第3章　Javaの基本制御構造

Java のその他の制御構造の構文（**if** 文、**for** 文、**while** 文）は基本的には C と全く同じである。制御構造の復習を兼ねて、これらの制御構造を使った例題を取り上げる。

3.1　if文

Java の if 文は C 言語と同じ書き方である。

if（条件式）文 $_1$
if（条件式）文 $_1$ **else 文** $_2$

条件式が成り立てば文 $_1$ を実行する。1 番めの形式は条件式が成り立たなければ何もしない 。2 番めの形式は文 $_2$ を実行する。 文 $_1$, 文 $_2$ は、当然ブロック（“{” と “}” で括った文の並び）でも良い。

ここで、条件式の型は _____ 型である。既に紹介した `Graphics` クラスの `draw3DRect` や `fill3DRect` の引数としても用いられていた。 C 言語と異なり整数型（`int` 型）とは区別されている。このため（C 言語では OK だった） `while (1) ...` のような文はエラーとなる。

問 3.1.1 *`int`* 型と *`boolean`* 型を区別することの長短をまとめよ。

.....
.....
.....
.....
.....

条件判断文としてはこの他に **switch〜case 文**もあるが、C 言語と同じなので、ここでは説明を割愛する。

例題 3.1.2
`Calendar` クラスを使って挨拶を行なう。

ファイル CalendarTest.java

```
import javax.swing.*;
import java.awt.*;
import java.util.*;

public class CalendarTest extends JApplet {
  @Override
```

```
  public void paint(Graphics g) {
    Calendar now = Calendar.getInstance();
    int day  = now.get(Calendar.DAY_OF_WEEK);
    int hour = now.get(Calendar.HOUR_OF_DAY);
    int min  = now.get(Calendar.MINUTE);
    if (day==Calendar.SUNDAY) {
      g.setColor(Color.RED);
    }
    if (hour<12) {
      g.drawString("おはようございます。", 30, 25);
    } else if (hour<18) {
      g.drawString("こんにちは。", 30, 25);
    } else {
      g.drawString("こんばんは。", 30, 25);
    }
    g.drawString("ただいま␣" + hour + "時␣" + min + "分です。", 30, 50);
  }
}
```

java.util.Calendar クラスの使用方法については、API ドキュメントを参照すること。このプログラムでは、日曜日だけ色を赤色に変更し、時間に応じて挨拶文を変えている

3.2　文字列（String）に関する演算子とメソッド

Java では、________を用いて ____（あるいは、String 型と int 型のオブジェクトを String 型に変換したものを連接する）ことができる。

例:

```
System.out.println("2+2 は" + (2+2));
System.out.println("2+3 は" + (2+3) + "です。");
```

一方、JDK 5.0 からは C 言語のような書式指定を行う printf や sprintf メソッドに相当するメソッドも使用できる。上の drawString の場合、String.format というクラスメソッドを使って、次のように書くこともできる。

```
g.drawString(String.format("ただいま %d 時 %d 分です。", hour, min), 30, 50);
```

詳細: この printf のようなメソッドは利用するのは簡単だが、総称クラス（Generics）・オートボクシング（Autoboxing）・可変個の引数（Varargs）など、いろいろな考え方が組合せられている。このうち総称クラスについては後述する。

可変個の引数を持つメソッドは API のドキュメントでは、

```
public static String format(String format, Object... args)
```

のように... を使って表されている。（この format メソッドは java.lang.String クラスのクラスメソッドである。）

3.3 for 文, while 文

while (条件式 $_1$) 文 $_1$
for (式 $_1$; 式 $_2$; 式 $_3$) 文 $_1$
for (型 変数名 : 式) 文 $_1$

while 文は条件式 $_1$ が成り立つ間、文 $_1$ の実行を繰り返す。
1 つめの形式の **for** 文はループに入る前に、まず式 $_1$ を評価する。式 $_2$ が成り立つ間、文 $_1$、式 $_3$ の実行を繰り返す。2 つめの形式の **for** 文は JDK5.0 で導入されたものである。**for-each** 文と呼ばれることもある。（ただし、each というキーワードを使うわけではないので注意する。）この場合、式は直感的には何かの集まりを表すデータ型（配列など — 正確には配列またはインタフェース `Iterable` を実装するクラス）でなければならない。コロン（:）の前で宣言された変数に、この列の要素が順に代入され、文の実行が繰り返される。この形式の **for** 文の使用例はもう少し後で紹介する。

繰り返し文としてはこの他に **do〜while 文**もあるが、C 言語と同じなのでここでは説明を割愛する。

例題 3.3.1 正多角形の描画
整数 n をパラメータとして受け取り、正 n 角形を描画する。

ファイル N_gon.java

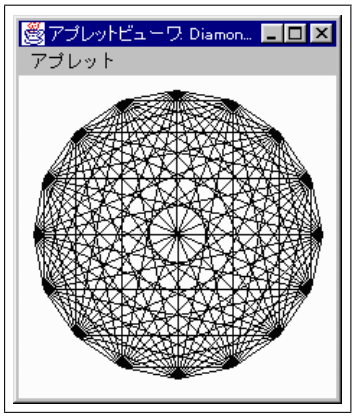
```
import javax.swing.*;
import java.awt.*;
import static java.lang.Math.*;

public class N_gon extends JApplet {
  @Override
  public void paint(Graphics g) {
    super.paint(g);

    int np = 7;
    int sc = 100;

    int i;
    double theta1, theta2;
    for (i=0; i<np; i++) {
      // 単位 ラジアン
      theta1 = PI*2*i/np;      // 360*i/n 度
      theta2 = PI*2*(i+1)/np;  // 360*(i+1)/n 度
      g.drawLine((int)(sc*(1.1+cos(theta1))), (int)(sc*(1.1+sin(theta1))),
                 (int)(sc*(1.1+cos(theta2))), (int)(sc*(1.1+sin(theta2))));
    }
  }
}
```

`Math.PI` は 円周率 π （=3.1415 . . . ）、`Math.sin`, `Math.cos` は正弦、余弦関数である。これらはクラスフィールド、クラスメソッドである。

問 3.3.2 *sin, cos* などの数学関数のグラフを描くアプレットを書け。
参考: (JDKDIR)/docs/ja/api/java.lang.Math.html

問 3.3.3 正 *n* 角形のすべての頂点を結んでできる図形（ダイアモンドパターン）を描画するアプレットを書け。

問 3.3.4 色のグラデーション（2 次元 — 縦方向と横方向が別の色に変わる）を作成するアプレットを書け。



ダイアモンドパターン

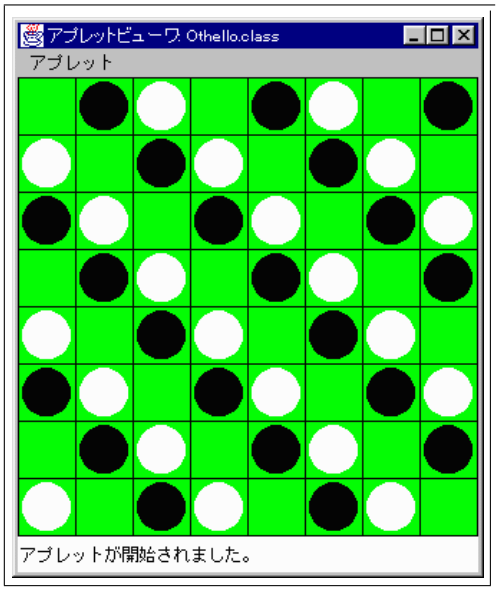
2 次元のグラデーション

（参考）1 次元のグラデーション

（参考）　1 次元のグラデーション

ファイル Gradation1.java

```
import javax.swing.*;
import java.awt.*;

public class Gradation1 extends JApplet {
  @Override
  public void paint(Graphics g) {
    super.paint(g);
    int scale = 4;
    int i;

    for (i=0; i<64; i++) {
      g.setColor(new Color(i*4, 0, 255-i*4));
      g.fillRect(i*scale, 0, scale, scale*10);
    }
  }
}
```



Othello.java

例題 3.3.5 グラフの描画
整数のデータを与え、そのデータの棒グラフを描く。

ファイル Graph.java

```
import javax.swing.*;
import java.awt.*;

public class Graph extends JApplet {
  @Override
  public void paint(Graphics g) {
    super.paint(g);
    int[] is = {10, 4, 6, 2, 9, 1};
    Color[] cs = {Color.RED, Color.BLUE};
    int scale = 15;
    int i, n = is.length;              // 配列の大きさ

    for (i=0; i<n; i++) {
      g.setColor(cs[i%2]);           // g.fillRect(0, i*scale, is[i]*scale, scale);
    }
  }
}
```

配列オブジェクトの _____というフィールド (?) によって配列の大きさ（要素数）を知ることができる。これも C 言語と異なる点である。**for** 文の中のブロックは変数 i が 0〜n-1 まで変化する間、繰り返される。

3.4 多次元配列

例題 3.4.1 *int* 型の *8*× *8* の大きさの配列の配列を調べて、*1* なら白丸、*2* ならば黒丸を画面上の対応する位置に描画する。

ファイル Othello.java

```
import javax.swing.*;
import java.awt.*;

public class Othello extends JApplet {
  @Override
  public void paint(Graphics g) {
    super.paint(g);
    int scale = 40;
    int space = 3;
    int[][] state = {{0,1,2,0,1,2,0,1}, {2,0,1,2,0,1,2,0}, {1,2,0,1,2,0,1,2},
                     {0,1,2,0,1,2,0,1}, {2,0,1,2,0,1,2,0}, {1,2,0,1,2,0,1,2},
                     {0,1,2,0,1,2,0,1}, {2,0,1,2,0,1,2,0}};
    int i,j;

    for (i=0; i<8; i++) {
```

```
      for (j=0; j<8; j++) {
        g.setColor(Color.GREEN);
        g.fillRect(i*scale, j*scale, scale, scale);
        g.setColor(Color.BLACK);
        g.drawRect(i*scale, j*scale, scale, scale);
        if (state[i][j]==1) {
          g.setColor(Color.WHITE);
          g.fillOval(i*scale+space, j*scale+space, scale-space*2, scale-space*2);

        } else if (state[i][j]==2) {
          g.setColor(Color.BLACK);
          g.fillOval(i*scale+space, j*scale+space, scale-space*2, scale-space*2);
        }
      }
    }
  }
}
```

2 次元配列（配列の配列）を宣言するには、上のように [] を 2 つ重ねる（3 次元以上も同様）。C 言語の場合のように要素数を宣言する必要はない。（ただし、C 言語でも最初の次元の要素数は省略することができる。）state は配列の配列で、例えば、state[0][1] は、0 番めの配列 {0,1,2,0,1,2,0,1} の 1 番めの数だから _ である。つまりこの位置 （0 列めの 1 行め）には白丸が描画される。
注意: なお、Java の 2 次元配列と C の 2 次元配列はメモリ上の配置の仕方が異なる。（もっとも Java でメモリ上の配置を意識する必要はほとんどない。）このため Java では C ではメモリの効率が悪い次のような 2 次元配列（異なるサイズの配列が混在している）

```
int[][] xss = {{1}, {1,2}, {1,2,3}};
```

も使用できる。

キーワード　if 文, if〜else 文, while 文, for 文, for-each 文, 配列, length メソッド, static, Math クラス, 多次元配列,