桂田研 Java 入門

桂田 祐史

2008 年 2 月 28 日

この文書はもともとは「2003 年度版 桂田研 Java 入門」として発行したものの、焼き直しである。

http://www.math.meiji.ac.jp/~mk/labo/java/intro-java/ で読める。リンクを張ってあるところにアクセスしたり、プログラムを入手するにはその方が便利かも。

目 次

1 イントロ

1.1 なぜ Java か

- 表面上は C や C++ に似ているので、初めて学ぶ際に抵抗感が少ない (実際、数値計算をする部分のコードはかなりの部分「コピーして使える」)。
- グラフィックス、GUI、ネットワーク・プログラミング等の機能を含んだクラス・ライブラリィが規格に含まれているので、これらの機能を利用したプログラムが (互換性を持った形で) 書ける。
- 多くのプラットホームで同じプログラムが動作する ("write once, run everywhere")。例えば Applet ならば WWW ブラウザー上で実行できる。
- (今のところ) 処理系がフリーで入手可能である。
- 比較的新しい (発表は 1995 年) ので、設計がモダンで、プログラミング言語についての重要な概念 (例えばオブジェクト指向プログラミング、ガーベージ・コレクション) をサポートしているので、「教養」としても役立つ[1]。

1.2 学生に勧めたいこと

- 本に載っているプログラムを試す。
- この文書で説明してあるプログラムを試す。

1.3 自分で試す

プログラム等は次のページから入手できますが、覚えるためには自分の手で打ち込んだ方が良いかもしれません。

『桂田研学生のための Java のページ』

`http://www.math.meiji.ac.jp/~mk/labo/java/`

この文書に載っているサンプル・プログラムの多くを、一つの書庫ファイルにまとめて公開してあります。

1. Windows 向けには (文字コードをシフトジスにしてある) `sample-sjis.lzh`[2]
   桂田研のノートパソコンでは、ダブルクリックで Lhaca が動いて、解凍してくれるはずです。

[1]「使われている」という意味での実用言語って結構古いものが多いですからね。数学科のようなところで、個人的にコンピューターに対する興味を持って自学自習しないと古いことしか知らない人になってしまいかねない。

[2]`http://www.math.meiji.ac.jp/~mk/labo/java/sample-sjis.lzh`

2. Linux (Knoppix を含む) 向けには (文字コードを日本語 EUC にしてある) `sample.tar.gz`[3] Linux では `tar xzf sample.tar.gz` とすれば、`sample` というディレクトリィが復元されます。

文字コードは自分で簡単に変換できます。やり方については相談して下さい。

1.4 Java 言語のプログラミング環境の準備

Java の処理系というのは、Sun Microsystem が作成している **JDK** (**Java Development Kit**) が本家であるわけだ。特に理由がなければこれを使うことを勧める。
「2007 年版 JDK のインストール法」[4] というのを書いたのでよろしく。

2 標準入出力を使ったアプリケーション

2.1 何もしないプログラム

次に掲げる `DoNothing.java`[5] は、何もしないプログラムである。

DoNothing.java

```
1  /*
2   * DoNothing.java --- 最短のアプリケーション（何もしない）
3   */
4
5  public class DoNothing {
6    public static void main(String args[]) {
7    }
8  }
```

- Java 言語のプログラムはクラスからなる[6]。
- `MyClass.java` というファイルには、MyClass という public なクラスの定義を書くことになる。そのクラスの定義の仕方は

  ```
  public class MyClass {
    中身
  }
  ```

  (つまり、Java のプログラムは中で定義しているクラスと同名でなければならない。)
- C と同様、アプリケーションは `main()` という名前の関数の実行から始まる。ただし C とは異なり `public static void main(String args[])` となる。

  (1) 関数の返す型が int から void になった。

[3] `http://www.math.meiji.ac.jp/~mk/labo/java/sample.tar.gz`
[4] `http://www.math.meiji.ac.jp/~mk/labo/2007/how-to-install-Java/`
[5] `http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothing.java`
[6] C 言語のプログラムは、強いて言えば「関数からなる」。

(2) public, static というオマジナイがついている— これについてはおいおい説明していく (もちろん自分で本を読んでも良い)。

(3) 引数が一つ (args) だけになった。これは Java の配列が、要素数の情報を length フィールドに持っているからであろう。例えば、コマンド・ラインの引数を表示するプログラムは次のように書ける (PrintArgs.java[7])。ここでは args そのものが引数の個数を args.length として持っていることに注意しよう。

PrintArgs.java

```
/*
 * PrintArgs.java
 */

public class PrintArgs {
  public static void main(String[] args) {
    int i;
    for (i = 0; i < args.length; i++)
      System.out.println(i + "番目の引数は" + args[i]);
  }
}
```

2.2 Hello world

C 言語のバイブルである Kernighan & Ritchie 著「プログラミング言語 C」の伝統にのっとり、C の親戚の言語の最初のサンプル・プログラムは、"Hello world" という文字列を表示するものと決まっている。その Java 言語版 HelloWorld.java[8] は次のようになる。

HelloWorld.java

```
1  /*
2   * HelloWorld.java --- Hello World
3   */
4
5  public class HelloWorld {
6    public static void main(String args[]) {
7      System.out.println("Hello, world.");
8    }
9  }
```

HelloWorld.class のコンパイル&実行

```
1  yurichan% javac HelloWorld.java
2  yurichan% java HelloWorld
3  Hello, world.
4  yurichan%
```

[7] http://www.math.meiji.ac.jp/~mk/labo/java/prog/PrintArgs.java
[8] http://www.math.meiji.ac.jp/~mk/labo/java/prog/HelloWorld.java

2.3 標準入力からのデータの読込み

「標準入力から二つの数を読込んで、和、差、積、商を計算して、標準出力に出力せよ。」という例題は、Fortran, Pascal, C, C++ などでは、よく取り上げられる。

C の場合、入門書には、次のような scanf() を用いたプログラムが載っている。

readwrite1.c

```
 1  /*
 2   * readwrite1.c
 3   */
 4
 5  #include <stdio.h>
 6
 7  int main()
 8  {
 9    double a, b, wa, sa, seki, syou;
10    printf("二つの数を入力してください: ");
11    scanf("%lf%lf", &a, &b);
12    wa = a + b;
13    sa = a - b;
14    seki = a * b;
15    syou = a / b;
16    printf("和=%g, 差=%g, 積=%g, 商=%g\n", wa, sa, seki, syou);
17    return 0;
18  }
```

ここで詳しい理由の説明はしないが[9]、scanf() をこのように使うのは良くないとされている[10](ある意味で scanf() は悪名高い関数である)。入力を文字列として読んでから、文字列を解析することが推奨されている。例えば以下のプログラムのようになる。

[9] C 言語の説明はこの文書の目的ではないから。

[10] それにもかかわらず、入門書に scanf() を使ったプログラムが載っているのは、簡単なプログラムならば、おかしなことが起こっても、実行を中断して、最初からやり直せば良い、と考えているからであろう。システム・プログラムなど、やり直しの許されないプログラムで scanf() を使うのはマズイ。

readwrite2.c

```
 1  /*
 2   * readwrite2.c
 3   */
 4
 5  #include <stdio.h>
 6
 7  int main()
 8  {
 9    char buf[BUFSIZ];
10    double a, b, wa, sa, seki, syou;
11    printf("二つの数を入力してください: ");
12    /* 一行読込む */
13    fgets(buf, sizeof(buf), stdin);
14    /* buf[] の内容を sscanf() で解析する */
15    if (sscanf(buf, "%lf%lf", &a, &b) != 2) {
16      fprintf(stderr, "Input Error\n");
17      exit(1);
18    }
19    wa = a + b;
20    sa = a - b;
21    seki = a * b;
22    syou = a / b;
23    printf("和=%g, 差=%g, 積=%g, 商=%g\n", wa, sa, seki, syou);
24    return 0;
25  }
```

さて、Java ではどうするか？実は初心者には結構な難題である。Java には scanf() も sscanf() も存在しない。これは scanf 系の関数が複雑過ぎるというのが一つの理由であろう (色々なエラーが起こり得るものを一つの関数に押し込めると、エラーの解析が大変である)。もう一つは、Java の主たる応用では、ユーザー・インターフェイスは標準入出力よりは GUI であり、もともと複雑な入力を解析する必要性が低いためであろう[11]。

一行に一つの数値しかないのであれば、次の ReadWrite1.java[12] のように、readLine() メソッドを使って文字列として読込み、Double.valueof().doubleValue(文字列) を使って double の値に変換する[13]、というコードが考えられる (これはまあまあ簡単である)。(try 〜 catch 構文については後述する (予定は…)。)

[11]例えば、テキスト・フィールドには一つの数値しか入力されないので、一つの数値を扱うメソッドがあれば十分であろう。

[12]http://www.math.meiji.ac.jp/~mk/labo/java/prog/ReadWrite1.java

[13]ちなみに文字列を int に変換するには Integer.parseInt(文字列) とする。

ReadWrite1.java

```
 1  /*
 2   * ReadWrite1.java
 3   */
 4
 5  import java.util.*;  // StringTokenizer
 6  import java.io.*;    // BufferedReader
 7
 8  public class ReadWrite1 {
 9    public static void main(String args[]) {
10      double a, b, wa, sa, seki, syou;
11
12      BufferedReader d = new BufferedReader(new InputStreamReader(System.in));
13      try {
14        System.out.print("数を入力してください: ");
15        a = Double.valueOf(d.readLine()).doubleValue();
16        System.out.print("数を入力してください: ");
17        b = Double.valueOf(d.readLine()).doubleValue();
18        wa = a + b;
19        sa = a - b;
20        seki = a * b;
21        syou = a / b;
22        System.out.println("和=" + wa + ", 差=" + sa
23                           + ", 積=" + seki + ", 商=" + syou);
24      }
25      catch(IOException e) {
26        System.out.println("IO Error");
27        System.exit(1);
28      }
29    }
30  }
```

一行の入力に二つ以上の数値を含める場合には、一つの数を表わす文字列を切り出してから、Double.valueof().doubleValue() を使って double の値に変換する、という手順になる。文字列の切り出しには StringTokenizer を用いる。まとめると、次の ReadWrite2.java[14] のようになる。

[14] http://www.math.meiji.ac.jp/~mk/labo/java/prog/ReadWrite2.java

ReadWrite2.java

```
 1  /*
 2   * ReadWrite2.java
 3   */
 4
 5  import java.util.*;  // StringTokenizer
 6  import java.io.*;    // BufferedReader
 7
 8  public class ReadWrite2 {
 9    public static void main(String args[]) {
10      double a, b, wa, sa, seki, syou;
11      System.out.print("二つの数を入力してください: ");
12
13      BufferedReader d = new BufferedReader(new InputStreamReader(System.in));
14      try {
15        String str = d.readLine();
16        StringTokenizer aSt = new StringTokenizer(str," ");
17        if (aSt.countTokens() != 2) {
18          System.out.print("Input Error\n");
19          System.exit(1);
20        }
21        a = Double.valueOf(aSt.nextToken()).doubleValue();
22        b = Double.valueOf(aSt.nextToken()).doubleValue();
23        wa = a + b;
24        sa = a - b;
25        seki = a * b;
26        syou = a / b;
27        System.out.println("和=" + wa + ", 差=" + sa
28                           + ", 積=" + seki + ", 商=" + syou);
29      }
30      catch(IOException e) {
31        System.out.println("IO Error");
32        System.exit(1);
33      }
34    }
35  }
```

ReadWrite1, ReadWrite2 の実行

```
1  yurichan% java ReadWrite1
2  数を入力してください: 12.34
3  数を入力してください: 56.78
4  和=69.12, 差=-44.44, 積=700.6652, 商=0.2173300457907714
5  yurichan% java ReadWrite2
6  二つの数を入力してください: 12.34 56.78
7  和=69.12, 差=-44.44, 積=700.6652, 商=0.2173300457907714
8  yurichan%
```

この結果の表示を「何か汚いように感じる」人のために次の小節がある。

2.4 書式の指定

定評のある Java 解説書 “Core Java” の著者である、Cay Horstmann のホームページ http://www.horstmann.com/ には、“The March of Progress” として、かつて次のようなことが載っ

ていた。

```
C
      printf("%10.2f", x);
C++
      cout << setw(10) << setprecision(2) << showpoint << x;
Java

      java.text.NumberFormat formatter
         = java.text.NumberFormat.getNumberInstance();
      formatter.setMinimumFractionDigits(2);
      formatter.setMaximumFractionDigits(2);
      String s = formatter.format(x);
      for (int i = s.length(); i < 10; i++)
         System.out.print(' ');
      System.out.print(s);
```

痛烈な皮肉だが (分かるかな？)、この著者は解決策として、Format というクラスを提示している[15]。それを用いると、次の TestFormat.java[16] のようなコーディングができる。

TestFormat.java

```
 1  /*
 2   * TestFormat.java -- C 言語の printf() 風の書式指定
 3   *
 4   * Format.java の入手
 5   *   Cay Horstmann (http://www.horstmann.com/) 作
 6   *   http://www.horstmann.com/corejava/Format.java
 7   *
 8   * コンパイルの準備
 9   *   mkdir corejave
10   *   cp どこか/Format.java corejava
11   *   cd corejava
12   *   javac Format.java
13   *   cd ..
14   */
15
16  import corejava.*;
17
18  public class TestFormat {
19    public static void main(String args[]) {
20      // Format.print() メソッド --- ただし一つの書式指定しか使えない
21      Format.print(System.out, "%12.8f\n", Math.PI);
22      // 吉田祐一郎はこういう使い方を勧めていた
23      Format fmt = new Format("%12.8f");
24      System.out.println("PI=" + fmt.form(Math.PI) +
25         ", e=" + fmt.form(Math.exp(1)));
26    }
27  }
```

[15] Format.java は http://www.horstmann.com/corejava/Format.java から入手可能である。
[16] http://www.math.meiji.ac.jp/~mk/labo/java/prog/TestFormat.java

TestFormat.class の実行結果

```
1  yurichan% java TestFormat
2    3.14159265
3  PI=  3.14159265, e=  2.71828183
4  yurichan%
```

確かにこれなら許せるというか、心安らぐプログラミングが出来そうである。
実は、現在では少し変わっていて、次のようになっている。

The March of Progress

```
The March of Progress

1980: C
    printf("%10.2f", x);
1988: C++
    cout << setw(10) << setprecision(2) << showpoint << x;
1996: Java

    java.text.NumberFormat formatter =
    java.text.NumberFormat.getNumberInstance();
    formatter.setMinimumFractionDigits(2);
    formatter.setMaximumFractionDigits(2); String s =
    formatter.format(x); for (int i = s.length(); i < 10; i++)
    System.out.print(' '); System.out.print(s);

2004: Java

    System.out.printf("%10.2f", x);
```

結局、Java にも printf() が入ったと言うことです (ちゃんちゃん)。

3 AWT を使ったアプリケーション

GUI を使ったプログラミングを始めよう。
候補として、AWT と Swing という二つのものが頭に浮かぶ。このうち、使いやすさという面では、後から登場した Swing を使うのが良いだろうが、WWW ページ上で走らせることを考えると、今でも AWT を使ったプログラミングが必要であろう。そこで、まずは AWT を使ったプログラムを説明する。

3.1 何もしないプログラム

以前も述べたが、GUI を使ったプログラムの開発は結構手間がかかって大変である。最近のプログラミング言語では、オブジェクト指向に基づいたクラス・ライブラリィを利用するこ

とで、この困難さと折り合いをつけている。Java も例外ではない。

一応、真似をするに足るプログラム DoNothingAWT4.java もそれなりに複雑なので、段階を追って説明する。

ウィンドウを出すためのクラス Frame を、とにかく使ってウィンドウを出してみたのが次のプログラム DoNothingAWT1.java[17] である。

DoNothingAWT1.java

```
 1  /*
 2   * DoNothingAWT1.java --- AWT でとにかくウィンドウを作る
 3   *  (ひどいプログラムなので良い子は真似をしてはいけない)
 4   */
 5
 6  import java.awt.*;
 7
 8  public class DoNothingAWT1 {
 9    public static void main(String args[]) {
10      Frame f = new Frame();
11      f.setSize(300, 300); // これがないとやたら小さいウィンドウになる
12      f.show(); // これがないとウィンドウを出さずに即終了
13    }
14  }
```

これは、まともに終了することが出来ない。Close を選択してもダメ。きちんと終了させるには、そのためのコードが必要。そういう修正をしたのが、次の DoNothing2.java[18] である。

DoNothing2.java

```
 1  /*
 2   * DoNothingAWT2.java --- AWT でとにかくウィンドウを作る
 3   *   (終了できるようになったが、全然発展性のない、ダメなプログラム)
 4   */
 5
 6  import java.awt.*;       // Frame
 7  import java.awt.event.*; // WindowAdapter 等
 8
 9  public class DoNothingAWT2 {
10    public static void main(String args[]) {
11      Frame f = new Frame();
12      // ちゃんと終了できるように
13      f.addWindowListener(new WindowAdapter() {
14        // 次の名前をミス・スペルすると、分かりにくいバグになる
15        public void windowClosing(WindowEvent e) {
16          System.exit(0);
17        }
18      });
19      f.setSize(300, 300); // これがないとやたら小さいウィンドウになる
20      f.show();            // これがないとウィンドウを出さずに即終了
21      // このウィンドウの中で何かをしようと思っても…できない！
22    }
23  }
```

13〜18 行目の文 (一つの文です!) に注目。しばらくは、これはとにかくこういうものだと覚えてしまうこと (どう書くかは簡単に覚えられないだろうが、とにかくコピーすれば良い)。

[17] http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothingAWT1/DoNothingAWT1.java
[18] http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothingAWT2/DoNothingAWT2.java

とにかくウィンドウを出して、終了できるようになったが、実は発展性 0 のプログラムである。Frame というのは、あくまでも雛形であって、そのまま使うものではない。「Frame を継承したクラスを作って、それで自分がやりたいことを出来るようにして行く」のが普通のやり方である。

次のプログラム DoNothing3.java[19] はそれを素直に実現したものである。Frame のサブクラス MyWindow を作って、main() の中で、MyWindow のインスタンス (オブジェクト) を作っている。

DoNothing3.java

```
 1  /*
 2   * DoNothingAWT3.java --- AWT でウィンドウを作る
 3   *    (最低限の発展性はある)
 4   */
 5
 6  import java.awt.*;       // Frame
 7  import java.awt.event.*; // WindowAdapter 等
 8
 9  // Frame を継承したクラス MyWindow を作る。
10  class MyWindow extends Frame {
11    // MyWindow クラスのコンストラクター
12    MyWindow() {
13      setSize(300, 300);
14      addWindowListener(new WindowAdapter() {
15        public void windowClosing(WindowEvent e) {
16          System.exit(0);
17        }
18      });
19      // 例えば、ここから下に自分のやりたいことが書ける !!
20    }
21  }
22
23  public class DoNothingAWT3 {
24    public static void main(String args[]) {
25      Frame f = new MyWindow();
26      f.show();
27    }
28  }
```

ところが、最近の Java のテキストでは、このような書き方はせずに、次の DoNothing4.java[20] のように書いてあるプログラムが多い (クラスの中のメソッド main で、インスタンスを生成している — 例えば C++ などでオブジェクト指向プログラミングに慣れている人でも、違和感を感じる人が多いようだ)。

[19] http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothingAWT3/DoNothingAWT3.java
[20] http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothingAWT4/DoNothingAWT4.java

DoNothing4.java

```
 1  /*
 2   * DoNothingAWT4.java --- AWT でウィンドウを作る
 3   *      一つのクラスで済ませてしまう --- 気持ち悪がる人もいる書き方だが流行
 4   */
 5
 6  import java.awt.*;       // Frame
 7  import java.awt.event.*; // WindowAdapter 等
 8
 9  public class DoNothingAWT4 extends Frame {
10    DoNothingAWT4() {
11      setSize(400, 400);
12      addWindowListener(new WindowAdapter() {
13        public void windowClosing(WindowEvent e) {
14          System.exit(0);
15        }
16      });
17      // この下に自分のやりたいことが書ける !!
18    }
19    public static void main(String args[]) {
20      // 自分で自分を産むみたいで気持ちが悪いという人はオジサンなのか？
21      Frame f = new DoNothingAWT4();
22      f.show();
23      // ちなみに、この後
24      // Frame g = new DoNothingAWT4(); g.show();
25      // とすると、もう一つウィンドウを出せます。
26    }
27  }
```

おまけ

Swing では、どうなるか気になる人もいると思うので、Swing 版 DoNothing `DoNothingSwing.java`[21] を載せておく。

[21] `http://www.math.meiji.ac.jp/~mk/labo/java/prog/DoNothingSwing.java`

DoNothingSwing.java

```
 1  /*
 2   * DoNothingSwing.java --- Swing でウィンドウを作る
 3   */
 4
 5  import java.awt.*;       // Frame
 6  import java.awt.event.*; // WindowAdapter 等
 7  import javax.swing.*;     // JFrame
 8
 9  public class DoNothingSwing extends JFrame {
10    DoNothingSwing() {
11      setSize(400, 400);
12      addWindowListener(new WindowAdapter() {
13        public void windowClosing(WindowEvent e) {
14          System.exit(0);
15        }
16      });
17    }
18    public static void main(String args[]) {
19      JFrame f = new DoNothingSwing();
20      f.show();
21    }
22  }
```

ほとんど違いがない。import を除けば、Frame が JFrame になっただけである。覚えることはそんなに増えないということで、安心しましょう。

3.2 HelloWorld

それでは何かをしてみる。ここではウィンドウに “Hello, world” という文字列を書くプログラム HelloWorldAWT.java[22] を掲げる。

[22] http://www.math.meiji.ac.jp/~mk/labo/java/prog/HelloWorldAWT/HelloWorldAWT.java

HelloWorldAT.java

```
 1  /*
 2   * HelloWorldAWT.java --- AWT でウィンドウを作り、HelloWorld と書く。
 3   */
 4
 5  import java.awt.*;       // Frame
 6  import java.awt.event.*; // WindowAdapter 等
 7
 8  public class HelloWorldAWT extends Frame {
 9    HelloWorldAWT() {
10      setSize(400, 400);
11      addWindowListener(new WindowAdapter() {
12        public void windowClosing(WindowEvent e) {
13          System.exit(0);
14        }
15      });
16    }
17    public void paint(Graphics g) {
18      Font f = new Font((g.getFont()).getName(), Font.BOLD, 24);
19      g.setFont(f);
20      g.drawString("Hello, world", 150, 150);
21    }
22    public static void main(String args[]) {
23      Frame f = new HelloWorldAWT();
24      f.show();
25    }
26  }
```

paint() メソッドを自分で書いていることに注意。このプログラムのどこからも paint() は呼ばれないように見えるが大丈夫。

3.3 四則演算プログラム — Java ではどうなるか？

Shisoku.java

Shisoku.java[23]

```
 1  /*
 2   * Shisoku.java --- 四則演算プログラム (GUI 版)
 3   */
 4
 5  import java.awt.*;       // Frame
 6  import java.awt.event.*; // WindowAdapter 等
 7
 8  public class Shisoku extends Frame implements ActionListener {
 9    TextField tf_a, tf_b, wa, sa, seki, syou;
10    Button bt, bt2;
11
12    Shisoku() {
13      setSize(400, 400);
14      addWindowListener(new WindowAdapter() {
15        public void windowClosing(WindowEvent e) {
16          System.exit(0);
17        }
```

[23] http://www.math.meiji.ac.jp/~mk/labo/java/prog/Shisoku/Shisoku.java

```
18      });
19
20      setLayout(new GridLayout(7,2));
21      add(new Label("a"));
22      tf_a = new TextField("0", 10);
23      add(tf_a);
24      add(new Label("b"));
25      tf_b = new TextField("0", 10);
26      add(tf_b);
27      add(new Label("a+b="));
28      wa = new TextField("0", 10); add(wa);
29      add(new Label("a-b="));
30      sa = new TextField("0", 10); add(sa);
31      add(new Label("a*b="));
32      seki = new TextField("0", 10); add(seki);
33      add(new Label("a/b="));
34      syou = new TextField("0", 10); add(syou);
35      bt = new Button("start");
36      add(bt);
37      bt.addActionListener(this);
38      bt2 = new Button("exit");
39      add(bt2);
40      bt2.addActionListener(this);
41    }
42
43    public void actionPerformed(ActionEvent e) {
44      if (e.getSource() == bt) {
45        double a = Double.valueOf(tf_a.getText().trim()).doubleValue();
46        double b = Double.valueOf(tf_b.getText().trim()).doubleValue();
47        wa.setText("" + (a + b));
48        sa.setText("" + (a - b));
49        seki.setText("" + (a * b));
50        syou.setText("" + (a / b));
51      }
52      else if (e.getSource() == bt2) {
53        System.exit(0);
54      }
55    }
56
57    public static void main(String args[]) {
58      Frame f = new Shisoku();
59      f.show();
60    }
61  }
```

4 アプレットでシミュレーション・プログラムを作る

4.1 常識的なこと

- アプレットは WWW ブラウザーからも実行できるシンプルなプログラムであるが、数値シミュレーション用プログラムの開発にもかなり使える (最近の Swing 中心の参考書の記述とはずれてしまうところが多いが、複雑なことはしないので、とりあえずサンプル・プログラムの真似をしてみて欲しい)。

- GUI は基本的には (前節で紹介した) AWT で出来ていると考えて良い。
- アプレットには main() メソッドはない。init() メソッドが呼ばれて初期化が行なわれ、再描画が必要な任意のタイミングに paint() が呼ばれる。
- MyApp.java というプログラムの概形は

MyApp.java
```
// 次のような注釈を入れておくと便利
// <APPLET code="MyApp.class" width=500 height=500></APPLET>

import java.applet.*;
import java.awt.*;

public class MyApp名前 extends Applet {
  // 必要な変数、メソッド（関数）の定義
  ...
  public void init() {
  }
  public void paint(Graphics g) {
  }
}
```

これをコンパイル&実行するには

```
oyabun% javac MyApp.java
oyabun% appletviewer MyApp.java &
```

あるいは (礼儀正しく) MyApp.html を書いて、WWW ブラウザーや appletviewer で実行する。

```
oyabun% appletviewer MyApp.html &
```

ただし MyApp.html には次のようなタグがあるとする。

```
  <APPLET code="MyApp.class" width=500 height=500></APPLET>
```

要するに、appletviewer は、“<APPLET code=’’MyApp.class’’ width=500 height=500></APPLET>” だけしか見ていないということでしょう。

4.2 紹介するプログラムについてコメント

- 以下紹介するプログラムで計算&描画の主たるコードは paint() の中に置く。こうすると、以下のような欠点がある。

– 例えばマウスを使ってウィンドウを移動したり、サイズを変えたりするごとに計算を最初から始めることになる。
– 長い計算をする場合、途中で止めるのが難しい。

…そういうわけで、大規模な計算をするプログラムを作るには不適当であるが、軽めのプログラムならば結構実用的で、簡単であろう。

- 実際に描画するウィンドウの座標系とシミュレートする問題の座標系は独立に設定し、必要に応じて座標変換する。space(), wx(), wy() を見てみよう。

4.3 1 変数関数のグラフを描く

定番の 1 変数関数のグラフを描くサンプル・プログラム DrawGraph.java[24] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/DrawGraph.html) は次のようにすればよい。

```
 1  // DrawGraph.java
 2
 3  // <APPLET code="DrawGraph.class" width=500 height=500> </APPLET>
 4
 5  import java.applet.*;
 6  import java.awt.*;
 7  import java.awt.event.*;
 8
 9  public class DrawGraph extends Applet implements ActionListener {
10
11    private static final String message = "graph of a function with 1 variable";
12    private int N = 100;
13    private double a = 0.0;
14    private double b = 2 * Math.PI;
15    private double wmargin = (b - a) / 10;
16    // 座標系の変換のためのパラメーター
17    private double ratiox, ratioy, X0, Y0;
18    // ユーザーとのインターフェイス (パラメーターの入力)
19    private Label labelN;
20    private TextField inputN;
21    private Button startB;
22
23    // 準備 (変数の用意と座標系の初期化など)
24    public void init() {
25      //
26      setLayout(null);
27
28      labelN = new Label("N");
29      add(labelN);
30      labelN.setBounds(100, 60, 200, 30);
31
32      inputN = new TextField("" + N);
33      add(inputN);
34      inputN.setBounds(300, 60, 100, 30);
35
```

[24] http://www.math.meiji.ac.jp/~mk/labo/java/prog/DrawGraph.java

```
36      startB = new Button("Restart");
37      add(startB);
38      startB.setBounds(400, 60, 100, 30);
39      startB.addActionListener(this);
40    }
41    // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
42    public void actionPerformed(ActionEvent e) {
43      if (e.getSource() == startB) {
44        String str = inputN.getText();
45        N = Integer.parseInt(str);
46        repaint();
47      }
48    }
49
50    // グラフを描きたい関数
51    private double f(double x) {
52      return Math.sin(3 * x) + Math.sin(5 * x);
53    }
54    // 座標変換の準備
55    private void space(double x0, double y0, double x1, double y1) {
56      X0 = x0; Y0 = y0;
57      ratiox = 500 / (x1 - x0);
58      ratioy = 500 / (y1 - y0);
59    }
60    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
61    private int wx(double x) {
62      return (int)(ratiox * (x - X0));
63    }
64    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
65    private int wy(double y) {
66      return 500 - (int)(ratioy * (y - Y0));
67    }
68    // x[], u[] の内容を折れ線グラフとして描く
69    private void drawGaph(Graphics g, double x[], double u[]) {
70      for (int i= 0; i < N; i++)
71        g.drawLine(wx(x[i]), wy(u[i]), wx(x[i + 1]), wy(u[i + 1]));
72    }
73
74    public void paint(Graphics g) {
75
76      double h = (b - a) / N;
77      double [] u, x;
78
79      // ベクトルを確保する
80      x = new double[N+1];
81      u = new double[N+1];
82      for (int i = 0; i <= N; i++)
83        x[i] = a + i * h;
84
85      // タイトルを表示する
86      g.setColor(Color.black);
87      g.setFont(new Font("Helvetica", Font.BOLD, 24));
88      g.drawString(message, 40, 30);
89
90      // 関数値を計算する
91      for (int i = 0; i <= N; i++)
92        u[i] = f(x[i]);
93
```

```
94      // 関数の値の範囲を調べる
95      double min, max;
96      min = u[0]; max = u[0];
97      for (int i = 1; i <= N; i++) {
98        if (u[i] > max)
99          max = u[i];
100       else if (u[i] < min)
101         min = u[i];
102     }
103     // 関数の値の範囲を元にして座標変換を決める
104     double hmargin = (max - min) / 10;
105     if (hmargin == 0) hmargin = 1;
106     space(a - wmargin, min - hmargin, b + wmargin, max + hmargin);
107
108     // 関数のグラフを描く
109     drawGaph(g, x, u);
110   }
111 }
```

4.4 常微分方程式の力学系

定数係数 1 階線形常微分方程式

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

を解いて相図を描くためのプログラム ODE1.java[25] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/ODE1.html) は次のようにすればよい。

```
1  // <APPLET code="ODE1.class" width=500 height=500> </APPLET>
2
3  import java.applet.*;
4  import java.awt.*;
5  import java.awt.event.*;
6
7  class GraphCanvas extends Canvas {
8
9    static final boolean DEBUG = false; // true;
10   private static final String message = "graph of a function with 1 variable";
11   // 問題に取って基本的なパラメーター
12   // 係数行列
13   private double a, b, c, d;
14   // 描画範囲
15   private double x_max =   1.0;
16   private double x_min = - 1.0;
17   private double x_margin = (x_max - x_min) / 10;
18   private double y_max =   1.0;
19   private double y_min = - 1.0;
20   private double y_margin = (y_max - y_min) / 10;
21
22   // 座標系の変換のためのパラメーター
23   private int CanvasX = 400, CanvasY = 400;
24   private double ratiox, ratioy, X0, Y0;
25
```

[25] http://www.math.meiji.ac.jp/~mk/labo/java/prog/ODE1.java

```
26    // コンストラクター
27    public GraphCanvas() {
28      super();
29    }
30    public GraphCanvas(int cx, int cy) {
31      super();
32      CanvasX = cx; CanvasY = cy;
33    }
34
35    public void compute(double A, double B, double C, double D) {
36      a = A; b = B; c = C; d = D;
37      repaint();
38    }
39
40    private boolean IsIn(double x, double y) {
41      return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
42    }
43    // 座標変換の準備
44    private void space(double x0, double y0, double x1, double y1) {
45      X0 = x0; Y0 = y0;
46      ratiox = CanvasX / (x1 - x0);
47      ratioy = CanvasY / (y1 - y0);
48    }
49    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
50    private int wx(double x) {
51      return (int)(ratiox * (x - X0));
52    }
53    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
54    private int wy(double y) {
55      return CanvasY - (int)(ratioy * (y - Y0));
56    }
57    // 力学系の右辺 f=(fx, fy)
58    private double fx(double x, double y) {
59      return a * x + b * y;
60    }
61    private double fy(double x, double y) {
62      return c * x + d * y;
63    }
64    // x[], y[] の内容をグラフにする
65    private void drawGraph(Graphics g, double x0, double y0, double T) {
66      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
67      if (T < 0.0)
68        h = - h;
69      int iter = (int)Math.rint(Math.abs(T / h));
70      double x = x0;
71      double y = y0;
72      double new_x, new_y;
73      for (int i = 0; i <= iter; i++) {
74        double k1x = h * fx(x, y);
75        double k1y = h * fy(x, y);
76        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
77        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
78        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
79        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
80        double k4x = h * fx(x + k3x, y + k3y);
81        double k4y = h * fy(x + k3x, y + k3y);
82        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
83        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
```

```
84        if (IsIn(x, y) && IsIn(new_x, new_y))
85            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
86        x = new_x; y = new_y;
87      }
88    }
89
90    public void paint(Graphics g) {
91      //
92      space(x_min - x_margin, y_min - y_margin,
93            x_max + x_margin, y_max + y_margin);
94      //
95      setBackground(Color.blue);
96      //
97      g.setColor(Color.black);
98      g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
99      g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
100     //
101     g.setColor(Color.yellow);
102     int n = 36;
103     double dt = 2 * Math.PI / n;
104     double Time = 10.0 / Math.sqrt(a * a + b * b + c * c + d * d);
105     for (int i = 0; i < n; i++) {
106       double t = i * dt;
107       drawGraph(g, Math.cos(t), Math.sin(t),   Time);
108       drawGraph(g, Math.cos(t), Math.sin(t), - Time);
109     }
110   }
111 }
112
113 public class ODE1 extends Applet implements ActionListener {
114
115   private int N = 20;
116   private double lambda = 0.5;
117   private double Tmax = 0.5;
118   // ユーザーとのインターフェイス（パラメーターの入力）
119   private Label label_a, label_b, label_c, label_d;
120   private TextField input_a, input_b, input_c, input_d;
121   private double a, b, c, d;
122   private Button startB;
123   //
124   private GraphCanvas gc;
125
126   private void ReadFields() {
127     a = Double.valueOf(input_a.getText()).doubleValue();
128     b = Double.valueOf(input_b.getText()).doubleValue();
129     c = Double.valueOf(input_c.getText()).doubleValue();
130     d = Double.valueOf(input_d.getText()).doubleValue();
131   }
132   // 準備（変数の用意と座標系の初期化など）
133   public void init() {
134     // ナル・レイアウト
135     setLayout(null);
136     // a, b, c, d を入力するためのテキスト・フィールド
137     add(label_a = new Label("a=")); label_a.setBounds(100, 30, 40, 30);
138     add(label_b = new Label("b=")); label_b.setBounds(250, 30, 40, 30);
139     add(label_c = new Label("c=")); label_c.setBounds(100, 70, 40, 30);
140     add(label_d = new Label("d=")); label_d.setBounds(250, 70, 40, 30);
141     add(input_a = new TextField("" + 1)); input_a.setBounds(150, 30, 100, 30);
```

```
142       add(input_b = new TextField("" + 0)); input_b.setBounds(300, 30, 100, 30);
143       add(input_c = new TextField("" + 0)); input_c.setBounds(150, 70, 100, 30);
144       add(input_d = new TextField("" + 1)); input_d.setBounds(300, 70, 100, 30);
145       // 再計算ボタン
146       startB = new Button("Restart");
147       add(startB);
148       startB.setBounds(420, 45, 50, 30);
149       startB.addActionListener(this);
150
151       // キャンバス
152       gc = new GraphCanvas();
153       add(gc);
154       gc.setBounds(50, 100, 400, 400);
155       ReadFields();
156       gc.compute(a, b, c, d);
157     }
158
159     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
160     public void actionPerformed(ActionEvent e) {
161       if (e.getSource() == startB) {
162         ReadFields();
163         gc.compute(a, b, c, d);
164       }
165     }
166   }
```

Java をサポートしたブラウザーで次のページにアクセスすると実行できる。

http://www.math.meiji.ac.jp/~mk/labo/members/java/ODE1.html

4.5 スレッドの利用

(ちゃんと書いてない。工事中と思って下さい。)

計算が重くなってくると、すべてを paint() でやるというのは無理になる。

UNIX+X Window System ならば、fork() して二つのプロセスを作って、片方が計算、もう片方が描画を担当して、プロセス間通信で…とかするところだが、Java にはスレッドがある。

4.5.1 クラスの宣言

```
public class クラスの名前 extends Applet implements Runnable {
...
}
```

4.5.2 スレッド生成のメソッド start() を用意

```
Thread スレッドの名前;
public void start() {
  スレッドの名前 = new Thread(this);
  スレッドの名前.start();
}
```

4.5.3 全体の進行 (計算？) は run() の中に書く

スレッドの主な中身は run() というメソッドの中に書く。定型的な部分があって、それを示すと、

```
for または while {
  // 計算など (1 コマ分程度？)
  // paint() を明示的に呼び出す
  repaint();

  // sleep() による待機
  try {
    Thread.sleep(待ち時間 (ミリ秒数));
  }
  catch(InterruptedException e) {
  }
}
```

4.5.4 1 コマ分の描画手続きを paint(Graphics g) に

簡単な作業だったら、ここにすべての描画手続きを記述しておく。paint() は run() の中から repaint() で呼び出されるだけでなく、ウィンドウの再描画要求発生時などにも呼び出されるので、なるべく軽くしておくのが吉。

ダブル・バッファリングをしていて、バッファーへの描画は run() で済せておくという場合は、次のようなシンプルなものになる。

```
public void paint(Graphics g) {
  g.drawImage(im, x, y, this);
}
```

4.5.5 stop() について

4.5.6 update() のオーバーライド

`update();` は

1. 画面全体をクリアする
2. `paint()` を呼び出す

という仕事をしている。画面全体のクリアをさぼった方が良いこともある。そういう場合は自分で

```
public void update(Graphics g) {
  paint(g);
}
```

のようなメソッドを作ってオーバーライドしてしまう。

5 テキスト・ファイルの入出力

(これはアプレットでなくアプリケーションの話です。)

5.1 由緒正しくは

次の `FileIOTest.java`[26] は「きちんと」書いたプログラムである。少しもののものしい。

[26] http://www.math.meiji.ac.jp/~mk/labo/java/prog/FileIOTest.java

FileIOTest.java

```
 1  /*
 2   * FileIOTest.java
 3   */
 4
 5  import java.io.BufferedReader;
 6  import java.io.BufferedWriter;
 7  import java.io.FileInputStream;
 8  import java.io.FileOutputStream;
 9  import java.io.InputStreamReader;
10  import java.io.OutputStreamWriter;
11
12  public class FileIOTest {
13
14      public static void main(String[] args) {
15  // TODO 自動生成されたメソッド・スタブ
16  try {
17      int a = 2, b = 3;
18      FileOutputStream out_s = new FileOutputStream("output.data");
19      OutputStreamWriter out_w = new OutputStreamWriter(out_s);
20      BufferedWriter out_b = new BufferedWriter(out_w);
21      out_b.write("" + a + "\n");
22      out_b.write("" + b + "\n");
23      out_b.close();
24      out_w.close();
25      out_s.close();
26  } catch (Exception e) {
27      e.printStackTrace();
28  }
29
30  try {
31      FileInputStream in_s = new FileInputStream("output.data");
32      InputStreamReader in_r = new InputStreamReader(in_s);
33      BufferedReader in_b = new BufferedReader(in_r);
34      String str = in_b.readLine();
35      int c, d;
36      c = Integer.parseInt(str);
37      str = in_b.readLine();
38      d = Integer.parseInt(str);
39      System.out.println("" + c + "+" + d + "=" + (d+c));
40      in_b.close();
41      in_r.close();
42      in_s.close();
43  } catch (Exception e) {
44      e.printStackTrace();
45  }
46      }
47  }
```

5.2 簡略化バージョン

次の FileIOTest2.java[27] は多少簡略化したバージョンである。

[27] http://www.math.meiji.ac.jp/~mk/labo/java/prog/FileIOTest2.java

```
 1  /*
 2   * FileIOTest2.java
 3   */
 4
 5  import java.io.BufferedReader;
 6  import java.io.BufferedWriter;
 7  import java.io.FileReader;
 8  import java.io.FileWriter;
 9
10  public class FileIOTest2 {
11
12      public static void main(String[] args) {
13          try {
14              String str;
15              int a, b;
16              // ファイルのオープン
17              BufferedReader in = new BufferedReader(new FileReader("input.data"));
18              BufferedWriter out = new BufferedWriter(new FileWriter("output.data"));
19              // 1 行ずつ読んで整数に変換
20              a = Integer.parseInt(in.readLine());
21              b = Integer.parseInt(in.readLine());
22              // 和を書き込む
23              out.write("" + (a + b) + "\n");
24              // ファイルのクローズ
25              in.close();
26              out.close();
27          } catch (Exception e) {
28              e.printStackTrace();
29          }
30      }
31
32  }
```

5.3 1 行に複数データ

次の FileIOTest3.java[28] では、1 行の中に複数のデータを入れて、それを分解して読み取っている。果たしてこうするものなのか迷うが…(何となく複雑な書式を使うのは、出力先が紙であった時代の名残であって、入力することを考えると「違う」のかな、と)

[28] http://www.math.meiji.ac.jp/~mk/labo/java/prog/FileIOTest2.java

```
 1  /*
 2   * FileIOTest3.java
 3   */
 4
 5  import java.io.BufferedReader;
 6  import java.io.BufferedWriter;
 7  import java.io.FileReader;
 8  import java.io.FileWriter;
 9  import java.util.StringTokenizer;
10
11  public class FileIOTest3 {
12
13      public static void main(String[] args) {
14          try {
15              BufferedReader in = new BufferedReader(new FileReader("input.txt"));
16              BufferedWriter out = new BufferedWriter(new FileWriter("output.txt"));
17              String s;
18              s = in.readLine();
19              StringTokenizer st = new StringTokenizer(s, " ");
20              if (st.countTokens() != 2) {
21                  System.out.println("Input Error");
22                  System.exit(1);
23              }
24              double a = Double.valueOf(st.nextToken()).doubleValue();
25              double b = Double.valueOf(st.nextToken()).doubleValue();
26              double wa, sa, seki, syou;
27              wa = a + b;
28              sa = a - b;
29              seki = a * b;
30              syou = a / b;
31              System.out.printf("%20.15f %20.15f %20.15f %20.15f\n", wa, sa, seki, syou);
32              s = String.format("%20.15f %20.15f %20.15f %20.15f\n", wa, sa, seki, syou);
33              out.write(s);
34              out.newLine();
35              out.close();
36              in.close();
37          }
38          catch (Exception e) {
39              e.printStackTrace();
40          }
41      }
42
43  }
```

A 熱方程式プログラム (いつまでも工事中)

A.1 1998 年秋の挑戦

とにかく Java を試そうとして、結構厚手の本を携帯して勉強しつつ、書き上げたのが “Heat1d_e_?.java” である。“Heat1d_e_5.java” は Canvas を使えと本に書いてあったからそうしたのだが、本当のところどうなのだろう。

結局計算は全部 paint() の中でやっているという手抜きで、スレッドを勉強して何とか解

決しようと考えたのだが…

A.1.1 Heat1d_e_3.java

次の Heat1d_e_3.java[29] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/Heat1d_e_3.html) は最初に書いた叩き台プログラムである。

```
 1  /*
 2   * Heat1d_e_3.java
 3   * 1998 年 9 月 20 日（子供が産まれるちょっと前だったわけか）
 4   *
 5   * コンパイル: javac Heat1d_e_3.java
 6   * 実行:       appletviewer Heat1d_e_3.java
 7   *
 8   * <APPLET code="Heat1d_e_3.class" width=500 height=500></APPLET>
 9   *
10   * 注意
11   *   何と言っても古いので JDK1.1 以降では警告される。
12   *  (1) reshape() よりも setBounds() を使えと言われる（置き換えれば OK)。
13   *  (2) action() もよせ、と言われる。
14   */
15
16  import java.applet.*;
17  import java.awt.*;
18
19  public class Heat1d_e_3 extends Applet {
20
21    static final boolean DEBUG = false; // true;
22    private static final String message = "1D Heat Equation";
23    private int N = 40;
24    private double Tmax = 0.5, lambda = 0.5;
25    // 座標系の変換のためのパラメーター
26    private double ratiox, ratioy, X0, Y0;
27    // ユーザーとのインターフェイス（パラメーターの入力)
28    private Label labelLambda, labelN;
29    private TextField inputLambda, inputN;
30    private Button startB;
31
32    // 準備（変数の用意と座標系の初期化など)
33    public void init() {
34      //
35      setLayout(null);
36
37      labelLambda = new Label("lambda (should be <= 1/2)");
38      add(labelLambda);
39      labelLambda.setBounds(100, 100, 200, 30);
40      inputLambda = new TextField("" + lambda);
41      add(inputLambda);
42      inputLambda.setBounds(300, 100, 100, 30);
43
44      labelN = new Label("N");
45      add(labelN);
46      labelN.setBounds(100, 130, 200, 30);
47      inputN = new TextField("" + N);
48      add(inputN);
```

[29] http://www.math.meiji.ac.jp/~mk/labo/java/heat/Heat1d_e_3.java

```
49      inputN.setBounds(300, 130, 100, 30);
50
51      startB = new Button("Restart");
52      add(startB);
53      startB.setBounds(250, 180, 50, 30);
54
55      space(-0.1, -0.1, 1.1, 1.1);
56    }
57    // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
58    public boolean action(Event evt, Object what) {
59      if (evt.target == startB) {
60        String str1 = inputLambda.getText();
61        String str2 = inputN.getText();
62        lambda = Double.valueOf(str1).doubleValue();
63        N = Integer.parseInt(str2);
64        repaint();
65      }
66      return true;
67    }
68
69    // 初期条件
70    private double f(double x) {
71      if (x <= 0.5)
72        return x;
73      else
74        return 1-x;
75    }
76    // 座標変換の準備
77    private void space(double x0, double y0, double x1, double y1) {
78      X0 = x0; Y0 = y0;
79      ratiox = 500 / (x1 - x0);
80      ratioy = 500 / (y1 - y0);
81    }
82    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
83    private int wx(double x) {
84      return (int)(ratiox * (x - X0));
85    }
86    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
87    private int wy(double y) {
88      return 500 - (int)(ratioy * (y - Y0));
89    }
90    // x[], y[] の内容をグラフにする
91    private void drawGaph(Graphics g, double x[], double u[]) {
92      for (int i= 0; i < N; i++)
93        g.drawLine(wx(x[i]), wy(u[i]), wx(x[i + 1]), wy(u[i + 1]));
94    }
95
96    public void paint(Graphics g) {
97
98      double h = 1.0 / N;
99      double tau = lambda * h * h;
100     double dt = 0.01;
101     int Jmax = (int) (Tmax / tau);
102     int skip = (int) (dt / tau + 0.5);
103     double [] u, unext, x;
104
105     // ベクトルを確保する
106     x = new double[N+1];
```

```
107       u = new double[N+1];
108       unext = new double[N+1];
109       for (int i = 0; i <= N; i++)
110         x[i] = i * h;
111
112       // これはあんまり意味がない。
113       g.setColor(Color.pink);
114       g.fillOval(10, 10, 330, 80);
115       g.setColor(Color.red);
116       for (int i=0; i<4;i++)
117         g.drawOval(10-i, 10-i, 330+2*i, 80+2*i);
118
119       // タイトルを表示する
120       g.setColor(Color.black);
121       g.setFont(new Font("Helvetica", Font.BOLD, 24));
122       g.drawString(message, 40, 75);
123
124       // 初期値を計算する
125       for (int i = 0; i <= N; i++)
126         u[i] = f(i * h);
127
128       // 初期値のグラフを描く
129       drawGaph(g, x, u);
130
131       // 時間に関するループ
132       for (int j = 1; j <= Jmax; j++) {
133         int i;
134         for (i = 1; i < N; i++)
135           unext[i] = (1 - 2 * lambda) * u[i]
136                   + lambda * (u[i-1] + u[i+1]);
137         for (i = 1; i < N; i++)
138           u[i] = unext[i];
139         u[0] = 0; u[N] = 0;
140
141         if (DEBUG)
142           for (i = 0; i <= N; i++)
143             System.out.println("u[" + i + "]=" + u[i]);
144         // 適当な間隔 (dt=0.01) でグラフを描く
145         if (j % skip == 0)
146           drawGaph(g, x, u);
147       }
148     }
149   }
```

A.1.2 Heat1d_e_5.java

次の Heat1d_e_5.java[30] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/Heat1d_e_5.html) は Canvas を使うように書き直したプログラムである。

```
1  /*
2   * Heat1d_e_5.java -- Canvas を利用したバージョン
3   * 1998 年 11 月 20 日
4   *
5   * コンパイル: javac Heat1d_e_5.java
6   * 実行:       appletviewer Heat1d_e_5.java
```

[30] http://www.math.meiji.ac.jp/~mk/labo/java/heat/Heat1d_e_5.java

```
 7  *
 8  * <APPLET code="Heat1d_e_5.class" width=500 height=500></APPLET>
 9  *
10  * 注意
11  *   何と言っても古いので JDK1.1 以降では警告される。
12  *  (1) reshape() よりも setBounds() を使えと言われる (置き換えれば OK)。
13  *  (2) action() もよせ、と言われる。
14  */
15
16 import java.applet.*;
17 import java.awt.*;
18
19 public class Heat1d_e_5 extends Applet {
20
21   private int N = 20;
22   private double lambda = 0.5;
23   private double Tmax = 0.5;
24   // ユーザーとのインターフェイス (パラメーターの入力)
25   private Label labelLambda, labelN, labelTmax;
26   private TextField inputLambda, inputN, inputTmax;
27   private Button startB;
28   // 差分法による熱方程式シミュレーション
29   private HeatCanvas c1;
30
31   // 準備 (変数の用意と座標系の初期化など)
32   public void init() {
33     //
34     setLayout(null);
35     // λ=(τ/h^2) を入力するためのテキスト・フィールド
36     labelLambda = new Label("lambda (should be <= 1/2)");
37     add(labelLambda);
38     labelLambda.reshape(100, 10, 200, 30);
39     inputLambda = new TextField("" + lambda);
40     add(inputLambda);
41     inputLambda.reshape(300, 10, 100, 30);
42     // N (区間の分割数) を入力するためのテキスト・フィールド
43     labelN = new Label("N");
44     add(labelN);
45     labelN.reshape(100, 45, 200, 30);
46     inputN = new TextField("" + N);
47     add(inputN);
48     inputN.reshape(300, 45, 100, 30);
49     // Tmax (計算時間) を入力するためのテキスト・フィールド
50     labelTmax = new Label("Tmax");
51     add(labelTmax);
52     labelTmax.reshape(100, 80, 200, 30);
53     inputTmax = new TextField("" + Tmax);
54     add(inputTmax);
55     inputTmax.reshape(300, 80, 100, 30);
56     // 再計算ボタン
57     startB = new Button("Restart");
58     add(startB);
59     startB.reshape(420, 60, 50, 30);
60     // 紙芝居「熱方程式」キャンバス
61     c1 = new HeatCanvas();
62     add(c1);
63     c1.reshape(50, 100, 400, 400);
64     c1.compute(lambda, N, Tmax);
```

```
65    }
66
67    // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
68    public boolean action(Event evt, Object what) {
69      if (evt.target == startB) {
70        String str1 = inputLambda.getText();
71        String str2 = inputN.getText();
72        String str3 = inputTmax.getText();
73        lambda = Double.valueOf(str1).doubleValue();
74        N = Integer.parseInt(str2);
75        Tmax = Double.valueOf(str3).doubleValue();
76        c1.compute(lambda, N, Tmax);
77      }
78      return true;
79    }
80
81  }
82
83  // HeatCanvas.java
84  // 1998 年 11 月 20 日
85
86  class HeatCanvas extends Canvas {
87
88    static final boolean DEBUG = false; // true;
89    private static final boolean Debug = false;
90    private static final String message = "1D Heat Equation";
91    // 問題に取って基本的なパラメーター
92    //   N:      区間の分割数
93    //   Tmax:   最終時刻
94    //   lambda: λ=τ/h^2
95    private int N;
96    private double Tmax, lambda;
97    // 座標系の変換のためのパラメーター
98    private int CanvasX = 400, CanvasY = 400;
99    private double ratiox, ratioy, X0, Y0;
100   //
101   private HeatCanvas c1;
102
103   // コンストラクター
104   public HeatCanvas() {
105     super();
106     space(-0.1, -0.1, 1.1, 1.1);
107   }
108   public HeatCanvas(int cx, int cy) {
109     super();
110     CanvasX = cx; CanvasY = cy;
111     space(-0.1, -0.1, 1.1, 1.1);
112   }
113
114   public void compute(double lambda, int N, double Tmax) {
115     this.lambda = lambda;
116     this.N = N;
117     this.Tmax = Tmax;
118     repaint();
119   }
120
121   // 初期条件
122   private double f(double x) {
```

```
123       if (x <= 0.5)
124         return x;
125       else
126         return 1.0 - x;
127     }
128     // 座標変換の準備
129     private void space(double x0, double y0, double x1, double y1) {
130       X0 = x0; Y0 = y0;
131       ratiox = CanvasX / (x1 - x0);
132       ratioy = CanvasY / (y1 - y0);
133     }
134     // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
135     private int wx(double x) {
136       return (int)(ratiox * (x - X0));
137     }
138     // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
139     private int wy(double y) {
140       return CanvasY - (int)(ratioy * (y - Y0));
141     }
142     // x[], y[] の内容をグラフにする
143     private void drawGaph(Graphics g, double x[], double u[]) {
144       for (int i= 0; i < N; i++)
145         g.drawLine(wx(x[i]), wy(u[i]), wx(x[i + 1]), wy(u[i + 1]));
146     }
147
148     public void paint(Graphics g) {
149
150       double h = 1.0 / N;
151       double tau = lambda * h * h;
152       double dt = 0.01;
153       int Jmax = (int) (Tmax / tau);
154       int skip = (int) (dt / tau + 0.5);
155       double [] u, unext, x;
156
157       // ベクトルを確保する
158       x = new double[N+1];
159       u = new double[N+1];
160       unext = new double[N+1];
161       for (int i = 0; i <= N; i++)
162         x[i] = i * h;
163
164       if (Debug) {
165           // これはあんまり意味がない。
166           g.setColor(Color.pink);
167           g.fillOval(10, 10, 330, 60);
168           g.setColor(Color.red);
169           for (int i=0; i<4;i++)
170             g.drawOval(10-i, 10-i, 330+2*i, 60+2*i);
171
172           // タイトルを表示する
173           g.setColor(Color.black);
174           g.setFont(new Font("Helvetica", Font.BOLD, 24));
175           g.drawString(message, 40, 50);
176       }
177
178       // 初期値を計算する
179       for (int i = 0; i <= N; i++)
180         u[i] = f(i * h);
```

```
181
182       // 初期値のグラフを描く
183       drawGaph(g, x, u);
184
185       // 時間に関するループ
186       for (int j = 1; j <= Jmax; j++) {
187         int i;
188         for (i = 1; i < N; i++)
189           unext[i] = (1 - 2 * lambda) * u[i]
190                   + lambda * (u[i-1] + u[i+1]);
191         for (i = 1; i < N; i++)
192           u[i] = unext[i];
193         u[0] = 0; u[N] = 0;
194
195         if (DEBUG)
196           for (i = 0; i <= N; i++)
197             System.out.println("u[" + i + "]=" + u[i]);
198         // 適当な間隔 (dt=0.01) でグラフを描く
199         if (j % skip == 0)
200           drawGaph(g, x, u);
201       }
202     }
203   }
```

A.2 2001年1月の作業

Heat1dyou.java[31] は(実行はhttp://www.math.meiji.ac.jp/~mk/labo/java/sample/Heat1dyou.html)、大したことはしていない(イベント処理を JDK 1.1 以降のそれに置き換えた)。

```
1  /*
2   * Heat1dyou.java --- Heat1d_e_4.java を JDK 1.2 に合わせて変更したもの
3   * 2001年1月18日
4   *
5   * コンパイル: javac Heat1dyou.java
6   * 実行:       appletviewer Heat1dyou.java
7   *
8   * <applet code ="Heat1dyou" width=500 height=500></applet>
9   */
10
11  import java.applet.*;
12  import java.awt.*;
13  import java.awt.event.*;
14
15  public class Heat1dyou extends Applet implements ActionListener {
16
17      public int N = 20;
18      public double lambda = 0.5;
19      public double Tmax = 0.5;
20      // ユーザーとのインターフェイス
21      public Label labelLambda, labelN, labelTmax;
22      public TextField inputLambda, inputN, inputTmax;
23      public Button startB;
24      // 差分法による熱方程式シミュレーション
25      public HeatCanvas c1;
26
```

[31] http://www.math.meiji.ac.jp/~mk/labo/java/heat/Heat1dyou.java

```
27      public void init() {
28          //画面のレイアウト、レイアウトマネージャーの設定
29          setLayout(new BorderLayout());
30
31          //North パネルの設定
32          Panel pn = new Panel();
33          pn.setLayout(new GridLayout(3,2));
34
35          //ボタン、テキストフィールドの準備
36          String s = "lambda (should be <= 1/2)";
37          labelLambda = new Label(s, Label.LEFT);
38          pn.add(labelLambda);
39
40          inputLambda = new TextField("" + lambda,10);
41          pn.add(inputLambda);
42
43          labelN = new Label("N", Label.LEFT);
44          pn.add(labelN);
45
46          inputN = new TextField("" + N,10);
47          pn.add(inputN);
48
49          labelTmax = new Label("Tmax", Label.LEFT);
50          pn.add(labelTmax);
51
52          inputTmax = new TextField("" + Tmax,10);
53          pn.add(inputTmax);
54
55          inputLambda.addActionListener(this);
56          inputN.addActionListener(this);
57          inputTmax.addActionListener(this);
58
59          add(pn, "North");
60
61          //Center パネルの設定
62          Panel pc = new Panel();
63          c1 = new HeatCanvas();
64          c1.setSize(400,400);
65          c1.compute(lambda, N, Tmax);
66          pc.add(c1);
67          add(pc,"Center");
68
69          //East パネルの設定
70          Panel pe = new Panel();
71          startB = new Button("Restart");
72          pe.add(startB);
73          add(pe,"East");
74          startB.addActionListener(this);
75      }
76
77      //イベント処理（ボタン）
78      public void actionPerformed(ActionEvent e) {
79          if (e.getSource() == startB) {
80              String str1 = inputLambda.getText().trim();
81              String str2 = inputN.getText().trim();
82              String str3 = inputTmax.getText().trim();
83              lambda = Double.valueOf(str1).doubleValue();
84              N = Integer.parseInt(str2);
```

```
85              Tmax = Double.valueOf(str3).doubleValue();
86              c1.compute(lambda, N, Tmax);
87          }
88      }
89  }
90
91  // HeatCanvas.java
92
93  class HeatCanvas extends Canvas {
94
95      static final boolean DEBUG = false; // true;
96      private static final String message = "1D Heat Equation";
97      // 問題に取って基本的なパラメーター
98      //   N:      区間の分割数
99      //   Tmax:   最終時刻
100     //   lambda: λ=τ/h^2
101     private int N;
102     private double Tmax, lambda;
103     // 座標系の変換のためのパラメーター
104     private int CanvasX = 400, CanvasY = 400;
105     private double ratiox, ratioy, X0, Y0;
106     //
107     private HeatCanvas c1;
108
109     // コンストラクター
110     public HeatCanvas() {
111         super();
112         space(-0.1, -0.1, 1.1, 1.1);
113     }
114     public HeatCanvas(int cx, int cy) {
115         super();
116         CanvasX = cx; CanvasY = cy;
117         space(-0.1, -0.1, 1.1, 1.1);
118     }
119
120     public void compute(double lambda, int N, double Tmax) {
121         this.lambda = lambda;
122         this.N = N;
123         this.Tmax = Tmax;
124         repaint();
125     }
126
127     // 初期条件
128     private double f(double x) {
129         if (x <= 0.5)
130             return x;
131         else
132             return 1.0 - x;
133     }
134     // 座標変換の準備
135     private void space(double x0, double y0, double x1, double y1) {
136         X0 = x0; Y0 = y0;
137         ratiox = CanvasX / (x1 - x0);
138         ratioy = CanvasY / (y1 - y0);
139     }
140     // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
141     private int wx(double x) {
142         return (int)(ratiox * (x - X0));
```

```
143     }
144     // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
145     private int wy(double y) {
146         return CanvasY - (int)(ratioy * (y - Y0));
147     }
148     // x[], y[] の内容をグラフにする
149     private void drawGaph(Graphics g, double x[], double u[]) {
150         for (int i= 0; i < N; i++)
151             g.drawLine(wx(x[i]), wy(u[i]), wx(x[i + 1]), wy(u[i + 1]));
152     }
153
154     public void paint(Graphics g) {
155
156         double h = 1.0 / N;
157         double tau = lambda * h * h;
158         double dt = 0.01;
159         int Jmax = (int) (Tmax / tau);
160         int skip = (int) (dt / tau + 0.5);
161         double [] u, unext, x;
162
163         // ベクトルを確保する
164         x = new double[N+1];
165         u = new double[N+1];
166         unext = new double[N+1];
167         for (int i = 0; i <= N; i++)
168             x[i] = i * h;
169
170         // これはあんまり意味がない。
171         g.setColor(Color.pink);
172         g.fillOval(10, 10, 330, 60);
173         g.setColor(Color.red);
174         for (int i=0; i<4;i++)
175             g.drawOval(10-i, 10-i, 330+2*i, 60+2*i);
176
177         // タイトルを表示する
178         g.setColor(Color.black);
179         g.setFont(new Font("Helvetica", Font.BOLD, 24));
180         g.drawString(message, 40, 50);
181
182         // 初期値を計算する
183         for (int i = 0; i <= N; i++)
184             u[i] = f(i * h);
185
186         // 初期値のグラフを描く
187         drawGaph(g, x, u);
188
189         // 時間に関するループ
190         for (int j = 1; j <= Jmax; j++) {
191             int i;
192             for (i = 1; i < N; i++)
193                 unext[i] = (1 - 2 * lambda) * u[i]
194                     + lambda * (u[i-1] + u[i+1]);
195             for (i = 1; i < N; i++)
196                 u[i] = unext[i];
197             u[0] = 0; u[N] = 0;
198
199             if (DEBUG)
200                 for (i = 0; i <= N; i++)
```

```
201                     System.out.println("u[" + i + "]=" + u[i]);
202             // 適当な間隔 (dt=0.01) でグラフを描く
203             if (j % skip == 0)
204                 drawGaph(g, x, u);
205         }
206     }
207 }
```

A.3 2005年12月の再挑戦

何でこんなに時間がかかるのでしょうね…人生に疑問を感じつつ。目標はいくつかの懸案の問題を解決すること。

A.3.1 NewHeat1D.java

まずはダブル・バッファリングを使うバージョン NewHeat1D.java[32] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/NewHeat1D.html) から。

```
 1 /*
 2  * NewHeat1D.java
 3  * 2005/12/26
 4  *
 5  * コンパイル: javac NewHeat1D.java
 6  * 実行:       java NewHeat1D
 7  *
 8  * ダブルバッファリングを利用。一度しか計算しない。
 9  *
10  */
11
12 import java.awt.Frame;
13 import java.awt.Graphics;
14 import java.awt.Image;
15 import java.awt.event.WindowAdapter;
16 import java.awt.event.WindowEvent;
17
18
19 public class NewHeat1D extends Frame {
20     Image im = null;
21     Graphics g2 = null;
22     int i, n, nMax;
23     int N = 40;
24     double lambda = 0.5;
25     double Tmax = 10.0;
26     double dt = 0.01;
27     double h, tau;
28     int skip;
29     double [] u, newu;
30     double ratiox, ratioy, X0, Y0;
31     public NewHeat1D() {
32         this.setSize(500, 500);
33         this.addWindowListener(new WindowAdapter() {
34                 public void windowClosing(WindowEvent ev) {
35                     System.exit(0);
```

[32] http://www.math.meiji.ac.jp/~mk/labo/java/heat/NewHeat1D.java

```
36                  }
37              });
38          this.setVisible(true);
39      }
40      public static void main(String[] args) {
41          // TODO 自動生成されたメソッド・スタブ
42          new NewHeat1D();
43      }
44      private void space(double x0, double y0, double x1, double y1) {
45          X0 = x0; Y0 = y0;
46          ratiox = 500 / (x1 - x0);
47          ratioy = 500 / (y1 - y0);
48      }
49      private int wx(double x) {
50          return (int)(ratiox * (x - X0));
51      }
52      private int wy(double y) {
53          return 500 - (int)(ratioy * (y - Y0));
54      }
55      private double f(double x) {
56          if (x <= 0.5)
57              return x;
58          else
59              return 1.0 - x;
60      }
61      public void paint(Graphics g) {
62          if (im == null) {
63              im = this.createImage(this.getWidth(), this.getHeight());
64              g2 = im.getGraphics();
65              space(-0.1, -0.1, 1.1, 1.1);
66              h = 1.0 / N;
67              tau = lambda * h * h;
68              skip = (int)Math.rint(dt / tau);
69              nMax = (int)Math.rint(Tmax / tau);
70              u = new double [N+1];
71              newu = new double [N+1];
72              // 初期値
73              for (i = 0; i <= N; i++)
74                  u[i] = f(i * h);
75              // 初期値のグラフを描く
76              for (i = 0; i < N; i++)
77                  g2.drawLine(wx(i * h), wy(u[i]), wx((i + 1) * h), wy(u[i+1]));
78              // 時間発展
79              for (n = 1; n <= nMax; n++) {
80                  // 陽的差分法
81                  for (i = 1; i < N; i++)
82                      newu[i] = (1 - 2 * lambda) * u[i] + lambda * (u[i + 1] + u[i - 1]);
83                  for (i = 1; i < N; i++)
84                      u[i] = newu[i];
85                  // 同次 Dirichlet 境界条件
86                  u[0] = 0.0;
87                  u[N] = 0.0;
88                  // 時刻が dt の整数倍のときにグラフを描く
89                  if (n % skip == 0) {
90                      for (i = 0; i < N; i++)
91                          g2.drawLine(wx(i * h), wy(u[i]), wx((i + 1) * h), wy(u[i+1]));
92                  }
93                  System.out.println(""+ n * tau);
```

```
94              }
95          }
96          g.drawImage(im, 0, 0, this);
97      }
98  }
```

A.3.2 NewHeat1Dv3.java

スレッドを使うと次の NewHeat1Dv3.java[33] (実行は http://www.math.meiji.ac.jp/~mk/labo/java/sample/NewHeat1Dv3.html) のようになる。

```
1  /*
2   * NewHeat1Dv3.java
3   *   version 2.0: 計算スレッドを用意した。
4   *   version 3.0: ボタン、テキスト・フィールドなどを用意して再計算可能にした。
5   *   version 4.0: (予定) 陰解法、境界条件の一般化
6   *   version 5.0: (予定) 初期条件の種類を増やす
7   */
8
9   import java.awt.Button;
10  import java.awt.Color;
11  import java.awt.Frame;
12  import java.awt.Graphics;
13  import java.awt.Image;
14  import java.awt.Label;
15  import java.awt.TextField;
16  import java.awt.event.ActionEvent;
17  import java.awt.event.ActionListener;
18  import java.awt.event.WindowAdapter;
19  import java.awt.event.WindowEvent;
20
21  public class NewHeat1Dv3 extends Frame implements Runnable,ActionListener {
22      /**
23       *
24       */
25      private static final long serialVersionUID = 1L;
26      /**
27       *
28       */
29      static final int ImgX = 500, ImgY = 500;
30      static final int WinX = ImgX + 200, WinY = ImgY;
31      Thread th = null;
32      Image im = null;
33      Graphics bg = null;
34      int i, n = -1, nMax;
35      int N;
36      double lambda, Tmax, dt, theta;
37      double h, tau;
38      int skip;
39      double [] x, u, newu;
40      double ratiox, ratioy, X0, Y0;
41      private String[] labelStr={"N", "lamba", "Tmax", "dt", "theta"};
42      private Label[] label;
43      private String[] txStr={"100", "0.5", "1.0", "0.01", "0.5"};
44      private TextField[] tf;
```

[33] http://www.math.meiji.ac.jp/~mk/labo/java/heat/NewHeat1Dv3.java

```
45      private String[] btStr = {"Start", "End"};
46      private Button[] bt;
47      private void readParameters() {
48          N = Integer.parseInt(tf[0].getText());
49          lambda = Double.valueOf(tf[1].getText()).doubleValue();
50          Tmax = Double.valueOf(tf[2].getText()).doubleValue();
51          dt = Double.valueOf(tf[3].getText()).doubleValue();
52          theta = Double.valueOf(tf[4].getText()).doubleValue();
53      }
54      public NewHeat1Dv3() {
55          this.setSize(WinX, WinY);
56          this.addWindowListener(new WindowAdapter() {
57                  public void windowClosing(WindowEvent ev) {
58                      System.exit(0);
59                  }
60              });
61          setLayout(null);
62          bt = new Button[btStr.length];
63          for (int i = 0; i < bt.length; i++) {
64              bt[i] = new Button(btStr[i]);
65              bt[i].addActionListener(this);
66              bt[i].setBounds(ImgX, (i+2)*20, (WinX - ImgX)/2, 20);
67              add(bt[i]);
68          }
69          label = new Label[labelStr.length];
70          tf = new TextField[txStr.length];
71          for (int i = 0; i < labelStr.length; i++) {
72              label[i] = new Label(labelStr[i]);
73              tf[i] = new TextField(txStr[i]);
74              label[i].setBounds(ImgX, (i+bt.length+2)*20, (WinX - ImgX) / 3, 20);
75              tf[i].setBounds(ImgX + (WinX-ImgX)/3, (i+bt.length+2)*20, (WinX-ImgX)/2, 20);
76              add(label[i]);
77              add(tf[i]);
78          }
79          this.setVisible(true);
80          bt[0].setEnabled(false);
81          space(-0.1, -0.1, 1.1, 1.1);
82          // 計算開始の準備
83          initcomputation();
84          // 計算スレッドを開始する
85          this.start();
86      }
87      public void start() {
88          if (th == null) {
89              th = new Thread(this);
90              th.start();
91          }
92      }
93      public void stop() {
94          if (th != null) {
95              th = null;
96          }
97      }
98      public static void main(String[] args) {
99          new NewHeat1Dv3();
100     }
101     // 座標変換の仕組み
102     private void space(double x0, double y0, double x1, double y1) {
```

```
103         X0 = x0; Y0 = y0;
104         ratiox = ImgX / (x1 - x0);
105         ratioy = ImgY / (y1 - y0);
106     }
107     // x 座標をユーザー座標からウィンドウ座標に
108     private int wx(double x) {
109         return (int)Math.rint(ratiox * (x - X0));
110     }
111     // y 座標をユーザー座標からウィンドウ座標に
112     private int wy(double y) {
113         return ImgY - (int)Math.rint(ratioy * (y - Y0));
114     }
115     // 初期データ
116     private double f(double x) {
117         if (x <= 0.5)
118             return x;
119         else
120             return 1.0 - x;
121     }
122     // 現在の u[] をグラフ化する
123     private void drawGraph() {
124         if (bg == null)
125             repaint();
126         for (int i = 0; i < N; i++)
127             bg.drawLine(wx(x[i]), wy(u[i]), wx(x[i+1]), wy(u[i+1]));
128     }
129     // ダブルバッファリングの定跡
130     public void paint(Graphics g) {
131         if (im == null) {
132             // createImage() はコンストラクターでは使えず、このタイミングでするしかないとか。
133             im = this.createImage(this.getWidth(), this.getHeight());
134             bg = im.getGraphics();
135         }
136         g.drawImage(im, 0, 0, this);
137     }
138     private void initcomputation() {
139         // パラメーターを読む
140         readParameters();
141         // 刻み幅を決める
142         h = 1.0 / N;
143         tau = lambda * h * h;
144         //
145         skip = (int)Math.rint(dt / tau);
146         if (skip <= 0) skip = 1;
147         nMax = (int)Math.ceil(Tmax / tau);
148         // ベクトルを 3 本用意する
149         x = new double [N+1];
150         u = new double [N+1];
151         newu = new double [N+1];
152         //
153         n = -1;
154         System.out.println("init");
155     }
156     // 計算スレッド
157     public void run() {
158         while (th != null) {
159             if (n == -1) {
160                 // 初期値を計算する
```

```
161                 for (int i = 0; i <= N; i++)
162                     x[i] = i * h;
163                 for (i = 0; i <= N; i++)
164                     u[i] = f(i * h);
165                 n++;
166                 repaint();
167                 bg.setColor(Color.white);
168                 bg.fillRect(0, 0, ImgX, ImgY);
169                 bg.setColor(Color.black);
170                 drawGraph();
171                 repaint();
172                 try {
173                     Thread.sleep(10);
174                 } catch (Exception ex) {
175                     ex.printStackTrace();
176                 }
177             }
178             else if (n < nMax) {
179                 do {
180                     for (i = 1; i < N; i++)
181                         newu[i] = (1 - 2 * lambda) * u[i] + lambda * (u[i + 1] + u[i - 1]);
182                     for (i = 1; i < N; i++)
183                         u[i] = newu[i];
184                     u[0] = 0.0;
185                     u[N] = 0.0;
186                     n++;
187                 } while (n % skip != 0);
188                 drawGraph();
189                 repaint();
190                 try {
191                     Thread.sleep(10);
192                 } catch (Exception ex) {
193                     ex.printStackTrace();
194                 }
195                 // System.out.println(""+ n * tau);
196             }
197             Thread.yield();
198         }
199     }
200     public void actionPerformed(ActionEvent e) {
201         // TODO 自動生成されたメソッド・スタブ
202         if (e.getSource() == bt[0]) {
203             bt[0].setEnabled(false);
204             bt[1].requestFocus();
205             bt[1].setEnabled(true);
206             initcomputation();
207             start();
208         }
209         if (e.getSource() == bt[1]) {
210             bt[0].setEnabled(true);
211             bt[0].requestFocus();
212             bt[1].setEnabled(false);
213             stop();
214         }
215     }
216 }
```

A.4 2006年1月の再挑戦

A.4.1 NewHeat1Dv4.java

各「コマ」を JPEG ファイルとして出力するようにした NewHeat1Dv4.java[34] である。これはファイル入出力をするため、アプレットでなく、アプリケーションにしてある。

それら JPEG ファイルを TMPEGEnc で MPEG ファイルに変換することに成功した。作成した MPEG ファイルを http://www.math.meiji.ac.jp/~mk/labo/java/NewHeat1Dv4.mpg においておく。

MPEG 作成の要点は

(1) イメージのサイズは縦横ともに 8 の倍数にする。

(2) JPEG ファイルの名前は「連番」とする。固定した名前＋連続した非負整数とし、番号は飛びがないようにしておく。

(3) TMPEGEnc で「ストリームの種類」は “System (Video のみ)”, 「設定」の「フレームレート」で標準より少しフレーム・レートを落とす。「映像ソース (V)」として、**最初の** JPEG ファイル名を指定する。「出力ファイル名」は好きなようにつけて、[圧縮開始] ボタンを押す。

```
 1  /*
 2   * NewHeat1Dv4.java --- 1 次元熱方程式
 3   *   コンパイルはもちろん javac NewHeat1Dv4.java
 4   *   実行は java NewHeat1Dv4
 5   *   スレッド周辺はまだ不完全というか良く分かっていない。
 6   *   TMPEGEnc で MPEG ファイルを作るためにイメージのサイズを 8 の倍数にした。
 7   *   実際に MPEG ファイルを作ることに成功した。
 8   *   映像ソースとして最初の JPEG ファイルを指定し、
 9   *   ストリームの種類 System(Video のみ), 設定でフレーム・レートを 7.992fps（つまり標準÷ 3）
10   *   そして圧縮開始するだけである。
11   *
12   *   今後の改良計画
13   *   (1) 境界条件色々に対応
14   *       http://www.math.meiji.ac.jp/~mk/labo/text/head-fdm-1.pdf 参照
15   *   (2) スレッドをきちんと理解してきちんと書く
16   *   (3) 2 次元化する
17   */
18  package jp.tuyano.eclipsebook3;
19
20  import java.awt.Button;
21  import java.awt.Color;
22  import java.awt.Frame;
23  import java.awt.Graphics;
24  import java.awt.Label;
25  import java.awt.TextField;
26  import java.awt.event.ActionEvent;
27  import java.awt.event.ActionListener;
28  import java.awt.event.WindowAdapter;
29  import java.awt.event.WindowEvent;
30  import java.awt.image.BufferedImage;
31  import java.io.File;
```

[34] http://www.math.meiji.ac.jp/~mk/labo/java/heat/NewHeat1Dv4.java

```
32 import javax.imageio.ImageIO;
33
34
35 public class NewHeat1Dv4 extends Frame implements Runnable,ActionListener {
36
37     private static final long serialVersionUID = 1L;
38     /**
39      *
40      */
41     static final int ImgX = 496, ImgY = 496;
42     static final int WinX = ImgX + 200, WinY = ImgY;
43     Thread th = null;
44     BufferedImage im = null;
45     Graphics bg = null;
46     int i, n = -1, nMax;
47     int N;
48     double lambda, Tmax, dt, theta;
49     double h, tau;
50     int skip;
51     double [] x, u, newu;
52     double ratiox, ratioy, X0, Y0;
53     private boolean result = false;
54     private int jpeg_file_number = 0;
55     // GUI
56     private String[] labelStr={"N", "lamba", "Tmax", "dt", "theta"};
57     private Label[] label;
58     private String[] txStr={"100", "0.5", "1.0", "0.01", "0.5"};
59     private TextField[] tf;
60     private String[] btStr = {"Start", "End"};
61     private Button[] bt;
62     // テキスト・フィールドに入力されたパラメーターを読む
63     private void readParameters() {
64         N = Integer.parseInt(tf[0].getText());
65         lambda = Double.valueOf(tf[1].getText()).doubleValue();
66         Tmax = Double.valueOf(tf[2].getText()).doubleValue();
67         dt = Double.valueOf(tf[3].getText()).doubleValue();
68         theta = Double.valueOf(tf[4].getText()).doubleValue();
69     }
70     public NewHeat1Dv4() {
71         this.setSize(WinX, WinY);
72         this.addWindowListener(new WindowAdapter() {
73             public void windowClosing(WindowEvent ev) {
74                 System.exit(0);
75             }
76         });
77         setLayout(null);
78         bt = new Button[btStr.length];
79         for (int i = 0; i < bt.length; i++) {
80             bt[i] = new Button(btStr[i]);
81             bt[i].addActionListener(this);
82             bt[i].setBounds(ImgX, (i+2)*20, (WinX - ImgX)/2, 20);
83             add(bt[i]);
84         }
85         label = new Label[labelStr.length];
86         tf = new TextField[txStr.length];
87         for (int i = 0; i < labelStr.length; i++) {
88             label[i] = new Label(labelStr[i]);
89             tf[i] = new TextField(txStr[i]);
```

```
90              label[i].setBounds(ImgX, (i+bt.length+2)*20, (WinX - ImgX) / 3, 20);
91              tf[i].setBounds(ImgX + (WinX-ImgX)/3, (i+bt.length+2)*20, (WinX-ImgX)/2, 20);
92              add(label[i]);
93              add(tf[i]);
94          }
95          this.setVisible(true);
96          bt[0].setEnabled(false);
97          // ダブル・バッファリング用のバッファーを準備
98          if (im == null) {
99              im = new BufferedImage(ImgX, ImgY, BufferedImage.TYPE_3BYTE_BGR);
100             bg = im.getGraphics();
101         }
102         space(-0.1, -0.1, 1.1, 1.1);
103         // 計算開始の準備
104         initcomputation();
105         // 計算スレッドを開始する
106         this.start();
107     }
108     public void start() {
109         if (th == null) {
110             th = new Thread(this);
111             th.start();
112         }
113     }
114     public void stop() {
115         if (th != null) {
116             th = null;
117         }
118     }
119     public static void main(String[] args) {
120         new NewHeat1Dv4();
121     }
122     // 座標変換の仕組み
123     private void space(double x0, double y0, double x1, double y1) {
124         X0 = x0; Y0 = y0;
125         ratiox = ImgX / (x1 - x0);
126         ratioy = ImgY / (y1 - y0);
127     }
128     // x 座標をユーザー座標からウィンドウ座標に
129     private int wx(double x) {
130         return (int)Math.rint(ratiox * (x - X0));
131     }
132     // y 座標をユーザー座標からウィンドウ座標に
133     private int wy(double y) {
134         return ImgY - (int)Math.rint(ratioy * (y - Y0));
135     }
136     // 初期データ
137     private double f(double x) {
138         if (x <= 0.5)
139             return x;
140         else
141             return 1.0 - x;
142     }
143     // 現在の u[] をグラフ化する
144     private void drawGraph() {
145         for (int i = 0; i < N; i++)
146             bg.drawLine(wx(x[i]), wy(u[i]), wx(x[i+1]), wy(u[i+1]));
147     }
```

```
148     // ダブルバッファリングの定跡
149     public void paint(Graphics g) {
150         g.drawImage(im, 0, 0, this);
151     }
152     // 計算の準備 (パラメーターを読み、変数の準備)
153     private void initcomputation() {
154         // パラメーターを読む
155         readParameters();
156         // 刻み幅を決める
157         h = 1.0 / N;
158         tau = lambda * h * h;
159         //
160         skip = (int)Math.rint(dt / tau);
161         if (skip <= 0) skip = 1;
162         nMax = (int)Math.ceil(Tmax / tau);
163         // ベクトルを 3 本用意する
164         x = new double [N+1];
165         u = new double [N+1];
166         newu = new double [N+1];
167         //
168         n = -1;
169     }
170     // 現在のバッファー im の内容を JPEG 形式で書き出す
171     void saveJpeg() {
172         String str;
173         if (jpeg_file_number < 10)
174             str = "0000" + jpeg_file_number;
175         else if (jpeg_file_number < 100)
176             str = "000" + jpeg_file_number;
177         else if (jpeg_file_number < 1000)
178             str = "00" + jpeg_file_number;
179         else if (jpeg_file_number < 10000)
180             str = "0" + jpeg_file_number;
181         else
182             str = "" + jpeg_file_number;
183         try {
184               result = ImageIO.write(im, "jpeg", new File("test" + str + ".jpg"));
185               jpeg_file_number++;
186             }
187             catch (Exception e) {
188               e.printStackTrace();
189               result = false;
190             }
191     }
192     // 計算スレッド
193     public void run() {
194         while (th != null) {
195             if (n == -1) {
196                 // 初期値を計算する
197                 for (int i = 0; i <= N; i++)
198                     x[i] = i * h;
199                 for (i = 0; i <= N; i++)
200                     u[i] = f(i * h);
201                 n++;
202                 repaint();
203                 bg.setColor(Color.white);
204                 bg.fillRect(0, 0, ImgX, ImgY);
205                 bg.setColor(Color.black);
```

```
206                 drawGraph();
207                 saveJpeg();
208                 repaint();
209                 try {
210                     Thread.sleep(10);
211                 } catch (Exception ex) {
212                     ex.printStackTrace();
213                 }
214             }
215             else if (n < nMax) {
216                 do {
217                     for (i = 1; i < N; i++)
218                         newu[i] = (1 - 2 * lambda) * u[i] + lambda * (u[i + 1] + u[i - 1]);
219                     for (i = 1; i < N; i++)
220                         u[i] = newu[i];
221                     u[0] = 0.0;
222                     u[N] = 0.0;
223                     n++;
224                 } while (n % skip != 0);
225                 drawGraph();
226                 saveJpeg();
227                 repaint();
228                 try {
229                     Thread.sleep(10);
230                 } catch (Exception ex) {
231                     ex.printStackTrace();
232                 }
233                 // System.out.println(""+ n * tau);
234             }
235         }
236     }
237     public void actionPerformed(ActionEvent e) {
238         // TODO 自動生成されたメソッド・スタブ
239         if (e.getSource() == bt[0]) {
240             bt[0].setEnabled(false);
241             bt[1].requestFocus();
242             bt[1].setEnabled(true);
243             initcomputation();
244             start();
245         }
246         if (e.getSource() == bt[1]) {
247             bt[0].setEnabled(true);
248             bt[0].requestFocus();
249             bt[1].setEnabled(false);
250             stop();
251         }
252     }
253 }
```

A.5 考えていること

最初は…

- 計算は専用スレッドでやるようにするか、その反対に表示の更新を (例えば Timer で) スレッドでやるか。

- 計算を専用のスレッドでやる場合、メインではとにかく初期値くらい表示して、計算スレッドを走らせる。計算スレッドの中で描画は Image に対して行ない、Image を画面に描いたかどうかフラグを用意しておき、Timer でチェックしてまだ描画していなければバッファーの内容を転送。

…なあんて考えていて、古い学生のプログラムを見ていたら、とっくに解決済みだった。
三井康之君の波動方程式のプログラムでは

```
public class 名前 extends Applet implements Runnable,ActionListener,ItemListener {
  Thread th = null;
  Graphics bg;
  Image buf;
  public void init() {
    ...
    buf = createImage(横幅, 縦高);
    bg = buf.getGraphics();
    ...
    // 以下 bg に描画する
  }
  public void start() {
    if (th == null) {
      th = new Thread(this);
      th.start();
    }
  }
  public void run() {
    while (running) {
      結構重い描画
      ...
      repaint();
    }
  }
  public void stop() {
    if (th != null) {
      running = false;
      th = null;,
    }
  }
  public void paint(Graphics g) {
    if (t == 0) {
      ...
    }
    g.drawImage(buf, 0, 0, this);
  }
}
```

田代航平君の渦糸のプログラムでは

```
public class クラス名 extends Applet implements Runnable,ActionListener {
  private Image off;
  private Thread th = null;
  // アプレットの最初は init() から
  public void init() {
  }
  public void paint(Graphics g) {
    update(g);
  }
  public void update(Graphics g) {
    g.drawImage(off, 0, 0, this);
  }
  public void start() {
    if (th == null) {
      th = new Thread(this);
      th.start();
    }
  }
  public void stop() {
    if (th != null) {
      th = null;
    }
  }
  public void run() {
    while (th != null) {
      // 計算
    }
  }
  public void actionPerformed(ActionEvent ev) {
    if (e.getSource() == bt[0]) {
      ...
      start();
    }
    else if (e.getSource() == bt[1]) {
      ...
      stop();
    }
  }
}
```

はてな？AWT のコンポーネントでは、再描画の要求があると、repaint(), update(), paint() の順に描画メソッドが呼び出されていく。何か描き足すときは paint() に…と物の本に書いてあった。田代君の paint() から update() を呼び出すのは無駄では？…いや、オーバーライドしているので、無駄は生じていない。しかし通常とは逆の呼び出し順になっていて気持ちが悪い。筆者としては

```
public void paint(Graphics g) {
  g.drawImage(off, 0, 0, this);
}
public void update(Graphics g) {
  paint(g);
}
```

と書きたい。通常 update() で行なわれるという、クリアする処理が省けている (狙いは田代

君も同じだろうけれど)。

B　MPEG ビデオ作成計画 (メモ)

(2006 年 1 月 7 日、試してみたらあっさり出来た。ここもそのうちに整理。)

個人的には剥製みたいな MPEG ファイルよりも、対話的なシミュレーション・プログラム＋台本みたいなのが理想だと思っているのだけど。

http://www.javadrive.jp/java2d/bufferedImage/index2.html

http://tokunagakenichi.net/java/sample/JPEG/JpegSave.java

```
import java.awt.image.*;

  BufferedImage im = null;
  ..
  // TYPE_INT_BGR というのもある
  im = new BufferedImage(幅, 高さ, BufferedImage.TYPE_3BYTE_BGR);
  ..
  Graphics g = im.getGraphics();
  g に色々描画

  getGraphics().drawImage(im, 0, 0, this); のようにして実際に表示

  // 出力するためのメソッド
  public void saveJpegFile(OutputStream output) throws IOException {
    JPEGImageEncode jpeg = JPEGCodec.createJPEGEncoder(output);
    jpeg.encode(image);
    output.flush();
    output.close();
  }

  // メソッドの呼び出し
  try {
    saveJpegFile(new FileOutputStream("test.jpg"));
  }
  catch (IOException e) {
    e.printStackTrace();
  }
```

- createImage(幅, 高さ) の代りに new BufferedImage()

```
BufferedImage im_w, im_r = null;
boolean result = false;

// im_w に用意してあれば次のようにして出力できる
try {
  result = ImageIO.write(im_w, "jpeg", new File("test.jpg"));
}
catch (Exception e) {
  e.printStackTrace();
  result = false;
}

// 読む方は
try {
  im_r = ImageIO.read(new File("my.png"));
}
catch (Exception e) {
  e.printStackTrace();
  im_r = null;
}
if (im_r == null) {
  im_r = new BufferedImage(...
}
Graphics bg = im_r.createGraphics();
...
```

C　学生のプログラム

C.1　空間 1 次元熱方程式

大葉敏文&佐藤晴郎 (2001 年 2 月 7 日) のプログラム一式 heat1.zip は http://www.math.meiji.ac.jp/~mk/labo/java/ から入手できる。

試しに実行したければ『一次元熱方程式の初期値境界値問題を差分方程式で解きグラフ化する。』[35] で。

C.2　波動方程式

三井康之 (2001 年 12 月 21 日) のプログラム一式 mitsui-windows.lzh や mitsui-knoppix.tar.gz は、http://www.math.meiji.ac.jp/~mk/labo/2007/ から入手できる。

[35] http://www.math.meiji.ac.jp/~haruo310/soturon/heat1d.html

試しに実行したければ「Java による波動方程式の差分解プログラムの公開テスト」[36] で。

C.3 渦糸

田代航平君の渦糸のプログラム (2003 年 2 月 11 日) はやはり `http://www.math.meiji.ac.jp/~mk/labo/java/` から入手できる。実行は`http://www.math.meiji.ac.jp/~mk/labo/java/uzu.html` で出来る。

D 他の人が作ったパッケージの利用

D.1 クラスパスの指定

javac や java コマンドを使う場合は、CLASSPATH 環境変数や、コマンドライン・オプションの `-classpath` (短縮形 `-cp`) を用いる。例えば

```
knoppix$ ls
jfftpack.jar  testfft.java
knoppix$ javac -cp jfftpack.jar testfft.java
knoppix$ java -cp jfftpack.jar:. testfft
```

(java コマンドの方は、`testfft.class` を見つけるために、カレント・ディレクトリィ "." の指定が必要である。)

環境変数を設定するには

Knoppix (bash) の場合

```
knoppix$ export CLASSPATH=~/jfftpack/jfftpack.jar:.
```

Windows の場合

```
C:\なんとか> set CLASSPATH=%CLASSPATH%;C:\なんとか\jfftpack.jar;.
```

という具合いである。

appletviewer の場合、`-cp` オプションはなく、環境変数も見ない (当然か)。後述の Colt ライブラリィの例では、jar コマンドで必要なクラスファイルを抜き出して利用した。

D.2 実例: jfftpack

FFTPACK の Java への移植である jfftpack が netlib で入手できる。ソースプログラムとして配布されているので、自分でコンパイルすることになる。

`ca.uol.aig.fftpack` という package 名であることに注意。

[36] `http://www.math.meiji.ac.jp/~ee88010/`

コンパイルして .jar ファイルを作る

```
mkdir -p ca/uol/aig/fftpack              (ディレクトリィを用意)
cp -p どこか/*.java ca/uol/aig/fftpack   (ソース・プログラムを用意)
javac ca/uol/aig/fftpack/*.java          (ソース・プログラムをコンパイル)
jar -cvf どこか/fftpack.jar .            (jar ファイルを作る)
jar -tvf どこか/fftpack.jar              (jar ファイルの中身を確認する)
```

D.3 実例: Colt library

Colt ライブラリィ[37]とは、CERN の Colt Project[38] で開発された、Java による科学技術計算のためのライブラリィである。

Bessel 関数だけが必要ならば、

```
knoppix$ jar xvf colt.jar cern/jet/math/Bessel.class
knoppix$ jar xvf colt.jar cern/jet/math/Constants.class
```

で cern/jet/math/{Bessel,Constants}.class だけ抜き出しておけば十分。

動作確認用 TestBessel.java

```
// TestBessel.java
import cern.jet.math.*;

public class TestBessel {
    public static void main(String []args) {
        System.out.println(""+Bessel.j0(1.0));
    }
}
```

D.4 実例: MitsuiWorld

(準備中)

E 参考書

入門書

書店に置いてないことも多いのだが、我々の目的には戸川 [3] は「役立つ」本である (別に [3] の目指しているところと我々のそれが同じと言うわけではないのだが、知りたいことがぎっ

[37]Copyright (c) 1999 CERN - European Organization for Nuclear Research.
[38]http://dsd.lbl.gov/~hoschek/colt/

しり詰まっていてお買得)。このコンパクトな本を肌身離さず持って短期間に集中的に勉強することが筆者としての一番のお奨め。

もう少し一般的な目的で、Java に関する説明もじっくり読んで理解してやろうという人には、定評のある結城 [4] を奨める。卒研の学生にはその時々で色々な参考書を使ってもらったが、のめり込んだ人の評判は高かった。筆者自身は初心者としてこの本を読んだことはないが、記述はていねいであることは良く分かるし、学生の証言もあるので、これから Java の勉強をはじめようという人に奨められると思う。

最近出た本で、柴田 [1] は評判が良いようである。C, C++ で人気のある著者が書いたというのが評価に関係している面もあると思うが、ていねいな感じで期待できる。一度ゼミで使ってみよう、と思っている。

グラフィックス目的で

2 変数関数のグラフ描きのために山本 [5] が役立った。

参考文献

[1] 柴田望洋, 明解 Java 入門編, ソフトバンククリエイティブ (2007).

[2] 掌田 津耶乃(しょうだ つやの), Eclipse 3 ではじめる Java プログラミング入門, 秀和システム (2005).

[3] 戸川 隼人, 演習と応用 Java, サイエンス社 (2004).

[4] 結城浩, 改訂第 2 版 Java 言語プログラミングレッスン (上), (下), ソフトバンククリエイティブ (株) (2005).

[5] 山本 芳人, Java による図形処理入門, 工学図書株式会社 (1998).
http://www.rs.kagu.sut.ac.jp/~yama/java/