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。

Q 3.1.1 次のプログラムの断片の出力を書け。

1.
```
int n = 2;
if (n <= 1) {
    System.out.printf("A");
}
if (n <= 2) {
    System.out.printf("B");
}
if (n <= 3) {
    System.out.printf("C");
}
```

答: ___

2.
```
int n = 2;
if (n <= 1) {
    System.out.printf("A");
} else if (n <= 2) {
    System.out.printf("B");
} else if (n <= 3) {
    System.out.printf("C");
}
```

答: _

ここで、条件式の型は boolean 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 3.1.2 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 3.1.3
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) {
    super.paint(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、＿＿＿＿＿＿（空欄 3.2.1）を用いて String 型と String 型のオブジェクトを連接する（あるいは、String 型と String 以外の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。）

Q 3.2.1 次の中で “1+1 は 2 です。” と出力する式はどれか？

1. ☐ `System.out.println("1+1 は +(1+1)+ です。")`
2. ☐ `System.out.println("1+1 は " + (1+1) + " です。")`
3. ☐ `System.out.println("1+1 は " + 1+1 + " です。")`
4. ☐ `System.out.println("1+1 は "(1+1)" です。")`
5. ☐ `System.out.printf("1+1 は %d です。%n", 1+1)`
6. ☐ `System.out.println(String.format("1+1 は %d です。", 1+1))`

3.3　for 文, while 文, do〜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 $_1$
do 文 $_1$ while (条件式 $_1$);

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。

Q 3.3.1 次のプログラムの断片の出力を書け。

```
    int i;
    for (i=0; i<4; i++) {
        System.out.printf("*%d", i);
    }
    System.out.printf("|%d", i);
```

答: _____

例題 3.3.2 正多角形の描画
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。

例題 3.3.3 二次関数のグラフの描画
数学関数のグラフを描くには、定義域を細かい区間に区切り、短い線分をつなぎ合わせれば良い。

ファイル Parabola.java

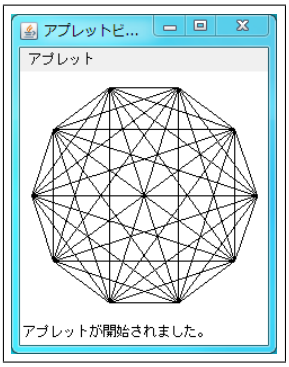
```
import java.awt.*;
import javax.swing.J*;

public class Parabola extends JApplet {
  @Override
  public void paint(Graphics g) {
    super.paint(g);
    double a = -0.0025, b = 1, c = 0;
    for (int x0=0; x0<200; x0+=10) {
      double y0 = a * x0 * x0 + b * x0 + c;
      int x1 = x0 + 10;
      double y1 = a * x1 * x1 + b * x1 + c;
      g.drawLine(x0, (int)y0, x1, (int)y1);
    }
  }
}
```

問 3.3.4 *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 3.3.5 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 3.3.6 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);
    }
  }
}
```

例題 3.3.7 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%cs.length]);     // % は余りを求める演算子
      g.fillRect(0, i*scale, is[i]*scale, scale);
    }
  }
}
```

配列オブジェクトの _____（空欄 3.3.1）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。

Q 3.3.8 ds という double 型の配列があったとき、ds の要素の平均値を求めるメソッドを定義する。2 箇所の空欄を同じ内容で埋めて定義を完成せよ。

```
    double average(double[] ds) {
        int n = 0
        int i;
        for (i = 0; i < [          ]; i++)  {
            n += ds[i];
        }
        return (double)n/[          ];
    }
```

C 言語では、配列の範囲外をアクセスしても通常エラーにならないが、Java では ArrayIndexOutOfBoundsException という例外が発生する。例外については次章で詳しく説明する。

Q 3.3.9 次のプログラムを実行して、エラーメッセージを確認せよ。
ファイル ArrayIndexOutOfBoundsExceptionTest.java

```
public class ArrayIndexOutOfBoundsExceptionTest {
  public static void main(String args[]) {
    int[] a = {1, 2, 3};
    for (int i = 0; i <= a.length; i++) {
      System.out.println(a[i]);
    }
  }
}
```

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

また、次の C 版も実行してみよ。
ファイル ArrayIndexOutOfBoundsExceptionTest.c

```
#include <stdio.h>

int main(void) {
    int a[] = {1, 2, 3};
    int i;
    for (i = 0; i <= 3; i++) {
        printf("%d\n", a[i]);
    }

    return 0;
}
```

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

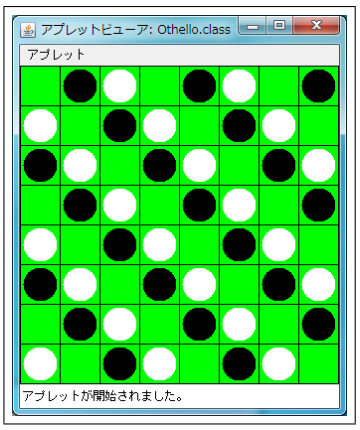
```
import javax.swing.*;
import java.awt.*;

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);
        }
      }
    }
  }
}
```



Othello.java

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 言語と異なり、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（空欄 3.4.1）が入っている部分である。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}, {2, 3}, {4, 5, 6}};
```

も使用できる。（C だと

```
int xss[][3] = {{1}, {2, 3}, {4, 5, 6}};
```

と宣言する必要がある。）

C 言語の場合	Java の場合

Q 3.4.2 `int[][] xss = {{1}, {2, 3}, {4, 5, 6}};` のとき次の式の値を答えよ。範囲外でエラーになるときは × を書け。

1. `xss[1][1]` 答: ___
2. `xss[0][1]` 答: ___
3. `xss[2][1]` 答: ___
4. `xss.length` 答: ___
5. `xss[1].length` 答: ___

キーワード　`if` 文, `if`〜`else` 文, `while` 文, `for` 文, `for-each` 文, 配列, `length` フィールド , `ArrayIndexOutOfBounds` 例外, `static`, `Math` クラス, 多次元配列,