オブジェクト指向とゲームプログラミング

○　Java 入門編 － 第６回　配列　○

配列の宣言と作成

Javaの配列は、クラスのオブジェクトや基本型の値を指す(参照する)オブジェクトです。C／C++と記述は似ているものの、異なる点があります。

Javaでは、配列を２つのステップで作成します。１つめのステップは配列の宣言で、２つめのステップは配列の作成です。

配列の宣言には、２つの記述方法があります。C／C++のように、配列名の後に"[]"を記述する方法と、データ型の後に"[]"を記述する方法です。Javaでは、配列型(配列オブジェクト)として扱うので、後者の方が一般的です。どちらの方法でも、要素数を宣言と同時に指定することはできません。要素数は配列の作成時に指定します。
int型の配列arrayを宣言するには、

```
int array[];
```

または

```
int[] array;
```

となります。

次に、配列(配列オブジェクト)を作成します。Javaでは、オブジェクトの作成にnewを使いますが、配列もオブジェクトなので、newを使って作成します。基本型の配列は、newの後にデータ型を記述し、要素数を"[]"で囲って指定します。いったん作成されると要素数の変更はできません。
上で宣言したarrayを要素数10の配列として作成するには、

```
array = new int[10];
```

となります。これでint型の値を10個格納する領域がメモリに確保されます。

配列の要素にアクセスするには、配列名の後に添字を"[]"で囲って指定します。添字の範囲は先頭が0、末尾が要素数から1を引いた値になります。上のarrayでは、array[0]からarray[9]までとなります。array[1]からarray[10]でない点に注意してください。

また、上の例では、配列の初期化を行っていませんが、デフォルトでは数値型の場合は0、boolean型の場合はfalseですべての要素が初期化されます。

以下のように、宣言と作成を１行にまとめて記述することもできます。

```
int[] array = new int[10];     // int array[] = new int[10]でも可
```

配列の初期化

配列を宣言する際に、初期化を行うことができます。この場合、メモリの確保が自動的に行われ、初期値の個数に合わせた配列が作成されます。
初期化に使う値は"{}"で囲み、値は","（カンマ）で区切ります。

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

上の例では、int型５個を格納する配列が作成され、array[0]から順に、1、2、3…と初期化されます。
配列の宣言後でも、初期値を使って作成することができます。

```
int[]   array;

array = new int[]{1, 2, 3, 4, 5};
```

この場合も、int型５個を格納する配列が作成され、array[0]から順に、1、2、3…と初期化されます。

要素数の取得

Javaの配列はオブジェクトです。配列オブジェクトには、lengthというメンバがあり、配列の要素数が格納されています。このメンバは参照のみが可能で、書き換えることはできません。

Javaでは、配列の要素にアクセスする際、添字が範囲内（0から配列の要素数-1）にあるかつねに検査されます。もし、範囲外の場合は例外「IndexOutOfBoundsException」が投げられます。lengthメンバを使用すれば、安全に全要素にアクセスすることができます。

```
int[]   array = {2, 4, 8, 16, 32};

for(int i = 0; i < array.length; i++)
    System.out.println(array[i]);
```

多次元配列

添字が複数ある多次元配列を使うこともできます。多次元配列も１次元配列と同じ手順で作成し扱うことができますが、"[]"が複数並ぶ点が異なります。

```
int[][] array2;
```

こうすると、array2という２次元配列が宣言されます。３次元配列なら、「int[][][]」となります。また、"[]"は配列名の後ろにも記述できるので、

```
int array2[][];
```

や

```
int[] array2[];
```

のように記述することもできます。

配列の作成も１次元配列と同じように行います。

```
array2 = new int[10][100];
```

とすると、int型１０行×１００列の２次元配列がメモリ上に作成されます。宣言と作成を１行にまとめて記述することもできます。

```
int array2 = new int[10][100];
```

Javaの多次元配列は、「配列の配列」という扱いなので、要素数が一定でない配列(矩形でない配列)を作成することができます。ただし、左から右の順に要素数が指定されていなければなりません。たとえば、

```
int array2 = new int[5][];      // ＯＫ！
```

はエラーになりませんが、

```
int array2 = new int[][10];     // エラー！
```

はエラーになります。

上のarray2は、「行」しか作成されていないので、各行に対応する配列を作成する必要があります。

```
array2[0] = new int[1];
array2[1] = new int[2];
array2[2] = new int[3];
array2[3] = new int[4];
array2[4] = new int[5];
```

とすると、array2[0]は１列、array2[1]は２列、array2[2]は３列…という、三角形の２次元配列が作成されます。

多次元配列の要素数は、「配列名.length」が行数、「配列名[添字].length」が列数になります。たとえば上のarray2では、「array2.length」は5、「array2[0].length」は1、「array2[4].length」は5となります。

多次元配列も初期値を与えて初期化することができます。長方形の配列も要素が一定でない配列も初期化することができます。

```
int[][] int_array2 = {{1, 2, 3, 4, 5},
                      {6, 7, 8, 9, 10}};
```

とすると２×５の２次元配列が作成され、対応する値によって初期化されます。

```
int[][] int_array2 = {{1, 2},
                      {3, 4, 5, 6},
                      {7, 8, 9, 10, 11, 12, 13, 14}};
```

とすると各列の要素数が一定でない配列が作成され、対応する値によって初期化されます。

配列の代入

配列を他の配列に代入することができます。ただし、代入先の配列は宣言のみが行われ、作成されていないことが条件です。

配列は「配列の宣言」と「配列の作成」という２つのステップで作成しています。配列の宣言とは、配列名と配列の要素のデータ型を指定しているだけです。配列の作成(＝配列名とメモリの関連づけ)を行うことではじめて配列の各要素が作成されます。したがって、配列の宣言だけを行い、その配列名を使って、メモリに存在する別の配列を参照することができます。ただし、配列の代入では、各要素がコピーされるのではなく、同じメモリ領域を指しているという点に注意してください。

```
int   tbl1[] = {1, 2, 3, 4, 5};
int   tbl2[] = tbl1;
```

この場合、tbl2の内容を書き換えると、tbl1の内容も書き換わります。つまり、配列の代入は、ある配列に別の配列名をつけ、どちらの配列名からも同じ配列を操作できるということです。

配列のコピー

配列の要素を他の配列の要素にコピーするには、１つずつ代入していく方法がありますが、Javaでは System.arraycopyという便利なメソッドが用意されています。

```
System.arraycopy(コピー元配列,
                 コピーを開始する先頭の添字,
                 コピー先配列,
                 コピーを受け取る先頭の添字,
                 コピー要素数)
```

コピー先の配列には、コピーを受け取るのに十分な要素数を確保しておく必要があります。要素数が足りないと、例外「IndexOutOfBoundsException」が投げられます。
たとえば、要素数100のtbl1の内容をtbl2にすべてコピーするには、

```
System.arraycopy(tbl1, 0, tbl2, 0, tbl1.length);
```

となります。tbl1[20]からtbl1[59]の内容をtbl2[60]からtbl2[99]にコピーするには、

```
System.arraycopy(tbl1, 20, tbl2, 60, 40);
```

となります。

オブジェクトの配列

intのような基本型の配列だけでなく、クラスのオブジェクトの配列を作成することもできます。配列の宣言と作成は基本型とまったく同じですが、各要素がnullで初期化されるという点が異なります。これは、各要素がオブジェクトを保持していないことを表しています。このままでは使用できないので、オブジェクトを作成し、各要素に代入しなければなりません。

```
String[]   strArray = new String[3];

strArray[0] = new String("Hard");
strArray[1] = new String("Gay");
strArray[2] = new String("wiseman");
```

この例では、要素数3のStringオブジェクトの配列を宣言し作成しています。各要素はnullで初期化されます。このままでは使用できないので、newを使ってインスタンスを作成し、各要素に代入します。

Stringオブジェクトの配列は、宣言の際に初期化することができます。各要素は、自動的にインスタンスが作成され、与えられた文字列で初期化されます。

```
String[]   strArray = {"Hard", "Gay", "wiseman"};
```