社工情報処理実習 1
– 課題 8 の解答例 –

安藤 和敏
筑波大学社会工学系

2000 年 6 月 13 日

1. コイン投げ

プログラムはファイル 1 を見よ。ここでは、numOfheads メソッド, main メソッドという順番で書いてあるが、逆順でもかまわない。

いんちきコインは、

```
if (Math.random() <= 1/2.0) {
```

の条件式中の 1/2.0 を、例えば 1/3.0 とか 2/3.0 に変えれば良い。

2. 方程式を解く – Newton 法 –

プログラムの例はファイル 2を見よ。

別の方程式, 例えば $f(x) = x^2 - 2$ を解きたいならば, メソッド f と fdash を

```
public static double f (double x) {

    return x*x -2;

}

public static double fdash (double x) {

    return 2*x;

}
```

と定義して, x の初期値を適当に定めて (例えば, 9.0 のまま) やれば良い。実行結果は、

```
banana:~/java% java newton2
1.4142135667734679
```

となって、確かに $\sqrt{2}$ に近い値になっている。

でも $f(x) = x^2 - 2 = 0$ の解はもう一つ $(-\sqrt{2})$ ある。こんどは初期値を -9 としてやると、

```
banana:~/java% java newton2
-1.4142135667734679
```

ちゃんと, $-\sqrt{2}$ がでますね。すばらしい!

```
/*************************************************************************

        coin1.java:  コイン投げ

        安藤 和敏（筑波大学社会工学系）
        2000.6.9

 *************************************************************************/
public class coin1 {

    public static int numOfheads (int N) {  // numOfheads メソッドの定義

        int head = 0;                       // 表の出る回数を数える変数

        for (int i=0; i<N; i++) { // 以下を N 回繰り返す

            if (Math.random() <= 1/2.0) {   // もし表がでたら
                head ++;                    // head に 1 を足す
            }

        } // for 文の終わり

        return head;              // head を返す

    }                                       // numOfheads メソッドの定義終り

    public static void main (String[] args) { // main メソッドの定義

        for (int k=0; k<10; k++) {        // 以下を 10 回繰り返す

            System.out.println(numOfheads(100)); // 表の出た回数を表示

        }
    }                                        // main メソッドの定義終り
}
```

ファイル **1.1:** coin1.java

```
/*****************************************************************************

        newton.java:  ニュートン法

        安藤 和敏 (筑波大学社会工学系)
        2000.6.9

 *****************************************************************************/
public class newton {

    public static void main (String[] args) { // main メソッドの定義

        double x = 9.0;                     // x の初期値 = 9.0
        double epsilon = 0.0001;            // ε  = 0.0001

        while (Math.abs(f(x))>= epsilon) {  // |f(x)| ≧ ε  である間は、
            x = x - f(x)/fdash(x);           // x を 新しい x で置き換える。
        }

        System.out.println(x);              // 解の出力

    }

    public static double f (double x) {     // メソッド f(x) の定義

        return x*x*x/100 -1 ;

    }

    public static double fdash (double x) {   // メソッド fdash(x) の定義

        return 3*x*x/100;

    }

} // newton クラスの終わり
```

ファイル **2.1:** newton.java

3. 数値積分 (前回の続き)

プログラムの例はファイル 3を見よ。メソッドを使ったおかげで、前のプログラムより簡明になった。

別の関数の積分を求めたければ, `f` メソッドを書き変えてやれば良い。例えば同じ $[0,5]$ 区間での, 積分 $\int_0^5 x^2 \mathrm{d}x$ を求めたければ、

```
public static double f (double x) {

    return x*x;

}
```

と書きかえてやると, 本当の積分値は $\frac{125}{3} = 41.6666\cdots$ であるが

```
banana:~/java% java daikei2xsq
近似値 = 41.67500000000001
```

という結果が得られる。

```
/*****************************************************************************

        daikei2.java:  台形公式による積分の近似（第２版）

        安藤 和敏（筑波大学社会工学系）
        2000.6.9

 *****************************************************************************/
public class daikei2 {
    public static void main (String[] args) {

        double a = 0.0;         // 積分区間の左端
        double b = 5.0;         // 積分区間の右端
        int N = 50;             // 分割した区間の数

        double h = (b-a)/N;        // 区間の幅 0.1 = (5-0)/50

        double sum = 0;   // 面積の途中計算結果を入れるための変数

        for (int i=0; i<N; i++) { // 以下を N 回繰り返す

            sum = sum + (f(i*h)+f((i+1)*h))*h/2;  // f(x) の呼び出し

        }                          // for の終わり

        System.out.println("近似値 = " + sum);  // 結果の出力

        System.out.println("本当の値 = " +
               ((5.0*5.0/2 - 2*Math.cos(5.0))-(0.0*0.0/2- 2*Math.cos(0.0))));

    } // main メソッドの終わり

    public static double f (double x) { // メソッド f(x) の定義

        return x + 2*Math.sin(x);

    }

} // daikei2 クラスの終わり
```

ファイル **3.1:** daikei2.java