目次

★6　配列 (p.64)

★6.1　Java の配列はオブジェクトである

★6.1.1　配列の作り方，使い方

まずは，C 言語と Java で整数の配列を扱うプログラムの例を見てみよう．

C 言語で int の配列

```
 1  int sum, i;
 2  int x[10];
 3
 4  sum = 0;
 5  for(i = 0; i < 10; i++){
 6    x[i] = i * i;
 7    printf("%d ", x[i]);
 8    sum += x[i];
 9  }
10  printf("\nsum = %d\n", sum);
```

Java で int の配列

```
 1  int[] x = new int[10];
 2
 3  int sum = 0;
 4  for(int i = 0; i < 10; i++){
 5    x[i] = i * i;
 6    System.out.print(x[i] + " ");
 7    sum += x[i];
 8  }
 9  System.out.println("");
10  System.out.println("sum = " + sum);
```

C 言語との最大の違いは，**Java の配列はオブジェクトである**所にある．

配列オブジェクトの生成の仕方: `new` 構成要素の型 [配列の大きさを表す式]

int の配列の生成例

```
int[] a;

a = new int[3];

int[] b = new int[5];
```

Turtle の配列の生成例

```
Turtle[] m = new Turtle[6];

int n = 100;

Turtle[] mm = new Turtle[n*n];
```

注意: 上記の `int[] a` や `Turtle[] m` などは，`int a[]` や `Turtle m[]` のように書いてもよい．

★ 6.1.2 インスタンス変数 length

配列はオブジェクトであり，length という int 型のインスタンス変数をもっている．配列の大きさが自動的に格納される．これを使うと，上記の 4 行目を

```
for(int i = 0; i < x.length; i++){
```

と書きかえられる．元のプログラムでは，配列の大きさを変更するには 2 カ所を書きかえないといけなかったが，このようにすれば変更は 1 カ所で済み，バグが混入しにくくなる．

Q1. 以下のソースを次のように書きかえなさい．

1. i 番目のかめさんは i+3 角形を描くようにする
2. 配列オブジェクトのインスタンス変数 length を使う（かめの匹数は 1 箇所にしか書かないようにする）
3. かめさんを 6 匹にする

T61.java（import 文は省略）

```
 3  public class T61 {
 4      public static void main(String[] args){
 5          TurtleFrame f = new TurtleFrame(600,300);
 6          HTurtle[] hm = new HTurtle[10];  // かめ 10 匹分の配列

 7          for(int i = 0 ; i < 10; i++){


 8              hm[i] = new HTurtle(i * 50 + 25,150,0); // 個々のかめインスタンスの生成
 9              f.add(hm[i]);
10          }
11          for(int i = 0; i < 10; i++){

12              hm[i].polygon(6, 10);    // hm[i] に代入されてる HTurtle へのメソッド呼び出し
13          }
14      }
15  }
```

★ 6.1.3 添字が配列の範囲をはずれると

C の場合，配列の添字が配列の範囲をはずれていても (☆ 1) 実行されてしまい，エラーが出ないこともある．また，エラーになるとしても Segmentation Fault や Bus Error といったエラーで，添字が原因かどうかは一見分からない．しかし Java の場合，添字が範囲をはずれていないかどうかは実行時にチェックされ，範囲をはずれていると以下のように「**例外**」が発生して実行が停止する．出力された例外の種類 java.lang.ArrayIndexOutOfBoundsException から，添字が範囲をはずれたことが原因であることがわかる (☆ 2)．

前頁のソースの 4 行目を i < 100 にしてみると

```
$ java G08Array
0 1 4 9 16 25 36 49 64 81 Exception in thread "main" java.lang.
ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10
  at G08Array.main(G08Array.java:7)
```

例外というのは，プログラムの実行時にエラーとなるような例外的な事象（配列の添字が範囲をはずれた，0 で除算した，ファイルが見つからない…）のことである．Java では例外を処理するプログラムを作りやすくなっているが，この授業では触れない（詳細は第 8 章，(☆ 3)）．

☆ 1) 例えば上記の 5 行目を i < 100 としたり，x[-1] にアクセスしたり

例外: Exception

☆ 2) array: 配列．index: 添え字．out of bounds: 範囲をはずれている．メッセージをよく読むと，添字の値 10 は length 10 に対してまずいということ，およびこの例外が発生した場所がソースの 7 行目であることもわかる．

☆ 3) ひとに使ってもらうプログラムを開発する際には，例外の処理をちゃんとすることがとても重要になる．

★ 6.2 配列の初期値 (p.67)

C 言語の場合，前の頁で説明したように初期値を与えずに配列をつくると，配列の中身は不定（どんな値が入っているか分からない）である．一方，**Java の場合，初期値を与えずに配列を生成すると，あらかじめ決められた値が自動的に代入される**．例えば，int 型の配列なら 0 が代入される (☆ 4)．また，以下のように自分で初期値を与えることもできる．この例では，配列の大きさは初期値の数から自動で決めさせている．

☆ 4) double 型など他の数を表す型も同様に 0 となる．boolean 型なら false．参照型なら null（何も指していないことを表す）．

配列に初期値を与える (C 言語)

```
int m = 5;
int mon[] = {0,31,28,31,30,31,30,31,
             31,30,31,30,31};
printf("%d 月は%d 日まであるよ\n", m, mon[m]);
```

配列に初期値を与える (Java)

```
int m = 5;
int[] mon = {0,31,28,31,30,31,30,
             31,31,30,31,30,31};
System.out.println(m + "月は" + mon[m]
                   + "日まであるよ");
```

Q2. int の配列を次のように与えると右のような出力をするプログラムを書きなさい．

```
int[] a = { 111, 32, 99, -100, 77 };
```

実行結果例

```
111 32 99 -100 77
和 = 219
最小値のインデックス = 3
最小値 = -100
```

教科書 p.69「可変長引数」,「拡張された for 文」はスキップします．

★ 6.3 配列の配列 (p.70)

★ 6.3.1 配列の配列を作ろう

Java では，多次元配列は「配列の配列」として作ることになる．以下の (3) が示すように，長さの異なる配列をまとめた多次元配列を作ることもできる．

多次元配列の例 (1)

```
int[][] a = new int[3][4];

for(int i = 0; i < a.length; i++){
    for(int j = 0; j < a[i].length; j++){
        a[i][j] = 100 * (i + 1) + j + 1;
    }
}
for(int i = 0; i < a.length; i++){
    for(int j = 0; j < a[i].length; j++){
        System.out.print(a[i][j] + " ");
    }
    System.out.println("");
}
```

こんな作り方もあり (2)

```
int[][] a = new int[3][];

for(int i = 0; i < a.length; i++){

    a[i] = new int[4];

    for(int j = 0; a[i].length; ...
      （以下左と同じ）
```

こんなんもあり (3)

```
int[][] a = new int[3][];

for(int i = 0; i < a.length; i++){
    a[i] = new int[i+2];
    for(int j = 0; j < a[i].length; j++){
        a[i][j] = 100 * (i + 1) + j + 1;
    }
}
      （以下上と同じ）
```

Q3. 上記の (1),(3) の実行結果を答えなさい．

★ 6.3.2 配列の配列の初期化

一次元の配列と同様に，多次元配列も初期化することができる．

多次元配列の初期化の例

```
int[][] a = {{2,5,4}, {7,2,1}};  // 2 x 3 の配列
int[][] b = {{2,5,4}, {7}};      // 要素の個数が異なる例
int[][] c = {{2,5,4}, {}, {7}};  // 大きさ 0 の配列もあり
d = new int[2];
int[][] e = {{2,5,4}, d};        // こんなんもあり
```

★ 6.4 main の引数 (p.73)

次のプログラムと実行例が示すように，main() メソッドの引数に指定した String の配列には，java コマンドでプログラムを起動した際の**コマンドライン引数**が格納されている．

Args01.java

```
1  public class Args01{
2      public static void main(String[] args){
3          System.out.println("引数は" + args.length
             + "個やね");
4          for(int i = 0; i < args.length; i++){
5              System.out.println(args[i]);
6          }
7      }
8  }
```

Args01 の実行結果

```
$ java Args01  1  2  SHAAAAAAN!
引数は 3 個やね
1
2
SHAAAAAAN!
$ java Args01
引数は 0 個やね
```

次の例から分かるように，main() メソッドの引数の変数名は自分で決めればよい．ただし，hoge[0] 等は String 型であるから，int として扱いたければ 7,8 行目のように変換する必要がある（⇒）．

Args02.java

```
1  public class Args02{
2      public static void main(String[] hoge){
3          if(hoge.length != 2){
4              System.out.println("整数二つ指定してね");
5              System.exit(0);
6          }
7          int x = Integer.parseInt(hoge[0]);
8          int y = Integer.parseInt(hoge[1]);
9          System.out.println( x + y );
10     }
11 }
```

Args02 の実行結果

```
$ java Args02
整数二つ指定してね
$ java Args02 37 12
49
$ java Args02 37 hoge
Exception in thread "main"
 java.lang.NumberFormatException:
 For input string: "hoge"
(以下略)
```

⇒ ここでででてくる Integer というクラスは,「ラッパークラス」というもの．詳しくは p.81．double 型に変換したいときは Double クラスの parseDouble メソッドを用いる．