（2008-05-20 版）

Java によるプログラミング入門 (6)

1　Lesson 4: API の利用

本章では Java の API の使用例として乱数と数学関数の使い方を示します．解説はしませんが，比較的容易に取り組める割には有用で面白いので，おうちなどでやってみてください．

1.1　タスク 4

タンクに流入する流量が以下のような場合のシミュレーションを考えてみましょう：

- ランダムに変化する場合，あるいは
- sin など三角関数に従って変化する場合

このためには乱数や三角関数などが必要です．以下，Java で標準の機能 [1] として提供されているものを見てゆきます．

[1] こういった機能を提供するものを，クラスライブラリとか API とかいいます

1.2　API を使う — 数学関数と乱数

これまでの例では四則演算だけを用いて来ましたが，数学関数や乱数などを使うには，数学関数クラス Math と乱数発生クラス Random を使う必要があります．

1.3 乱数の使い方

Random クラスを用いると，擬似的に乱数のように見える数列を作り出すことができます．その使い方は

- Random クラスにアクセスするためにはプログラムの先頭で

  ```
  import java.util.Random;
  ```

  として「util というパッケージの Random というクラスを使うぞ」と指定しておきます．

- Random クラスのオブジェクト（インスタンス）を作ります[2]．次の例の最初のものは乱数系列をコンピュータの時計を用いて初期化します．2 行目は初期化のためのパラメータ (この場合 1）を与えて乱数系列を生成します．前者は実行の度に結果が異なります．後者は毎回同じ系列が得られます[3]．

  ```
  Random series1 = new Random();
  Random series2 = new Random(1);
  ```

- 乱数発生装置のメソッドを呼び出す形で乱数値を得ます．nextInt なら整数値，nextDouble なら実数値を得られます．[4]

  ```
  int number1 = myRandomSeries1.nextInt(6);
  int number2 = myRandomSeries2.nextInt(6);
  ```

[2] 乱数発生装置を手に入れる，というイメージです．

[3] ゲームなどを作るには前者が好ましく，数値実験などで再現性が求められるものは後者が好ましい．

[4] 詳しくは表 1 参照

以下の例を打ち込み，“何度か” 実行してみて 2 つの系列によって実行毎に結果が異なるものと，毎回同じ結果が得られるものがあることを確認してください．

```
 1: import java.util.Random;
 2: public class SampleOfRandom {
 3:   public static void main(String args[]) {
 4:     Random series1 = new Random();
 5:     Random series2 = new Random(1);
 6:     System.out.println("Series 1, Series 2");
 7:     for (int i=0; i<10; i++) {
 8:       int number1 = series1.nextInt(6);
 9:       int number2 = series2.nextInt(6);
10:       System.out.println(number1+","+number2);
11:     }
12:   }
13: }
```

- 1 行目で util パッケージの Random クラスを使うことを宣言します．
- 4 行目では毎回異なる乱数系列を，5 行目では毎回同じ乱数系列を作ります．
- 7〜11 行目は for ループです．10 回繰り返されます．
- 8 行目では毎回異なるほうの乱数系列から，0 から 5 の整数を生成します．
- 9 行目では毎回同じほうの乱数系列から，0 から 5 の整数を生成します．

表 1　Java の乱数クラス (Random)

返り値の型	メソッド名	説明
	Random()	コンストラクタ．系列の初期値はシステムの時計で決定.
	Random(long *seed*)	コンストラクタ．系列を決定する種 *seed* を与える.
boolean	nextBoolean()	次のブーリアン乱数
void	nextBytes(byte[] *bytes*)	次のランダムバイト列．*bytes*[] に値が返る
double	nextDouble()	次の一様乱数．範囲は 0 から 1.
float	nextFloat()	次の一様乱数．範囲は 0 から 1.
double	nextGaussian()	次のガウス乱数．平均は 0 ，標準偏差は 1.
int	nextInt()	次の整数乱数.
int	nextInt(int n)	次の整数乱数．0 以上 n 未満の一様乱数．JDK 1.2 以降で使える.
long	nextLong()	次のロング型整数乱数.
void	setSeed(long *seed*)	乱数の初期化の種に *seed* を設定する.

1.4 付録．数学関数を使う

数学関数を使うには Math クラスを使います．

- 先の Random クラスと異なり，数学関数の定義は利用する場合によらず定まっていますので，インスタンスを作らずにクラス Math のメソッドを読んで使います[5]．
- 良く使うので import 命令の不要なパッケージとなっています．

[5]static 宣言されているメソッドで Java の用語ではクラスメソッドと呼びます

次の例は三角関数表を CSV 形式で作成するものです．実行結果を表計算ソフトで読み込んでグラフを作成してみてください．

```
1: public class SampleMath {
2:   public static void main(String args[]) {
3:     System.out.println("angle in degree, cos, sin");
4:     for (int angleInDegree=0; angleInDegree<360; angleInDegree+=10) {
5:         double angleInRadian = Math.PI * angleInDegree/180;
6:       System.out.println(angleInDegree + ","
7:         + "," + Math.cos(angleInRadian) +"," + Math.sin(angleInRadian));
8:     }
9:   }
10: }
```

表 2　Java の数学関数クラス (Math)

とくに断りがなければ引数，戻り値とも double (倍精度実数) 型.

定数	
Math.E	自然対数の底
Math.PI	円周率
絶対値関数，整数化関数	
Math.abs(a)	a の絶対値．a の型 (double, float, int, long) に応じて，同じ型で返す.
Math.ceil(double a)	a より大きい最小の整数 (切り上げ)
Math.floor(double a)	a より小さい最大の整数 (切り捨て)
Math.round(double a)	a の四捨五入．戻り値は int あるいは long 型
Math.rint(double a)	a の四捨五入．戻り値は double 型
最大値関数，最小値関数	
Math.min(a, b)	a と b の最小値．引数，戻り値は double, float, int, long などの型
Math.max(a, b)	a と b の最大値.

表 3　Java の数学関数クラス (Math), つづき

三角関数	
Math.cos(double a)	正弦関数
Math.sin(double a)	余弦関数
Math.tan(double a)	正接関数
Math.acos(double a)	逆余弦関数
Math.asin(double a)	逆正弦関数
Math.atan(double a)	逆正接関数
Math.atan2 (double a, double b)	座標 (a, b) を局座標表示をしたときの角度.
Math.toDegrees(double a)	度への変換
Math.toRadians(double a)	ラジアンへの変換
巾乗関数，巾乗根関数，指数関数，対数関数	
Math.sqrt(double a)	平方根
Math.exp(double a)	指数関数
Math.log(double a)	自然対数関数
Math.pow(double a,fouble b)	巾乗 a^b

1.5 付録．不動小数点数の落とし穴

Java で使う数値には不動小数点型 (double と float があります．精度の高い double 型を常用すればいいでしょう）と整数型 (精度によって byte, short, int, long があります，通常 int 型を用いますが，金額など桁数の多いものを扱う場合には long 型を使う必要があります）があります．数値の計算には幾つか落とし穴があります．

- 整数型どうしの割り算では商は切り捨てられます．これは結果を浮動小数点型に代入する場合でも変わりません．例えば先の三角関数の例でラジアンへの変換を次のように書くと意図した計算にはなりません．

```
angleInRadian = Math.PI*(angleInDegree/180);
```

- 不動小数点数は内部では 2 進数で表現されているため，10 進数の小数 (例えば 0.1) は正確には表現できません [6]．従って次のような表現は直感的な理解と異なるかもしれません．

```
for(double a = 0.0; a < 1.0; a += 0.1) {
  System.out.println(a);
}
```

ちなみに，不動小数点数は常に近似誤差を含んでいると考える必要があります．等式の条件判断 "==" やその否定の "!=" などは極めて危険であり不等式を用いるべきです．

[6] 10 進数で 1/3 が無限小数になるのと同じこと