配列(2)

2次元配列, String

http://java2005.cis.k.hosei.ac.jp/

授業の前に自己点検

- 配列変数に格納される配列のIDと配列の実体の区別ができていますか
- 配列変数の宣言と配列の実体の生成の区別ができていますか
- メソッドの引数に配列が渡されるとき、実際に渡されるものは何ですか
  - このことの重要な帰結は何ですか
- 引数の値渡しと参照渡しということばを例を挙げて説明できますか
- 授業で扱う例題はすべて基本的な処理の定石です
  - 国語や英語の勉強では暗誦すべきものに相当します
  - 何も見なくてもすらすらと口をついて出てきますか

配列変数の宣言、配列の実体の生成、初期化

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

省略形

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

変数の宣言とその初期化、配列の実体の生成とその初期化をまとめて書くと

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

配列の実体の生成とその初期化をまとめて書くと

```
int[] a;
a = new int[4];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
```

配列のIDを格納するための変数を宣言

長さ4の配列の実体を生成し、そのIDをa
に格納する

aが指している配列本体の各要素に値を
格納する

配列の長さ

```
int[] a = {1, 2, 3};
for (int i = 0; i < a.length; i++) {
  System.out.println(a[i]);
}
```

実行結果

```
1
2
3
```

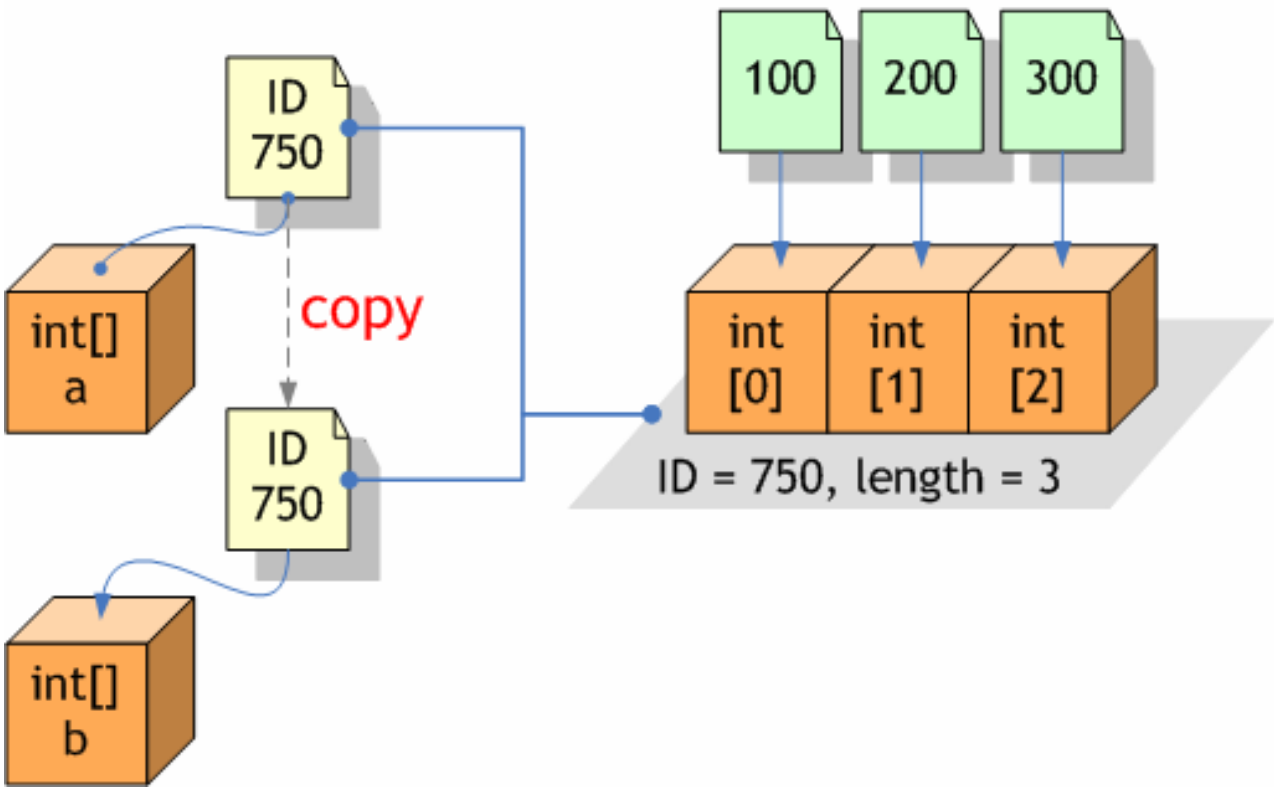
擬似コードで書くと

最後の要素のインデックスは

```
a = {1, 2, 3}
for i を 0 から (配列 a の長さ - 1) まで
  print a[i]
```

配列のIDのコピー

```
int[] a = {100, 200, 300};
int[] b = a;
```

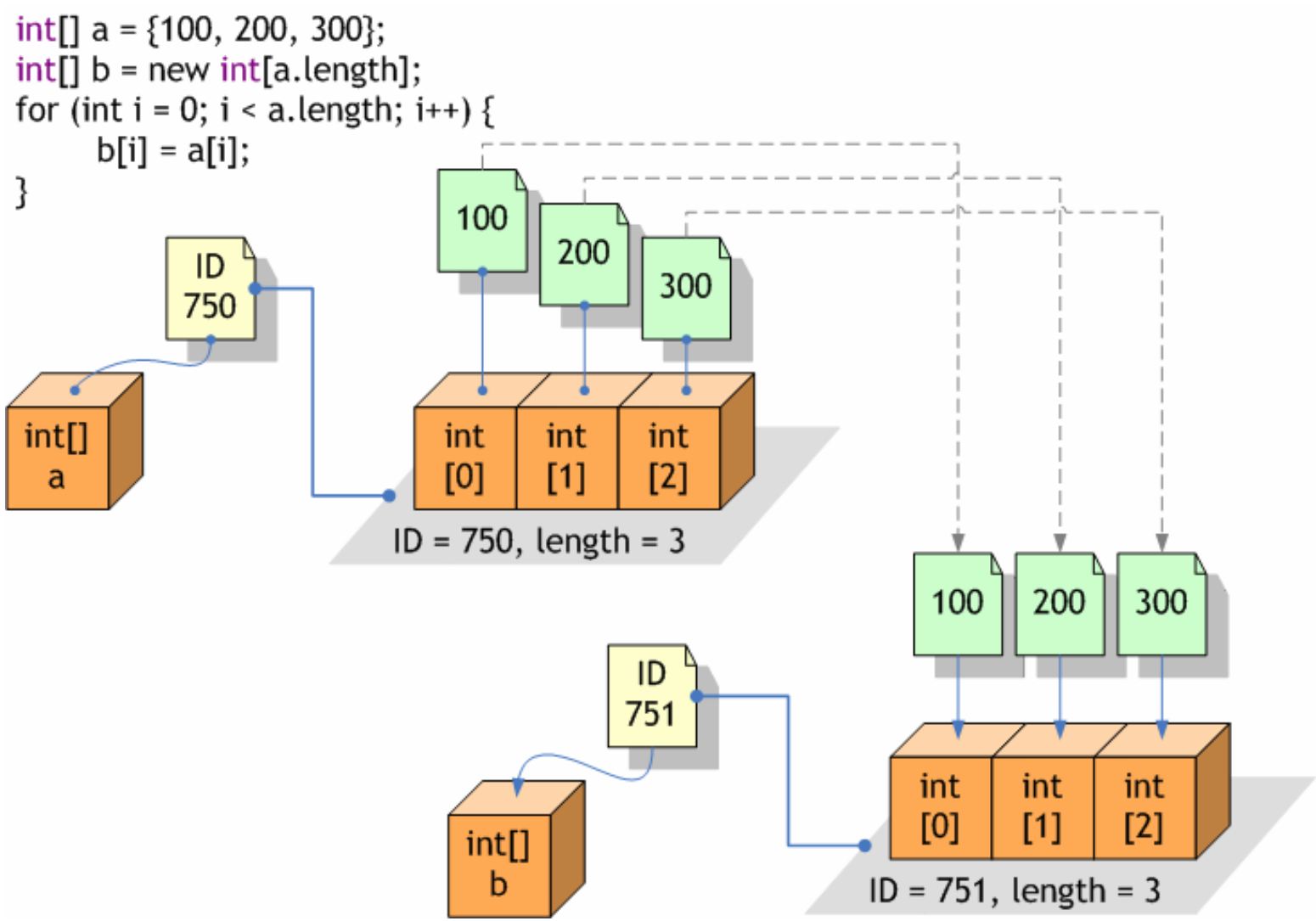


a と b が指し示す**配列の実体(ID=750)** が同じものになってしまっている。

つまり、b の各要素に加えられた変更は a の各要素にも反映してしまうことになる。

配列の実体のコピー

```
int[] a = {100, 200, 300};
int[] b = new int[a.length];
for (int i = 0; i < a.length; i++) {
        b[i] = a[i];
}
```



配列の型はメソッドの引数、戻り値にも使用できる

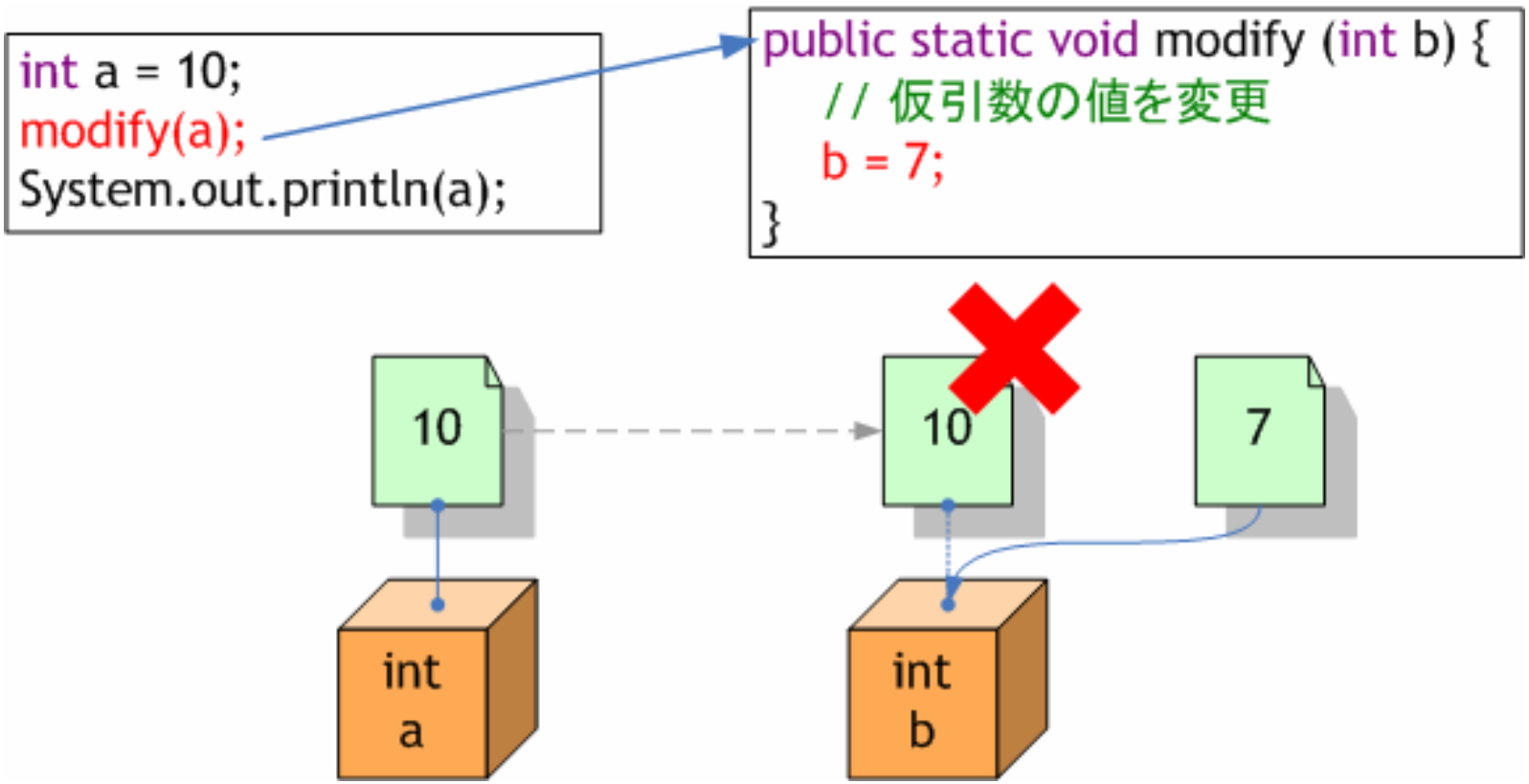
```
public static void main(String[] args) {
  int[] a = createArray();
  ...
}

public static int[] createArray() {
  return new int[]{10, 20, 30, 40};
}

public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.println(array[i]);
  }
}
```

int 配列型

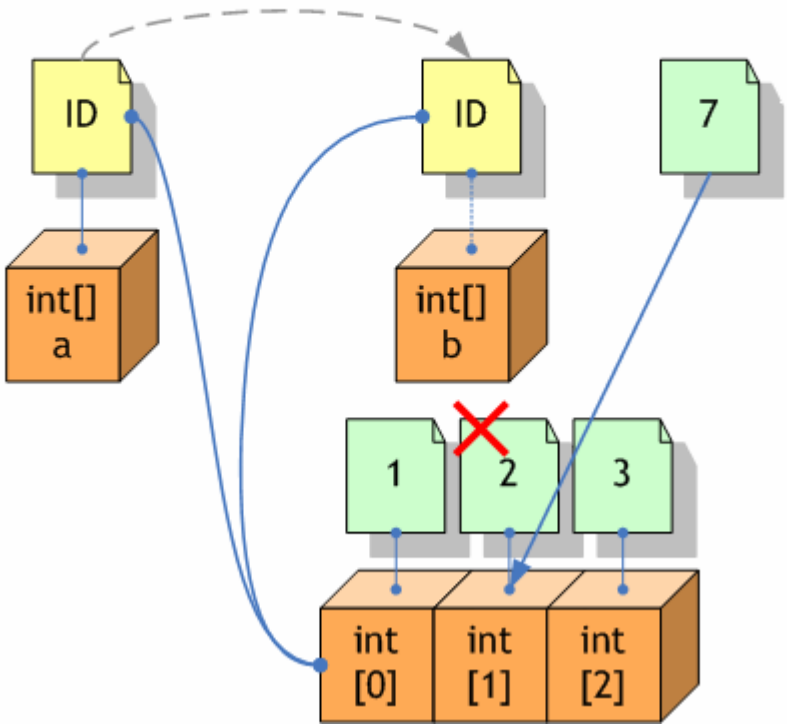
引数の値渡しと参照渡し
int型, double型は値渡し



引数が配列型だと参照渡し
コピーされるのは配列の実体のID

```
int[] a = {1, 2, 3};
modify(a);
System.out.println(a);
```

```
public static void modify (int[] b) {
    // 仮引数の指す要素を変更
    b[1] = 7;
}
```



今回のテーマ

- ２次元配列
  - 2次元の画像処理を行う場合や、行列を用いた演算をするような場面でよく利用される
  - 配列の要素がまた配列
- いろいろな型
  - char型
  - boolean型
  - String型

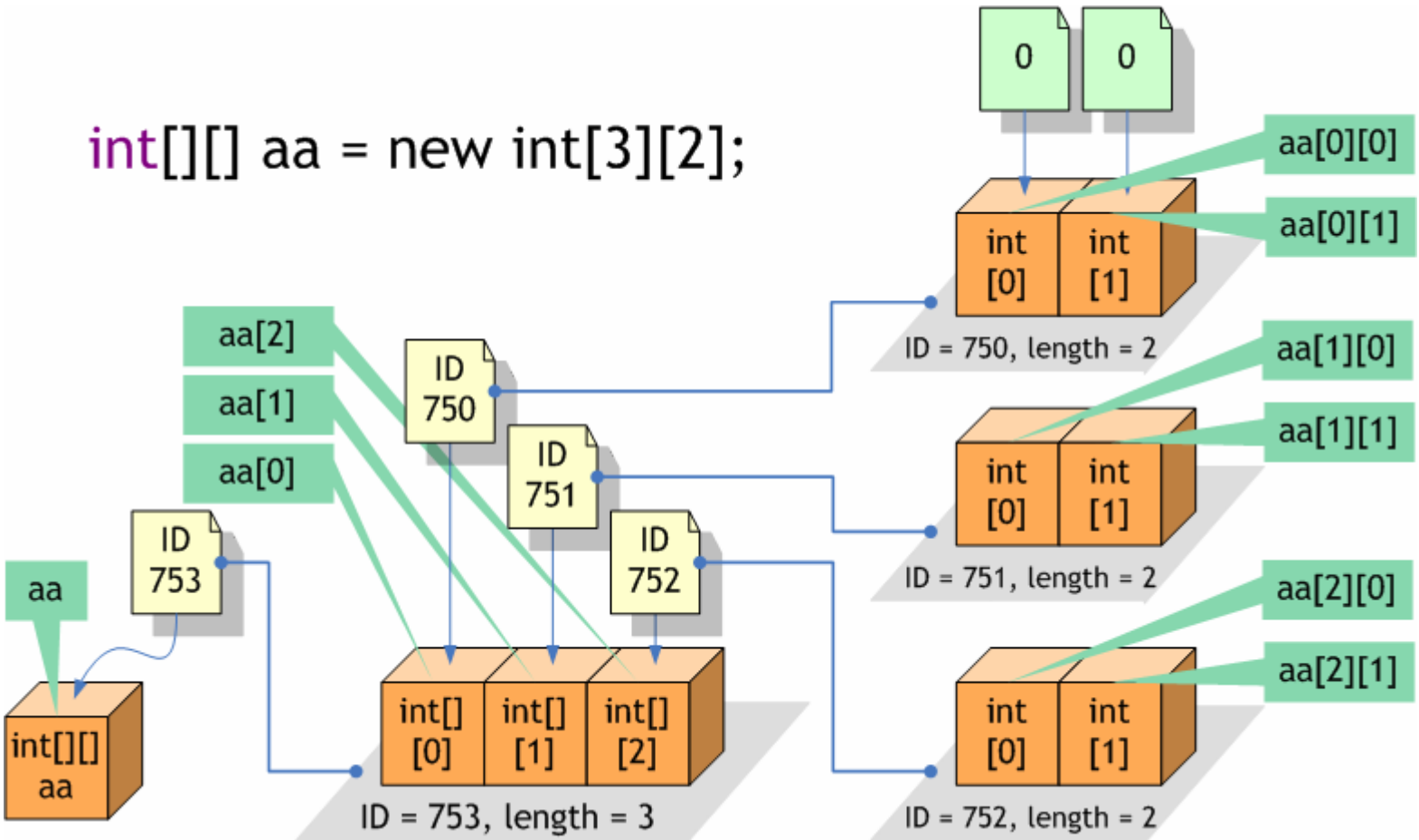
2次元配列は行列を扱うのに便利

```
int[][] aa=new int[3][2];
```

この例では、3行2列の行列を「宣言している」と考えることができる。

$$\begin{pmatrix} aa[0][0] & aa[0][1] \\ aa[1][0] & aa[1][1] \\ aa[2][0] & aa[2][1] \end{pmatrix}$$

実際は下の概念図のようになっているので数学でいう行列のように長方形にデータが並んでいるのとは違う。このことを意識しないといけない場面がある。



２次元配列の変数の宣言

```
int [][] aa;
```

２次元配列の実体を指すIDを格納するための変数aaを宣言する。IDを書き付けた紙をしまうための箱がつくられる。

```
int [][] aa;
aa = new int[3][2];
```


要素を格納する箱をつくるには

```
int [][] aa = new int[3][2];
```

まとめて書ける

３次の実正方行列を扱いたいのなら以下のように宣言すればよい。

```
double [][]  bb = new double[3][3];
```

要素まで一度に初期化する

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

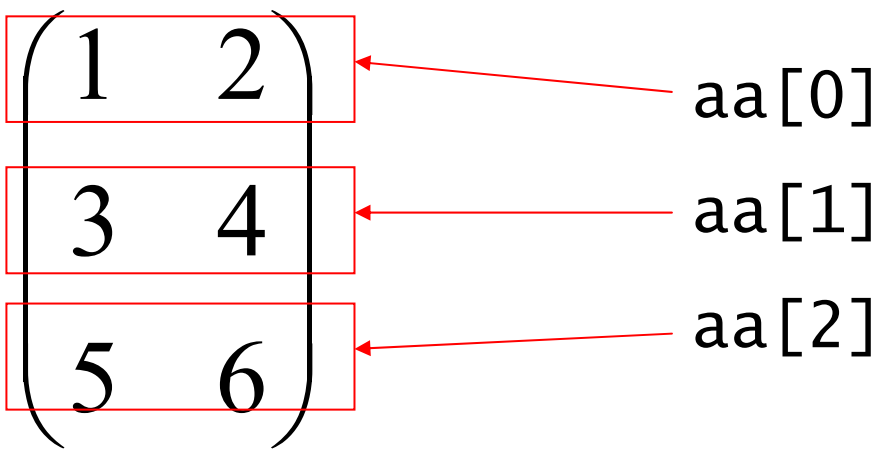
３行２列の行列ができたことになる

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

行単位に管理されている

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

3行2列の行列ができたことになる



２次元配列を走査するときの書き方

```
int[][] aa = {{1,2}, {3,4}, {5,6}};
for (int i = 0; i < aa.length; i++) {
  for (int j = 0; j < aa[i].length; j++) {
    System.out.println("aa[" + i + "][" + j + "] = "
                          + aa[i][j]);
  }
}
```

aa.length　　は行数

aa[i].length　は第i行の長さ（列数）

char型　文字を扱う

```
char c = 'C';
char space = ' ';
char a = 'あ';
```

char型の変数に文字を
格納することができる

```
char a = 'A';
System.out.println(a);
```

char型のデータつまり文字は
println命令やprint命令で表
示することができる

char型のデータの比較

```
char a = 'A';
if (a == 'A') {
 ...
}
```

==で等しいか判定できる

```
char c = 'C';
if ('A' <= c && c <= 'Z') {
 // 大文字のときの処理
}
else if ('a' <= c && c <= 'z') {
 // 小文字のときの処理
}
```

文字には番号が振られているので大小の比較ができる

```
char c = 'C';
if (Character.isUpperCase(c)) {
 // 大文字のときの処理
}
else if (Character.isLowerCase(c)) {
 // 小文字のときの処理
}
```

こちらの方が読みやすい書き方

boolean型 trueとfalseの２つの値だけ

```
if (a < 0) ...
```


この式はboolean型で、式の値はtrueまたはfalseのいずれか

booleanを戻り値にもつメソッド

```
public static void main(String[] args) {
 ...
  if (isEquilateral(a, b, c))
    System.out.println("入力された三角形は正三角形");
  else if (isIsosceles(a, b, c))
    System.out.println("入力された三角形は二等辺三角形");
  else
    System.out.println("入力された三角形は通常の三角形");
  }

public static boolean
isEquilateral(double a, double b, double c) {
  return (a == b && b == c);
}

public static boolean
isIsosceles(double a, double b, double c) {
  return (a == b || b == c || c == a);
}
```

boolean型を明示的に使う

```
// 無限ループ
while (true) {
  ...
}
```

```
// 探索
boolean found = false;
for (int i = 0; i < array.length; i++) {
  if (array[i] == searching) {
    found = true;
  }
}

if (found) {
 // 見つかった場合の処理
}
```

String型　文字列を扱うための型

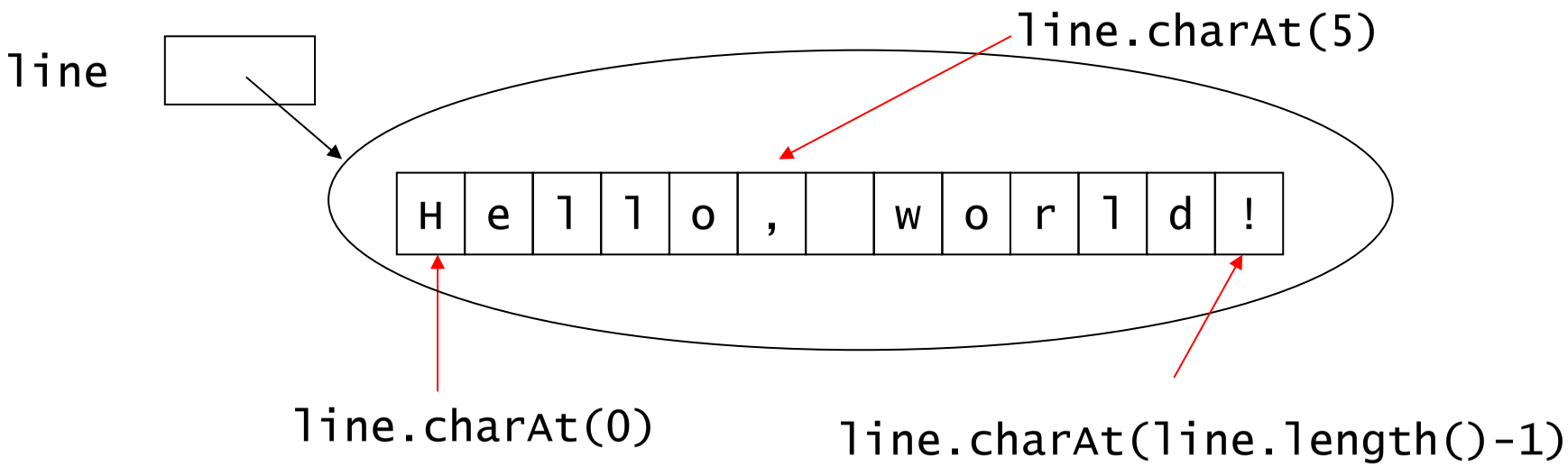
```
String line = "Hello, world!";
int length = line.length();
```

文字列lineの長さは
line.length()で取得できる。

配列aaの長さはaa.length

charAt命令

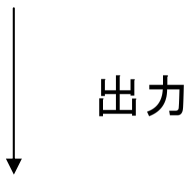
```
String line = "Hello, world!";
// 先頭の文字を取得
char head = line.charAt(0);
// 末尾の文字を取得
char tail = line.charAt(line.length() - 1);
```



charAt命令をループの中で使う

```
String line = "Hello, world!";

for(int i = line.length() - 1; i >= 0; i--){
  System.out.print(line.charAt(i));
}
```




```
!dlrow ,olleH
```

コンソールから入力された文字列

```
int input = Integer.parseInt(reader.readLine());
```


実は2段階の操作をまとめて書いたものだった

```
String line = reader.readLine();
int input = Integer.parseInt(line);
```

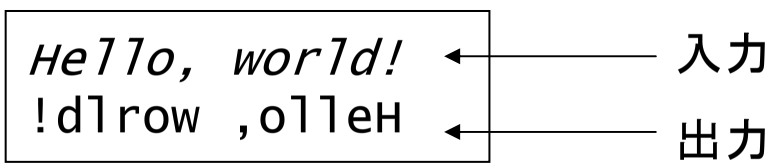
readLine命令が返すのはString型のデータ、つまり文字列

readLine命令で獲得したStringをそのまま処理する

```
public static void main(String[] args) throws IOException {
  BufferedReader reader =
    new BufferedReader(new InputStreamReader(System.in));

  String line = reader.readLine();

  for(int i=line.length()-1; i>=0; i--){
    System.out.print(line.charAt(i));
  }
  System.out.println();
}
```



char[]型 と String型

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

```
char[] array = {'H', 'e', 'l', 'l', 'o'};
```

char []型だと要素の変更ができる

Stringは変更ができない（その代わり便利な点もあるがここでは説明しない）

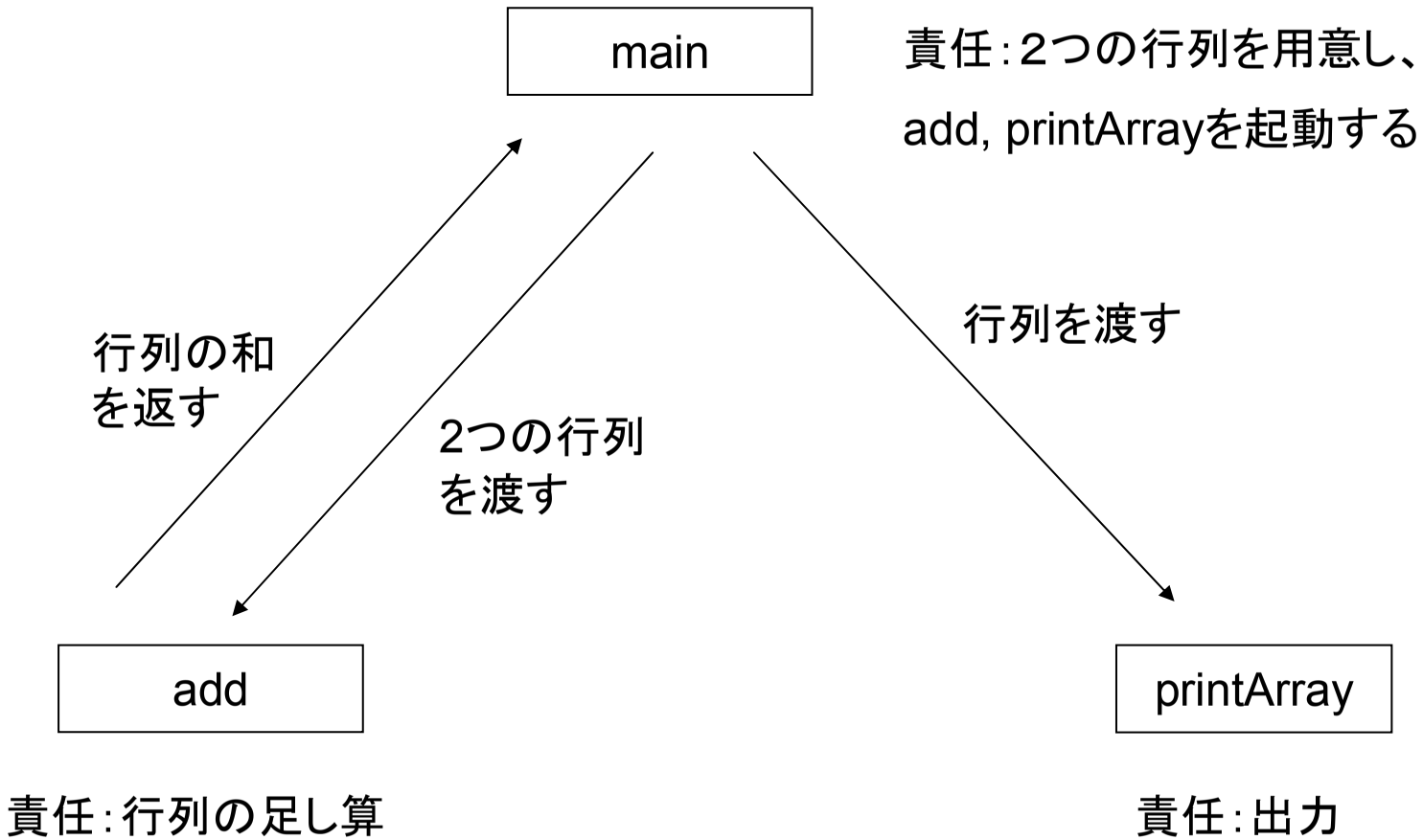
相互の変換もできる（詳しくは講義資料参照）

一緒にやってみよう

- 今回の演習で使うテストドライバをいつものように指示通り正確にインストールする
  - テストドライバの導入に成功すると
    - プロジェクト「java2005」の中の「test」というフォルダに「12」という名前のフォルダが作成される。
    - このフォルダには今週使用するテスト一式が入っている。
- j1.lesson12 というパッケージを作成する
- 演習資料にあるReversePrint, MatrixAdd を擬似コードの検討から開始し、一連のテストを手順通りに実行せよ。

MatrixAddの解説

どのメソッドも責任がある
プログラムは責任のネットワーク



addメッソッド

```
public static int[][] add(int[][] a, int[][] b) {
  int[][] c = new int[a.length][a[0].length];

  for (int i = 0; i < c.length; i++) {
    for (int j = 0; j < c[i].length; j++) {
      c[i][j] = a[i][j] + b[i][j];
    }
  }
  return c;
}
```

行数

第i行の長さ

どの行も同じ長さを想定しているので
列の数と考えてよい

printMatrixメソッド

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


第i行の出力

課題

各自のペースで今回の課題をやってください。

ヒント： 行列の掛け算 c[0][0]

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20} & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$$

```
for(int k=0; k<3; k++){
  c[0][0]+=a[0][k]*b[k][0];
}
```

ヒント： 行列の掛け算 c[0][1]

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} * & a_{00}b_{01} + a_{01}b_{11} + a_{02}b_{21} & * \\ * & * & * \\ * & * & * \end{pmatrix}$$

```
for(int k=0; k<3; k++){
  c[0][1]+=a[0][k]*b[k][1];
}
```

ヒント： 行列の掛け算 c[0][2]

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} * & * & a_{00}b_{02} + a_{01}b_{12} + a_{02}b_{22} \\ * & * & * \\ * & * & * \end{pmatrix}$$

```
for(int k=0; k<3; k++){
  c[0][2]+=a[0][k]*b[k][2];
}
```

ヒント： 行列の掛け算 c[1][0]

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} * & * & * \\ a_{10}b_{00} + a_{11}b_{10} + a_{12}b_{20} & * & * \\ * & * & * \end{pmatrix}$$

```
for(int k=0; k<3; k++){
  c[1][0]+=a[1][k]*b[k][0];
}
```

ヒント： 行列の掛け算 c[i][j]

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix}$$

$$= \begin{pmatrix} & * & * \\ * & a_{i0}b_{0j} + a_{i1}b_{1j} + a_{i2}b_{2j} & * \\ * & * & * \end{pmatrix}$$

```
c[i][j]=0;
for(int k=0; k<3; k++){
  c[i][j]+=a[i][k]*b[k][j];
}
```

ヒント： 行列の掛け算

```
for (int i; … ; … ){
  for (int j; … ; … ){
    c[i][j]の計算
  }
}
```