初級プログラミング Ⅰ

第6回　配列

中谷　祐介

第6回の構成

◆ 第1講　配列の基本
- Java における配列について，基本事項を理解する．

◆ 第2講　配列の利用（1）
- 配列を利用した簡単なプログラムを作成する．

◆ 第3講　配列の利用（2）
- 配列を利用した簡単なプログラムを作成する．

◆ 第4講　二次元配列
- 二次元配列について，基本事項を理解する．

第6回　配列

第1講　配列の基本

第1講の学習目標

- Java における配列について，基本事項を理解する：
  - 配列とは
  - 配列の宣言
  - 基本的な配列の利用
  - 配列の初期化

前回までの講義では…

- Java におけるプログラムの流れを制御する構造を学んだ
- 条件分岐：ある条件に従ってその後の処理を変える.
  - `if` 文 ― ある条件に従って，二方向に分岐する.
  - `switch` 文 ― 整数値に基づいて，多方向に分岐する.
- 繰り返し：同様の処理を繰り返す.
  - `for` 文 ― 一定の回数繰り返す
  - `while` 文 ― ある条件が成立している間繰り返す

- 作成できるプログラムの幅が広がった.

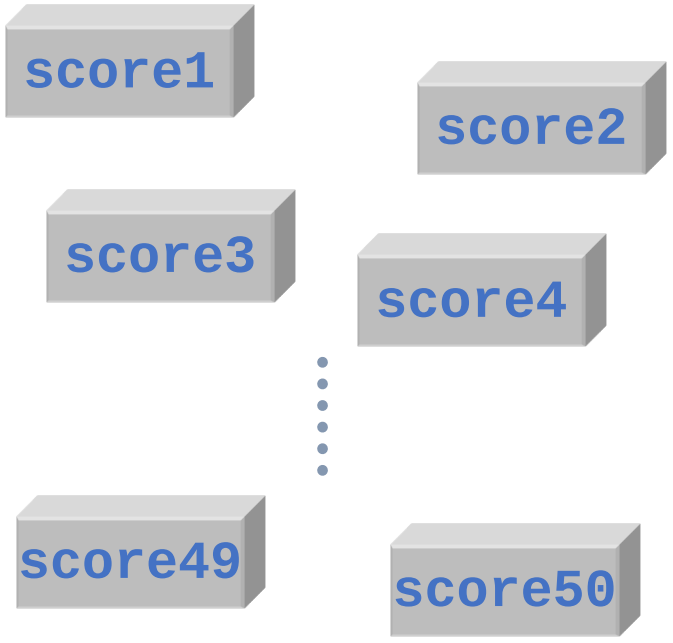
例題6-1

例題6-1

ある科目のテストを実施し，50人分の点数のデータが得られているとします．このとき，全体の平均点と最高点を求め，標準出力に出力するプログラムを作成してください．

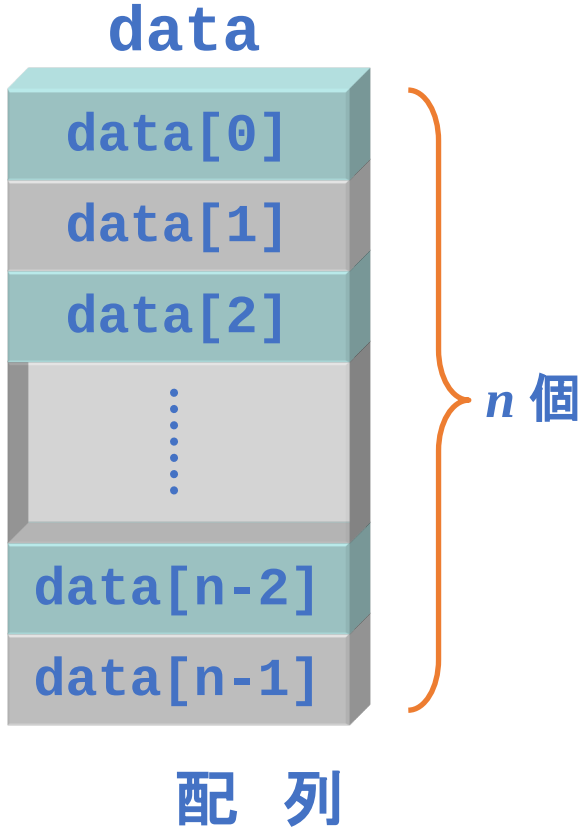
- 50人分のデータを扱うためには，50個の変数が必要．
- 50個の変数の宣言や管理は大変．最高点などを求める処理も複雑．
- 50人分のデータは「テストの点数」（整数値）という共通の性質を持つ変数として扱える．
- プログラミングでは，**配列**を利用すると，多数の変数を効率よく扱える．

➡ 第6回では，配列の基本を理解し，配列を用いたプログラムを作成できるようになることを目的とする．



配列

- **配列**は，同じ型の変数を複数個まとめて扱うことができるデータの構造である．
- 配列は，変数と同様に利用することができる．
  - 利用するには宣言をする必要がある．
  - 目的に応じて型を指定する．
- 配列を使用すると，図のように n 個の変数を1つの変数名で利用することができる．
- 配列を宣言するためには，以下を行う：
  1. 配列変数の宣言
  2. 配列の実体の生成
  3. 配列変数と実体の関連付け



配列の宣言（1）

◆ **例** 5個の int 型（整数）の値を格納できる配列 data を宣言する.

◆ 配列のための変数を宣言する:

```
int[] data;     // 方法1
int data[];     // 方法2
```

- どちらの記述も同じ意味. 本講義では「方法1」で統一する.
- この宣言により, data という変数名が使用できる.

◆ 配列の実体を生成し, 変数と関連付ける:

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

- 「new int[5]」により, int 型5個分の領域を生成.
- 「=」で変数 data と関連付ける.

配列の宣言（2）

◆ **例** 5個の int 型（整数）の値を格納できる配列 data を宣言する.

◆ 前頁より，配列の宣言には以下の3つの手続きが必要：

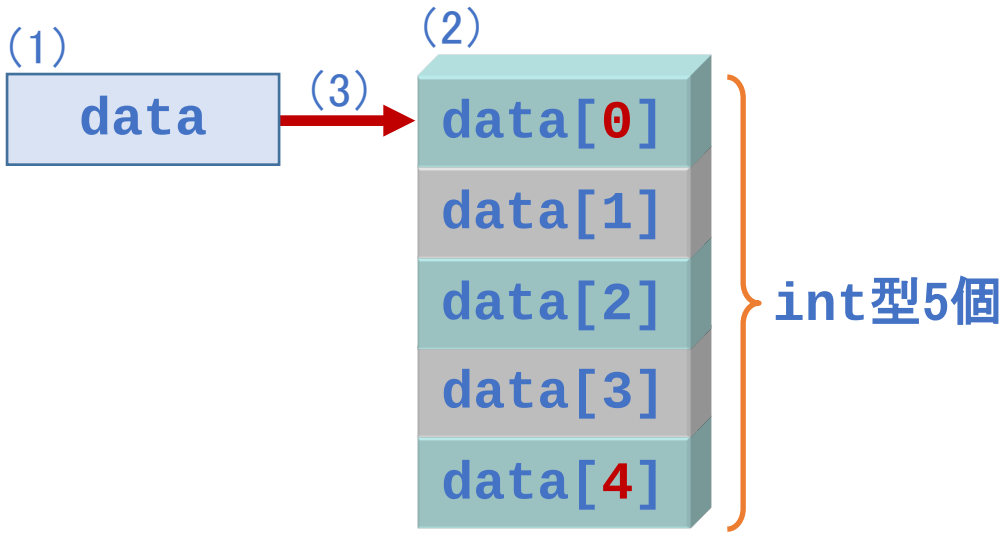
```
int[] data;              ← (1)配列名を宣言する.
data = new int[5];       ← (2)配列の実体を生成し，(3)配列名と関連付ける.
```

- 上記のイメージが，右下図.

◆ 上記をまとめて，次のように記述できる：

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

- 本科目では，まとめて記述する.
- data[0] ～ data[4] の5個の**要素**が利用できる（data[5]は存在しない）.
- 各要素は，int 型の変数と同じ扱い.
- [] の中の数字を**添字**と呼ぶ.



配列 data の宣言のイメージ

基本的な配列の利用（1）

[Sample_06_01.java]

◆ 配列を宣言することで，配列の各要素を変数と同様に扱える．

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

a[0] = 10;
a[1] = a[0]+15;
a[2] = a[1]*4;
System.out.println("a[2] : " + a[2]);
```

```
a[2] : 100
```

- 要素数「3」の int 型の配列 a を宣言．
- 配列 a の各要素 a[0]，a[1]，a[2] を int 型の変数として使用できる．

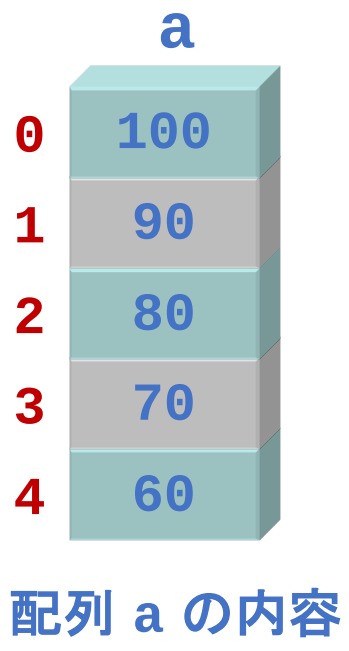
基本的な配列の利用（2）

[Sample_06_02.java]

◆ 多くの場合，配列の処理は繰り返し（for文）と組み合わせて使用する：

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

for (int i=0; i<5; i++) {
    a[i] = 100 - i*10;
    System.out.println(a[i]);
}
```

```
100
90
80
70
60
```



配列 a の内容

- 配列の添字には変数を利用できる．
- 上の for 文は i が「0」から「4」までの5回の繰り返しを行っている．

基本的な配列の利用（3）

- Java の配列では，配列の長さ（要素数）の情報が保存されている．
- 配列の長さは，「配列名`.length`」で取得することができる．
- 前頁の内容は，以下と同じ：

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

for (int i=0; i<a.length; i++) {
    a[i] = 100 - i*10;
    System.out.println(a[i]);
}
```

```
100
90
80
70
60
```

  - 配列 a は要素数「5」の配列．
  - a.length で，配列 a の要素数「5」を取得できる．

基本的な配列の利用（4）

[Sample_06_03.java]

◆ 標準入力に入力した値を，配列の各要素に格納する例：

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

for (int i=0; i<3; i++) {
    a[i] = scan.nextInt();
}

for (int i=0; i<3; i++) {
    System.out.println(a[i]);
}
```

- 配列の各要素 a[i] は，変数として利用できる．
- 実行例

標準入力

```
100
200
300
```

標準出力

```
100↵
200↵
300↵
```

配列の初期化（1）

[Sample_06_04.java]

- 変数と同様に，配列に対しても初期化を行うことができる．
- 配列の初期化の例：

```
int[] a = {10, 20, 30, 40, 50};     // 配列 a の初期化
for (int i=0; i<a.length; i++) {
    System.out.print(a[i] + " ");
}
```

```
10 20 30 40 50
```

  - {} で囲んだ中に，値を「,」で区切って列挙する．
  - 列挙した値を，配列の先頭要素から順に格納する．
  - 値の個数から配列の要素数が決定されるため，new を用いた記述は必要なし．

- 配列の初期化は，配列の宣言と同時に行う場合のみ可能．以下の場合は，エラー：

```
int[] a;
a = {10, 20, 30, 40, 50};      ← エラー
```

配列の初期化（2）

[Sample_06_04.java]

- 変数の場合，宣言しただけでは変数に値は代入されていない：

```
int a;
System.out.println(a);      ← エラー
```

  - 変数 a は宣言のみで値は不定．その内容を参照すると，エラーとなる．

- 配列の場合，宣言を行うと，配列の各要素には型に応じた初期値が格納される．
- 配列の型と初期値は右表のとおり．
- 配列の初期値の確認の例：

```
int[] b = new int[5];
for (int i=0; i<b.length; i++) {
    System.out.print(b[i] + " ");
}
```

```
0 0 0 0 0
```

  - int 型の配列 b を宣言すると，b の各要素には「0」が格納される．

配列の初期値

配列の型	型の意味	初期値
int	整数値	0
double	実数値	0.0
char	文字	'¥u0000'（空文字）
boolean	真偽値	false（偽）

第1講のまとめ

- Java における配列について，基本事項を理解した：
  - 配列とは
  - 配列の宣言
  - 基本的な配列の利用
  - 配列の初期化

第6回　配列

第1講　配列の基本

終わり

第6回　配列

第2講　配列の利用（1）

第2講の学習目標

- 配列を利用した簡単なプログラムが作成できるようになる:
  - 例題を通してのプログラム作成
  - `final` 変数
  - 配列の要素の最大値

例題6-1（再掲）

例題6-1

ある科目のテストを実施し，50人分の点数のデータが得られているとします．このとき，全体の平均点と最高点を求め，標準出力に出力するプログラムを作成してください．

◆プログラムの流れ

- 配列を宣言する．
- 50人分の点数データを配列に格納する．
- 全体の平均点を求める．
- 全体の最高点を求める．
- 結果を出力する．

◆「例題6-1」のプログラムを配列を利用して作成することで，配列の理解を深める．

配列の宣言と点数データの設定

- 点数データを格納する配列の宣言：

```
int[] score = new int[50];
```

  - 要素数「50」の int 型の配列 score を宣言.

- 点数データの設定 ：i 番の学生に対して「(i*87 + 31) % 101」
  - 「0」～「100」の値として設定.
  - 計算式に意味はない．一見，ランダムに見える値として設定.

- 配列の宣言と点数データの設定：

```
int[] score = new int[50];
for (int i=0; i<50; i++) {
    score[i] = (i*87 + 31) % 101;
}
```

  - 学生の人数（50人）が変更された場合，2カ所変更する必要あり.

final 変数

- 「例題6-1」において，学生の人数「50」は定数として扱われ，変更されることはない
- 定数として扱う変数は，**final 変数**という変数として宣言できる．
- final 変数の宣言と初期化：

```
final int SIZE = 50;
```

  - 変数宣言の前に「final」を付けることで，final 変数として宣言できる．
- final 変数は，値を変更することができない：

```
final int SIZE = 50;
SIZE = 100;                ← エラー
```

  - final 変数の宣言後（初期化後）は，その変数に値を代入するとエラーとなる．

```
final int SIZE;
SIZE = 50;                 ← OK
SIZE = 100;                ← エラー
```

  - 一度目の代入（初期化）はできるが，二度目以降はエラー．

final 変数の使用例（1）

[Sample_06_05.java]

final 変数の使用例

```
final int SIZE = 5;

//SIZE = 10;                                    // この行を有効にするとエラーとなる.
System.out.println("SIZE : " + SIZE);

int[] a = new int[SIZE];
for (int i=0; i<SIZE; i++) {
    a[i] = 100 - i*10;
}

for (int i=0; i<SIZE; i++) {
    System.out.print (a[i] + " ");
}
```

```
SIZE : 5
100 90 80 70 60
```

- 先頭の行の「5」を変更することで，プログラム内の SIZE の値をすべて変更できる.

final 変数の使用例（2）

◆「例題6-1」での final 変数の使用

```
final int SIZE = 50;

int[] score = new int[SIZE];
for (int i=0; i<SIZE; i++) {
    score[i] = (i*87 + 31) % 101;
}
```

- 学生の人数「50」を，定数として final 変数 SIZE で宣言．
- int 型の配列 score を宣言し，各要素に点数データを代入．

◆次頁以降の内容では，以下は事前に宣言されているとする：

- final 変数 SIZE（「50」で初期化）．
- 50人分の点数データを格納した int 型の配列 score．

全体の平均点

- 全体の平均点は次で計算できる：（全学生の点数の合計）/（人数）
- 全体の平均点を求めるプログラムは以下のとおり:

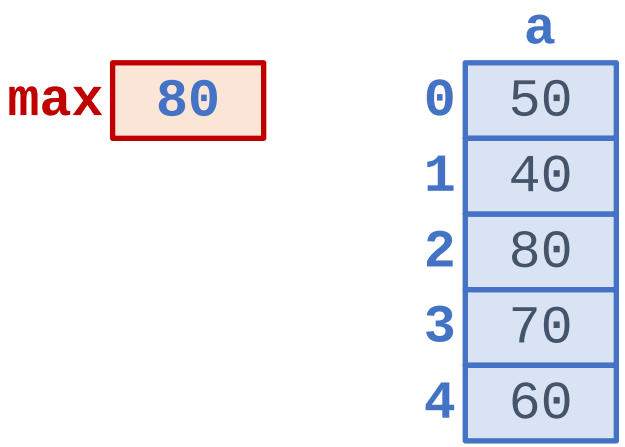
```
int total = 0;
for (int i=0; i<SIZE; i++) {
    total += score[i];
}
double average = (double)total / SIZE;
System.out.printf("平均点 : %.2f¥n", average);
```

- 合計を格納するために，int 型の変数 total を宣言し「0」で初期化.
- total に，点数データ score[i] を順次，加える.
- 全体の合計(total)を学生の人数(SIZE)で割る.
- このとき，キャスト演算子を使用し「(double)total」として double 型に変換.
- これにより，「double 型 / int 型」の計算とし，double 型で計算.

配列の要素の最大値（1）

- 「例題6-1」では，「50人の学生の最高点を求める」ことを行う．
- 一般的には，「配列の要素の最大値・最小値を求める」問題と考えることができる．
- 配列 a の要素の最大値を求める手順

1. 配列 a の先頭要素を，最大値の候補として変数 max に格納．
2. 配列 a の2番目から最後の各要素に対して，以下を繰り返し行う：
   - 要素の値（a[i]）と max の値を比較する．
   - a[i] が大きければ，最大値の候補として max の値を更新する．
3. 繰り返しが終了した段階で，変数 max に格納されている値が最大値．



配列の要素の最大値（2）

[Sample_06_06.java]

◆ 配列 a の要素の最大値を求めるプログラム

```
int[] a = {50, 40, 80, 70, 60};

int max = a[0];                 // 先頭の要素を，最大値の候補として max に格納
for (int i=1; i<5; i++) {       // 2番目から最後の各要素に対して繰り返し
    if (max < a[i]) {           //    要素の値と max の値を比較．要素の値が大きければ，
        max = a[i];             //    最大値の候補として max の値を更新
    }
}
System.out.println(max);
```

```
80
```

全体の最高点

◆ 全学生の点数の最高点を求めるプログラム

```
int max = score[0];
for (int i=1; i<SIZE; i++) {
    if (max < score[i]) {
        max = score[i];
    }
}
System.out.println("最高点 : " + max);
```

- 最高点を格納するために，int 型の変数 max を宣言し score[0] で初期化.
- 2番目の点数から順に max と比較し，点数が大きければ，最高点の候補として max を更新.
- for 文が終了した段階で，max の内容が最高点.

「例題6-1」のプログラム例（1）

Example_06_01.java（1/2）

```
1  public class Example_06_01 {
2      public static void main(String[] args) {
3          final int SIZE = 50;
4
5          int[] score = new int[SIZE];
6          for (int i=0; i<SIZE; i++) {
7              score[i] = (i*87 + 31) % 101;
8          }
9
10         for (int i=0; i<SIZE; i++) {
11             System.out.printf("%4d", score[i]);
12             if ((i+1) % 10 == 0) System.out.println();
13         }
14
```

◆ 実行例

標準出力

```
  31  17   3  90  76  62  48  34  20   6
  93  79  65  51  37  23   9  96  82  68
  54  40  26  12  99  85  71  57  43  29
  15   1  88  74  60  46  32  18   4  91
  77  63  49  35  21   7  94  80  66  52
平均点：49.58
```

「例題6-1」のプログラム例（2）

Example_06_01.java (2/2)

```
15          int total = 0;
16          for (int i=0; i<SIZE; i++) {
17              total += score[i];
18          }
19          double average = (double)total / SIZE;
20          System.out.printf("平均点 : %.2f¥n", average);
21
22          int max = score[0];
23          for (int i=0; i<SIZE; i++) {
24              if (max < score[i]) {
25                  max = score[i];
26              }
27          }
28          System.out.println("最高点 : " + max);
29      }
30  }
```

◆ 実行例

標準出力

```
77 63 49 35 21  7 94 80 66 52↵
平均点 : 49.58↵
最高点 : 99↵
```

第2講のまとめ

- 配列を利用した簡単なプログラムを作成した:
  - 例題を通してのプログラム作成
  - `final` 変数
  - 配列の要素の最大値

第6回　配列

第2講　配列の利用（1）

終わり

第6回　配列

第3講　配列の利用（2）

第3講の学習目標

- 配列を利用した簡単なプログラムが作成できるようになる:
  - 例題を通してのプログラム作成
  - `Scanner` クラス
  - `break` 文, `continue` 文の復習

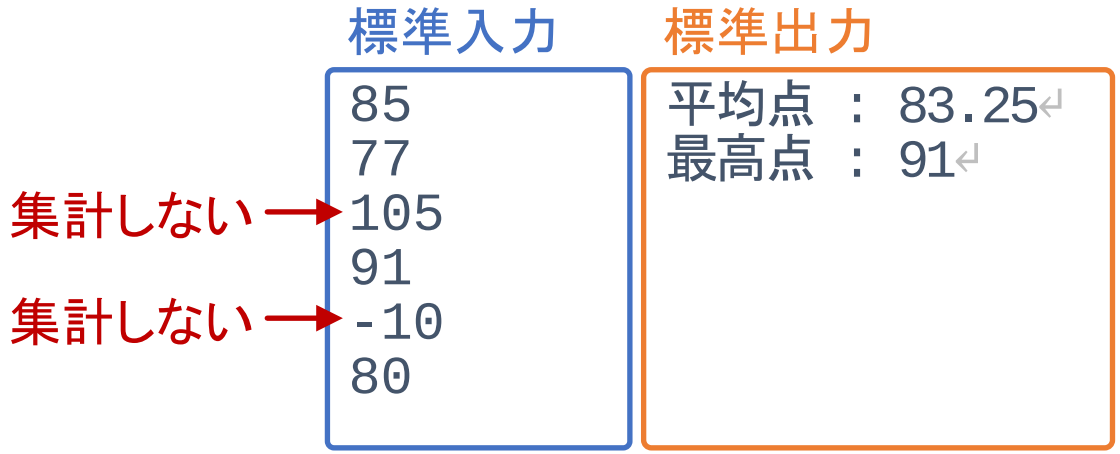
例題6-2（1）

例題6-2

ある科目のテストを実施し，点数のデータ（0点～100点）が得られているとします．
このとき，以下の要件に従って，全体の平均点と最高点を求め，標準出力に出力するプログラムを作成してください：

- 点数のデータを標準入力に入力する．
- 集計するデータの最大数は「50」とする．
- 適切でないデータ（負の値，101以上の値）が入力された場合は，集計に含めない．

◆ 実行例



例題6-2（2）

◆プログラムの流れ

- 配列を宣言する.
- 要件に従って，点数データを配列に格納する.
  - 点数データは，標準入力から読み込む.
  - 集計するデータの最大数は「50」.
  - 適切でないデータ（負の値，101以上の値）は，集計に含めない.
- 全体の平均点を求める.
- 全体の最高点を求める.
- 結果を出力する.

「例題6-1」のプログラムを使用できる.

データの入力（1）

- 最大50個の点数データを標準入力から読み込む：

```
Scanner scan = new Scanner(System.in);
final int SIZE = 50;

int[] score = new int[SIZE];
int num = 0;
while (num < SIZE) {
    int n = scan.nextInt();

    score[num] = n;
    num++;
}
```

  - データの最大数を final 変数 SIZE として宣言し，「50」で初期化．
  - 点数データを格納する配列 score を，要素数 SIZE で宣言．
  - 50個の点数データを標準入力から読み込み，配列 score の要素に格納．

データの入力（2）

◆ 前頁で示したプログラムに対して，問題の要件を追加する：

- 点数データは，標準入力から読み込む．
  ⇒ 標準入力にデータがなくなったら，読み込みを終了する．
- 集計するデータの最大数は「50」．
  ⇒ 対応済み．
- 適切でないデータ（負の値，101以上の値）は，集計に含めない．
  ⇒ 適切でないデータは，配列に格納しない．

Scanner クラス（1）

- Scanner クラスには，数値をプログラム内に読み込む道具がある：
  - nextInt メソッド：整数値（int 型）を読み込むメソッド．
  - nextDouble メソッド：実数値（double 型）を読み込むメソッド．

- Scanner クラスには，次に読み込む数値があるかどうかを調べる道具がある：
  - hasNextInt メソッド：次に読み込む整数値があるかどうかを調べるメソッド．
    整数値があれば true，なければ false となる．
  - hasNextDouble メソッド：次に読み込む実数値があるかどうかを調べるメソッド．
    実数値があれば true，なければ false となる．

Scanner クラス（2）

[Sample_06_07.java]

◆ 標準入力に入力した整数値を10倍して標準出力に出力するプログラム：

```
Scanner scan = new Scanner(System.in);
final int SIZE = 5;

int num = 0;
while (num < SIZE) {
    if (!scan.hasNextInt()) break;
    int n = scan.nextInt();
    System.out.println(n*10);
    num++;
}
```

- 入力できるデータの最大数は5個．
- break 文に出会うと，繰り返しから抜け出す．
- 標準入力に次に読み込む整数値がない場合（!scan.hasNextInt()），繰り返し処理から抜け出す．

◆ 実行例1

標準入力	標準出力
1 3 5	10↵ 30↵ 50↵

◆ 実行例2

標準入力	標準出力
1 3 5 7 9 11	10↵ 30↵ 50↵ 70↵ 90↵

標準入力からのデータの読み込み

◆「例題6-2」において，標準入力にデータがなくなったら，読み込みを終了する：

```
Scanner scan = new Scanner(System.in);
final int SIZE = 50;

int[] score = new int[SIZE];
int num = 0;
while (num < SIZE) {
    if (!scan.hasNextInt()) break;
    int n = scan.nextInt();

    score[num] = n;
    num++;
}
```

- 標準入力にデータがなくなったら，while 文を抜ける．
- もしくは，データの最大数（50個）まで繰り返す．

continue 文

[Sample_06_08.java]

- continue 文に出会うと，その位置からその繰り返し回の終わりまでをスキップし，次の回に処理を移す．
- 標準入力に正の整数値を5個入力すると，1桁の数値のみ出力するプログラム：

```
Scanner scan = new Scanner(System.in);
final int SIZE = 5;

int num = 0;
while (num < SIZE) {
    int n = scan.nextInt();
    num++;
    if (n > 9) continue;
    System.out.println(n);
}
```

- 実行例

標準入力	標準出力
1 3 20 7 40	1↵ 3↵ 7↵

- 読み込んだ値（n）が「9」より大きい場合，出力をスキップする．

適切なデータのみの集計

◆「例題6-2」において，適切でないデータ（負の値，101以上の値）は，配列に格納しない：

```
Scanner scan = new Scanner(System.in);
final int SIZE = 50;

int[] score = new int[SIZE];
int num = 0;
while (num < SIZE) {
    if (!scan.hasNextInt()) break;
    int n = scan.nextInt();
    if ((n < 0) || (n > 100)) continue;

    score[num] = n;
    num++;
}
```

- 標準入力から読み込んだデータ（n）が「0」より小さいか「100」より大きいときには，配列へのデータの格納をスキップ．
- 入力したデータ数は，変数 num に格納されている．

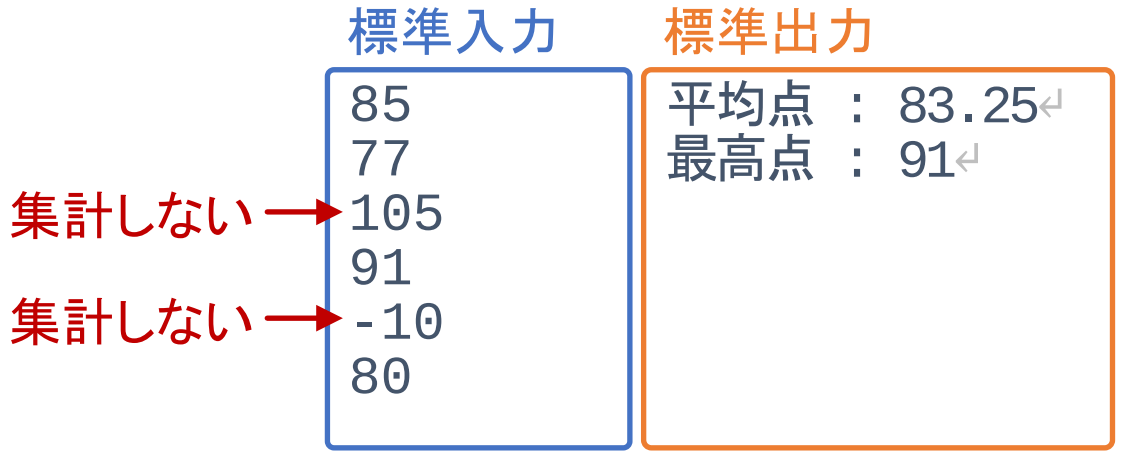
例題6-2（再掲）

例題6-2

ある科目のテストを実施し，点数のデータ（0点～100点）が得られているとします．
このとき，以下の要件に従って，全体の平均点と最高点を求め，標準出力に出力するプログラムを作成してください：

- 点数のデータを標準入力に入力する．
- 集計するデータの最大数は「50」とする．
- 適切でないデータ（負の値，101以上の値）が入力された場合は，集計に含めない．

◆ 実行例



「例題6-2」のプログラム例（1）

Example_06_02.java(1/2)

```
1  import java.util.Scanner;
2
3  public class Example_06_02 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6          final int SIZE = 50;
7
8          int[] score = new int[SIZE];
9          int num = 0;
10         while (num < SIZE) {
11             if (!scan.hasNextInt()) break;
12             int n = scan.nextInt();
13             if ((n < 0) || (n > 100)) continue;
14
15             score[num] = n;
16             num++;
17         }
```

「例題6-2」のプログラム例（2）

Example_06_02.java (2/2)

```
18
19          int total = 0;
20          for (int i=0; i<num; i++) {
21              total += score[i];
22          }
23          double average = (double)total / num;
24          System.out.printf("平均点 : %.2f¥n", average);
25
26          int max = score[0];
27          for (int i=0; i<num; i++) {
28              if (max < score[i]) {
29                  max = score[i];
30              }
31          }
32          System.out.println("最高点 : " + max);
33      }
34  }
```

◆ 実行例

標準入力

```
85
77
105
91
-10
80
```

標準出力

```
平均点 : 83.25
最高点 : 91
```

第3講のまとめ

- 配列を利用した簡単なプログラムを作成した:
  - 例題を通してのプログラム作成
  - `Scanner` クラス
  - `break` 文, `continue` 文の復習

第6回　配列

第3講　配列の利用（2）

終わり

第6回　配列

第4講　二次元配列

第4講の学習目標

- 二次元配列について，基本事項を理解する：
  - 二次元配列
  - 二次元配列の宣言，初期化
  - 二次元配列の処理

例題6-3

例題6-3

3科目のテストを実施し，5人分の点数のデータ（0点～100点）が得られているとします．このとき，各学生の3科目の合計点を計算し，標準出力に出力するプログラムを作成してください．

◆ どのようにプログラムを作成する？

- 科目ごとに点数データを格納した配列を宣言：

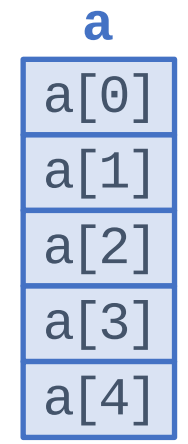
```
final int SIZE = 5;

int[] score1 = new int[SIZE];
int[] score2 = new int[SIZE];
int[] score3 = new int[SIZE];
```

- 3つの配列を組み合わせて使用することになり，処理が煩雑．
- 配列 score を，さらに配列にしては…？

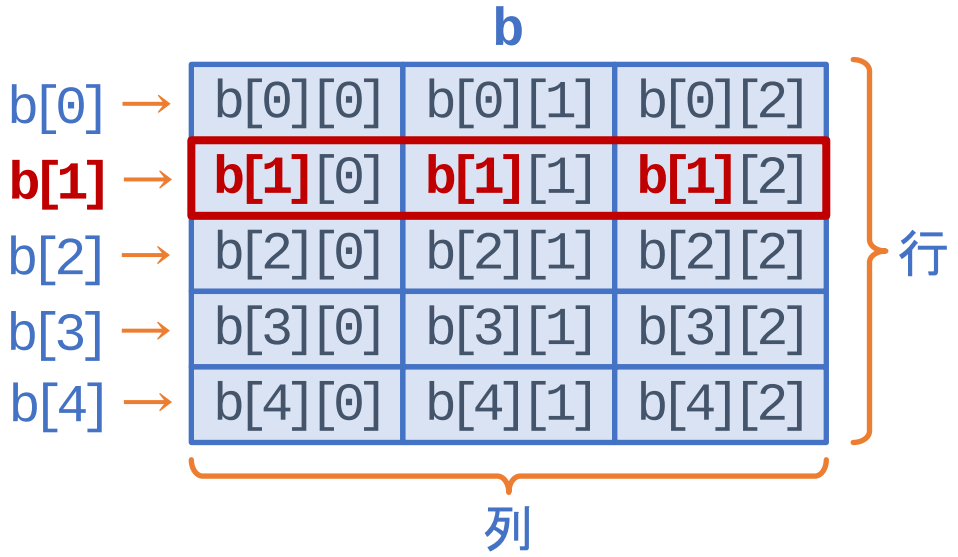
◆ **二次元配列**というものを使用して，1つの配列ですべてのデータを管理する．

二次元配列

- 配列（一次元配列）は，値を要素として構成する．
  - 配列 a に対して，配列の要素 a[i] は値（int 型，double 型，等）．
- **二次元配列**は，配列を要素として構成する配列である．
  - 二次元配列 b に対して，配列の要素 b[i] は配列．
  - 配列 b[i] の要素 b[i][j] は値．
  - 二次元配列の縦方向を**行**，横方向を**列**と呼ぶ．
  - 右図の配列 b は，5行3列の二次元配列．
- 二次元配列の処理には，二重ループを使用する．



配列のイメージ



二次元配列のイメージ

二次元配列の宣言

- 配列の宣言：**例** 要素数「5」の int 型の配列 a の宣言

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

- 二次元配列の宣言：**例** 5行3列の int 型の二次元配列 b の宣言

```
int[][] b = new int[5][3];
```

  - この宣言により，右下図をイメージとする配列を作成．
  - b[0][0] ～ b[4][2] の15個の要素を利用できる．
  - 各要素 b[i][j] は，int 型の変数と同じ扱い．
  - 「int[][] b」は，「int b[][]」としてもよい．
  - 二次元配列を宣言すると，（一次元）配列と同様に，型に応じた初期値が各要素に格納される．

a

a[0]
a[1]
a[2]
a[3]
a[4]

配列 **a**

b

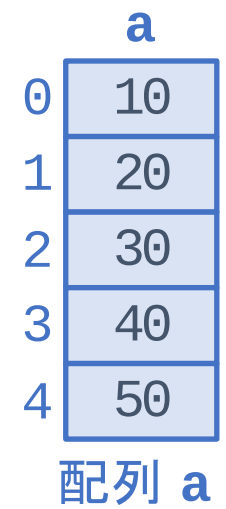
b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]
b[2][0]	b[2][1]	b[2][2]
b[3][0]	b[3][1]	b[3][2]
b[4][0]	b[4][1]	b[4][2]

二次元配列 **b**

二次元配列の初期化

◆（一次元）配列の初期化

```
int[] a = {10, 20, 30, 40, 50};
```

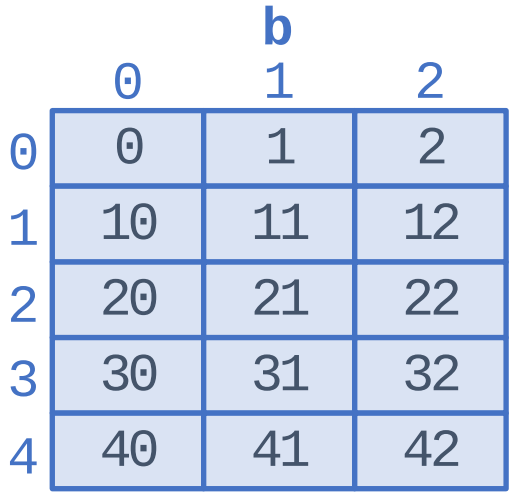


◆二次元配列の初期化

```
int[][] b = {{0,1,2},{10,11,12},{20,21,22},{30,31,32},{40,41,42}};
```

- 各行ごとに初期化の内容を {} で囲んだ中に，「,」で区切って列挙．
- 各単位で初期化した内容をさらに {} で囲んだ中に，「,」で区切って列挙．
- 次のように記述すると，理解しやすい：

```
int[][] b = {
        {0, 1, 2},
        {10, 11, 12},
        {20, 21, 22},
        {30, 31, 32},
        {40, 41, 42}
};
```



二次元配列の処理

[Sample_06_09.java]

- 二次元配列の処理には，二重ループを使用する．

```
int[][] b = {
        {0, 1, 2},
        {10, 11, 12},
        {20, 21, 22},
        {30, 31, 32},
        {40, 41, 42}
};

for (int i=0; i<5; i++) {
        for (int j=0; j<3; j++) {
                System.out.print(b[i][j] + " ");
        }
        System.out.println();
}
```

```
0 1 2
10 11 12
20 21 22
30 31 32
40 41 42
```

i が「1」のとき →

b

b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]
b[2][0]	b[2][1]	b[2][2]
b[3][0]	b[3][1]	b[3][2]
b[4][0]	b[4][1]	b[4][2]

二次元配列 b

- 5行3列の2次元配列 b の各要素の出力．

二次元配列の大きさ(要素数)

[Sample_06_10.java]

◆ (一次元)配列の長さ(要素数) : 配列名.length

```
int[] a = new int[5];
for (int i=0; i<a.length; i++) {
  a[i] = i*10;
  System.out.print(a[i] + " ");
}
```

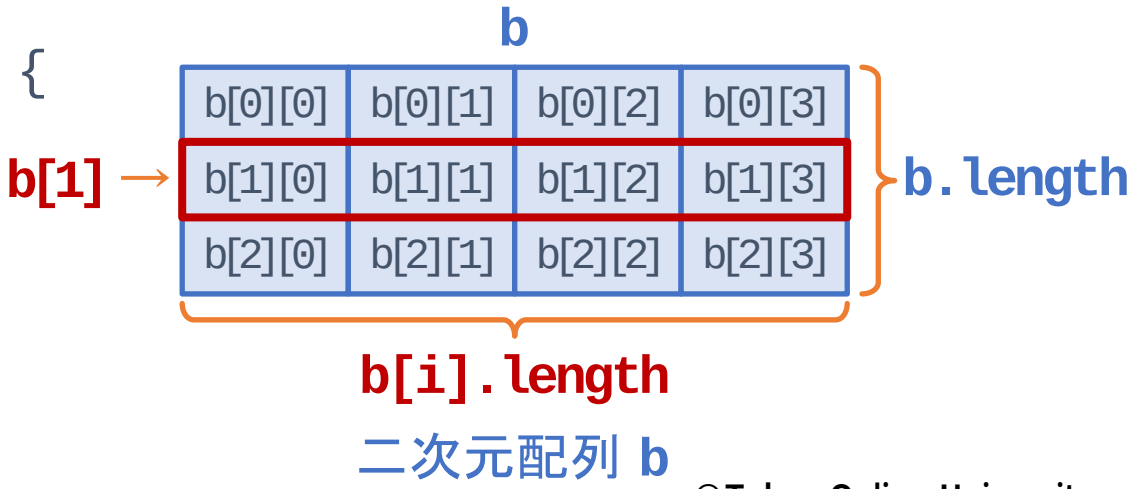
```
0 10 20 30 40
```

◆ 二次元配列の大きさ(要素数)

配列名.length ⇒ 行の数, 配列の行.length ⇒ 列の数

```
int[][] b = new int[3][4];
for (int i=0; i<b.length; i++) {
  for (int j=0; j<b[i].length; j++) {
    b[i][j] = i*10+j;
    System.out.print(b[i][j] + " ");
  }
  System.out.println();
}
```

```
0 1 2 3
10 11 12 13
20 21 22 23
```



[例題6-3] 配列の宣言と点数データの設定

◆ 3科目5人分の点数データを格納する配列の宣言：

```
final int NUM = 5;          // 学生数
final int SUB = 3;          // 科目数

int[][] score = new int[NUM][SUB];
```

- 5行3列の int 型の二次元配列 score を宣言.

	科目A	科目B	科目C
学生0	score[0][0]	score[0][1]	score[0][2]
学生1	score[1][0]	score[1][1]	score[1][2]
学生2	score[2][0]	score[2][1]	score[2][2]
学生3	score[3][0]	score[3][1]	score[3][2]
学生4	score[4][0]	score[4][1]	score[4][2]

◆ 点数データの設定 ― i 番の学生に対して科目ごとに計算式を設定：

```
for (int i=0; i<NUM; i++) {
    score[i][0] = (i*87 + 71) % 101;
    score[i][1] = (i*79 + 23) % 101;
    score[i][2] = (i*59 + 55) % 101;
}
```

- 「0」～「100」の値として設定.
- 計算式に意味はない. 一見, ランダムに見える値として設定.

[例題6-3] 点数データの処理

◆ 学生ごとの点数データの処理

- 「学生2」の3科目の点数データ:
  score[**2**][0], score[**2**][1], score[**2**][2]
- 1番目の添字の値を固定することで，特定の学生の処理を行える.

	科目A	科目B	科目C
学生0	score[0][0]	score[0][1]	score[0][2]
学生1	score[1][0]	score[1][1]	score[1][2]
学生2	score[**2**][0]	score[**2**][1]	score[**2**][2]
学生3	score[3][0]	score[3][1]	score[3][2]
学生4	score[4][0]	score[4][1]	score[4][2]

◆ 科目ごとの点数データの処理

- 「科目B」の5人分の点数データ:
  score[0][**1**], score[1][**1**], … , score[4][**1**]
- 2番目の添字の値を固定することで，特定の科目の処理を行える.

	科目A	科目B	科目C
学生0	score[0][0]	score[0][**1**]	score[0][2]
学生1	score[1][0]	score[1][**1**]	score[1][2]
学生2	score[2][0]	score[2][**1**]	score[2][2]
学生3	score[3][0]	score[3][**1**]	score[3][2]
学生4	score[4][0]	score[4][**1**]	score[4][2]

[例題6-3] 各学生の3科目の合計点

◆「学生i」の3科目の点数データ:

- score[**i**][0], score[**i**][1], score[**i**][2]
- 2番目の添字を使用して合計点を計算.

	科目A	科目B	科目C
学生0	score[0][0]	score[0][1]	score[0][2]
学生1	score[1][0]	score[1][1]	score[1][2]
学生2	score[**2**][0]	score[**2**][1]	score[**2**][2]
学生3	score[3][0]	score[3][1]	score[3][2]
学生4	score[4][0]	score[4][1]	score[4][2]

◆各学生の3科目の合計点の計算

```
for (int i=0; i<NUM; i++) {
    int total = 0;
    for (int j=0; j<SUB; j++) {
        total += score[i][j];
    }
    System.out.println("学生" + i + " : " + total);
}
```

- 内側の繰り返しで, 各学生の3科目の合計点を計算する.
- 外側の繰り返しで, 学生単位の繰り返しを行う.

「例題6-3」のプログラム例

Example_06_03.java

```
1  public class Example_06_03 {
2      public static void main(String[] args) {
3          final int NUM = 5;          // 学生数
4          final int SUB = 3;          // 科目数
5
6          int[][] score = new int[NUM][SUB];
7          for (int i=0; i<NUM; i++) {
8              score[i][0] = (i*87 + 71) % 101;
9              score[i][1] = (i*79 + 23) % 101;
10             score[i][2] = (i*59 + 55) % 101;
11         }
12
13         for (int i=0; i<NUM; i++) {
14             int total = 0;
15             for (int j=0; j<SUB; j++) {
16                 total += score[i][j];
17             }
18             System.out.println("学生" + i + " : " + total);
19         }
20     }
21 }
```

◆ 実行結果

標準出力

```
学生0 : 149
学生1 : 71
学生2 : 195
学生3 : 117
学生4 : 140
```

第4講のまとめ

- 二次元配列について，基本事項を理解した：
  - 二次元配列
  - 二次元配列の宣言，初期化
  - 二次元配列の処理

第6回　配列

第4講　二次元配列

終わり