メソッド2

前回は簡単なメソッドを用いたプログラムを利用して、メソッドの宣言と呼び出しについて学習した。この場合の引数と返却値は int や doulbe、void といった基本的な変数型を用いた。今回は引数や返却値によって参照型である配列を扱うメソッドについて学習していく。また、メソッドの多重定義について学ぶ。

引数で配列を扱うメソッド

以下のプログラムは引数に配列を用いるメソッドを利用したプログラムである。main メソッドで生成した配列について、参照型である配列名 a を引数として sum メソッドを呼び出す。sum メソッド本体では配列の各要素の合計を求め、その値を返却している。

この場合、配列名 a には配列本体の参照先が格納されているため、sum メソッドには配列本体を引数で渡しているわけではないことに注意すること。

プログラム例１

```
public class Prog09_01
{
        static int sum(int[] x)
        {
                int y = 0;
                for(int i = 0; i < x.length; i++) y += x[i];
                return y;
        }

        public static void main(String[] args)
        {
                int[] a = {1, 2, 3, 4, 5};
                System.out.println("配列 a の合計は" + sum(a) + "です。");
        }
}
```

実行結果

```
配列 a の合計は 15 です。
```

返却値で配列を扱うメソッド

以下のプログラムは返却値で配列を扱うメソッドを利用したプログラムである。main メソッドから整数値を引数として make_array メソッドを呼び出し、配列本体の生成と代入を行っている。返却値には「配列本体への参照」である x を用い、main メソッド内で宣言した「配列本体への参照」である a に格納している。main メソッドでは配列本体の生成は行っていないことに注意すること。

プログラム例２

```
public class Prog09_02
{
        static int[] make_array(int n)
        {
                int[] x = new int[n];
                for(int i = 0; i < n; i++) x[i] = i;
                return x;
        }

        public static void main(String[] args)
        {
                int[] a;
```

```
                int element = 3;

                a = make_array(element);
                for(int i = 0; i < element; i++)
                        System.out.println("a[" + i + "] = " + a[i]);
        }
}
```

実行結果

```
a[0] = 0
a[1] = 1
a[2] = 2
```

多次元配列の受け渡し

ここまでにメソッドを利用した1次元配列の受け渡しの例を示した。ここでは多次元配列を引数に用いた例を示す。以下のプログラムはメソッドによって行列の和を求めるプログラムの例である。行列の和を求めるメソッドと行列を表示するメソッドを用意した。

プログラム例３

```
public class Prog09_03
{
        static void add_matrix(int[][] x, int[][] y, int[][] z)
        {
                for(int i = 0; i < x.length; i++)
                        for(int j = 0; j < x[i].length; j++)
                                z[i][j] = x[i][j] + y[i][j];
        }

        static void print_matirx(int[][] x)
        {
                for(int i = 0; i < x.length; i++)
                {
                        for(int j = 0; j < x[i].length; j++)
                                System.out.print(x[i][j] + " ");
                        System.out.println();
                }
        }

        public static void main(String[] args)
        {
                int[][] a = {{1, 2, 3}, {4, 5, 6}};
                int[][] b = {{7, 8, 9}, {7, 3, 1}};
                int[][] c = new int[2][3];

                add_matrix(a, b, c);
                System.out.println("行列 a"); print_matirx(a);
                System.out.println("行列 b"); print_matirx(b);
                System.out.println("行列 c"); print_matirx(c);
        }
}
```

実行結果

```
行列 a
```

```
1 2 3
4 5 6
行列 b
7 8 9
7 3 1
行列 c
8 10 12
11 8 7
```

参考として、「int[][] x = new int[2][3]」として生成した場合の 2 次元配列の内部構造を示しておく。

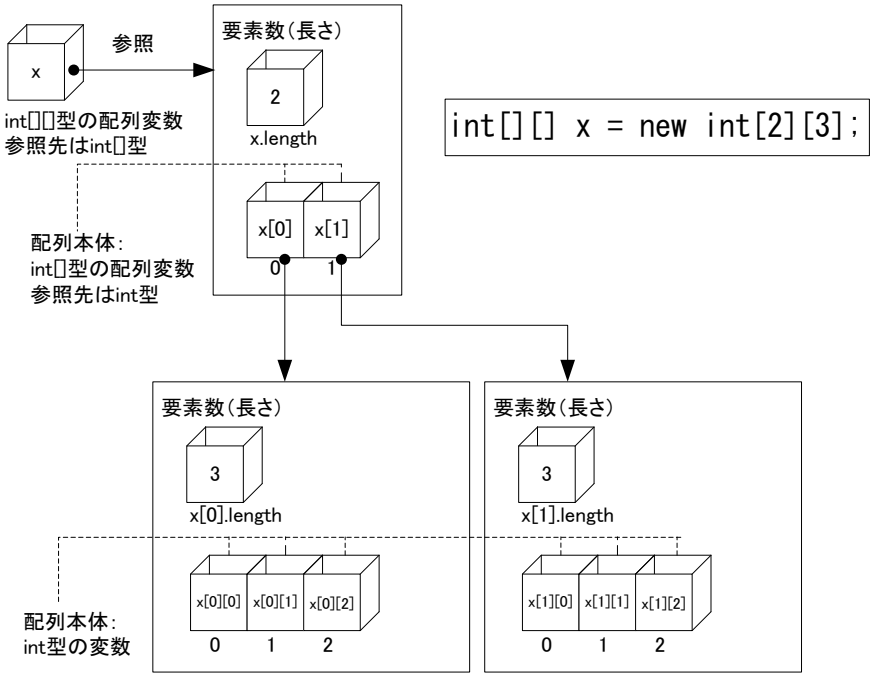


図 1　2 次元配列の内部構造

多重定義（オーバーロード）

メソッドによっては引数の型や個数が違うだけで、同じような処理を行うものが複数必要になる場合がある。例えば、2 つの値を引数として、その平均値を求めて返却するメソッドが欲しいとする。この値が int 型のものと、double 型のものが欲しい場合、異なるメソッド名を与えるとプログラムが煩雑になる。

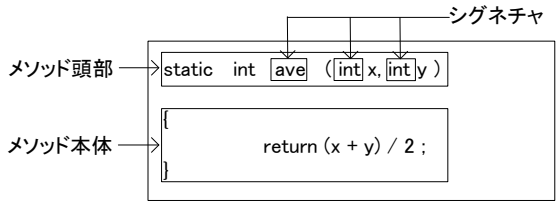
Java では同じクラス内に同じ名前のメソッドが複数存在することが許されており、同一クラス内で宣言することを「メソッドを多重定義する」という。ただし制約があり、「同じシグネチャのメソッドは多重定義できない」ということになっている。

「シグネチャ」とは「メソッド名・仮引数の型・仮引数の個数の組み合わせ」のことである。メソッドの宣言の書式は以下の通りだった。下線部の組み合わせがシグネチャであるので、仮引数の型と個数が異なる場合に多重定義をすることができる。

書式

static　返却型　メソッド名（仮引数の型　引数名, …, 仮引数の型 引数名）
{
　　メソッド本体
}

使用例



シグネチャに含まれるもの
○メソッド名
○仮引数の型と個数の組み合わせ
×返却型は含まれない
×仮引数の変数名

多重定義ができない場合を示していく。以下のプログラムの場合、異なる部分は返却型になるので、この場合にはコンパイルエラーになる。

```
static int ave(int x, int y)
{
        return (x + y) / 2;
}
```

```
static double ave(int x, int y)
{
        return (double)(x + y) / 2;
}
```

シグネチャに返却型が含まれない理由は呼び出す時を考えるとわかる。呼び出すには以下のような形になる。

```
        ave(3, 4);
```

この場合、コンパイラはどちらのメソッドを呼び出せばよいか区別できない。また、以下のように、仮引数の名前が異なる場合も同様の理由でエラーとなる。

```
static int ave(int x, int y)
{
        return (x + y) / 2;
}
```

```
static int ave(int a, int b)
{
        return (a + b) / 2;
}
```

最後に解説で用いた多重定義のプログラムについて、main 関数を付けたものを示しておく。

プログラム例４

```
public class Prog09_04
{
        static int ave(int x, int y)          // 平均を整数で求める
        {
                return (x + y) / 2;
        }

        static double ave(double x, double y) // 平均を実数で求める
        {
                return (x + y) / 2.0;
        }
```

```
        public static void main(String[] args)
        {
                int a = 3, b = 6;
                double c = 3.0, d = 6.0;
                System.out.println("整数での平均:" + ave(a, b));
                System.out.println("実数での平均:" + ave(c, d));
        }
}
```

実行結果

```
整数での平均:4
実数での平均:4.5
```

演習

プログラム例１～４を作成して、出力を確認しなさい。

課題５の続き　以下のプログラムは main メソッドを含めて動作する状態にすること。
kadai5_5　配列の要素の最小値を求めるメソッドを作成しなさい。
kadai5_6　配列 a と配列 b の全要素を交換するメソッドを作成しなさい。要素数が等しくない場合は小さい方の要素数分の要素を交換する。
kadai5_7　プログラム例３は３つの配列の行列数が等しいことを前提としている。3 つの配列の要素数が等しければ加算を行って true を返し、等しくなければ加算を行わず、false を返すメソッドに書き換えなさい。
kadai5_8　２つの int 型整数 a,b の最小値、３つの int 型整数 a,b,c の最小値、配列 a の要素の最小値を求める多重定義されたメソッド群を作成しなさい。