)。

（１）自由エネルギーを例題とした、数値計算、クラフの描画、およびデータの入出力（前節参照）
¥tmp_program¥F_curve_FePt.java　自由エネルギーの数値計算、曲線の描画、データ入出力

（２）拡散相分離シミュレーションのプログラム

¥PD1D¥PD1D_001.java	相分離における一次元濃度プロファイルの時間発展（データ保存）
¥PD1D ¥plot1D_c.java	計算結果の表示
¥PD1D¥select_data1D_c.java	計算結果のファイルからデータの抽出
¥PD2D¥PD2D_001.java	相分離における二次元濃度場の時間発展（データ保存）
¥PD2D ¥plot_c.java	計算結果の表示
¥PD2D¥bplot_c.java	計算結果のイメージデータを保存
¥FeCr¥FeCr_PD_2D_001.java	相分離における二次元濃度場の時間発展（データ保存）
¥FeCr¥plot_c.java	計算結果の表示
¥FeCr¥bplot_c.java	計算結果のイメージデータを保存

（３）変位型変態シミュレーションのプログラム

¥FePt¥FePt2D_001.java	変位型変態における組織形成の時間発展（データ保存）
¥FePt¥FePt2D_A001.java	変位型変態(外部応力下)における組織形成の時間発展（データ保存）
¥FePt ¥plot_s12.cpp	計算結果の表示

¥FePt¥bplot_s12.cpp　　計算結果のイメージデータを保存

ソースコード内に、上記のプログラムの内容説明をコメント文として直接書き込んでいるので、適宜参照していただきたい。

５．おわりに

先に説明した F_curve_FePt.java は一例題に過ぎないが、これを修正・改良することによって、様々な技術系の数値計算に対応したプログラムを自由に作成することができる。実際、上記の拡散相分離および変位型変態の組織形成シミュレーションは全て、F_curve_FePt.java を修正してプログラミングしている。また F_curve_FePt.java は、各種の実験データを解析するプログラムの作成等にも役立つと思う。

ここで説明したバッチファイルやソースコード、また本書にて説明した各種シミュレーションのソースコード等は、全て著者のホームページ（http://tkoyama.web.nitech.ac.jp/Phase-Field_Modeling.htm）よりダウンロードできるので、ご自由にお使いいただきたい。各種シミュレーションのソースコードのコンパイルおよび実行方法も、上記にて説明した手法と同じである。また本書の内容修正情報やより進んだ解析法（シミュレーションのソースコードを含む）なども追加・更新していく予定であるので随時参照していただきたい。なお本プログラム等に関して不具合・トラブルが発生しても、著者および(株)内田老鶴圃は責任を負いませんのでご了承ください。

参考文献

[1] 宮坂雅輝：エッセンシャル Java (2nd edition), ソフトバンククリエイティブ, (2003).
[2] 小山敏幸: まてりあ, **48**(2009), 466.
[3] 柴田望洋: 明解 C 言語 入門編, ソフトバンククリエイティブ, (2004).