配列（1）

配列，引数の値渡しと参照渡し

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

授業の前に自己点検
以下のことがらを友達に説明できますか？

- 再帰呼び出しはどのように実行されますか
  - Pascalの三角形, Factorial, Fibonacciなどの例で説明できますか
  - Hanoiの塔も説明できるとプログラミング中級者
  - 今週金曜日の補習でまとめて復習します
- 擬似コードの役割について説明できますか
- テストケース作成について説明できますか
  - 同値分割法
  - 境界値分析法

擬似コード(pseudocode)

- 擬似コードとは
  - 自然言語と通常のプログラミング言語を混ぜたようなコード (プログラム)
  - アルゴリズム辞典などでよく使われる
- 目的は
  - プログラムの処理や流れを簡単に記述するものである。
  - アイデアを整理したり他人に伝えるために用いる。
- 望ましい性質
  - どのプログラミング言語にも書き直しがしやすい

擬似コードの基本事項を適用してみる

「コンソールに入力された値の絶対値を表示する」

擬似コード

```
input = 入力された値
print(|input|)
```

Javaでの実現

```
package j1.lesson10;

import java.io.*;
public class Absolute {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int input = Integer.parseInt(reader.readLine());
    System.out.println(Math.abs(input));
  }
}
```

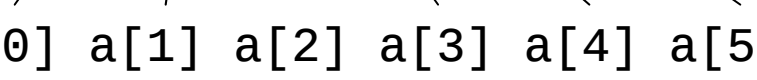
今回のテーマ

- 一次元配列
  - 同じ型のデータを一列に並べたもの
- 引数の値渡しと参照渡し

宣言と参照

配列変数の宣言と
配列の初期化

```
int[] a = {3, 5, 7, 9, 11, 13};
```



参照はこのように　a[0] a[1] a[2] a[3] a[4] a[5]

これを実行すると

```
System.out.println(a[4]);
```



```
11
```

初期化付き生成

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

は

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

の省略形。右辺で配列の実体がつくられ、そのIDが書かれた紙がaという箱に格納される。

配列変数の宣言と同時の初期化のときだけ省略できる。

配列の実体を生成するための式

```
new 型の名前[] { 初期値[0]を表す式, 初期値[1]を表す式, 初期値[2]を表す式, ... }
```

例えば

```
double[] b = new double[]{1.0, 2.0};
```

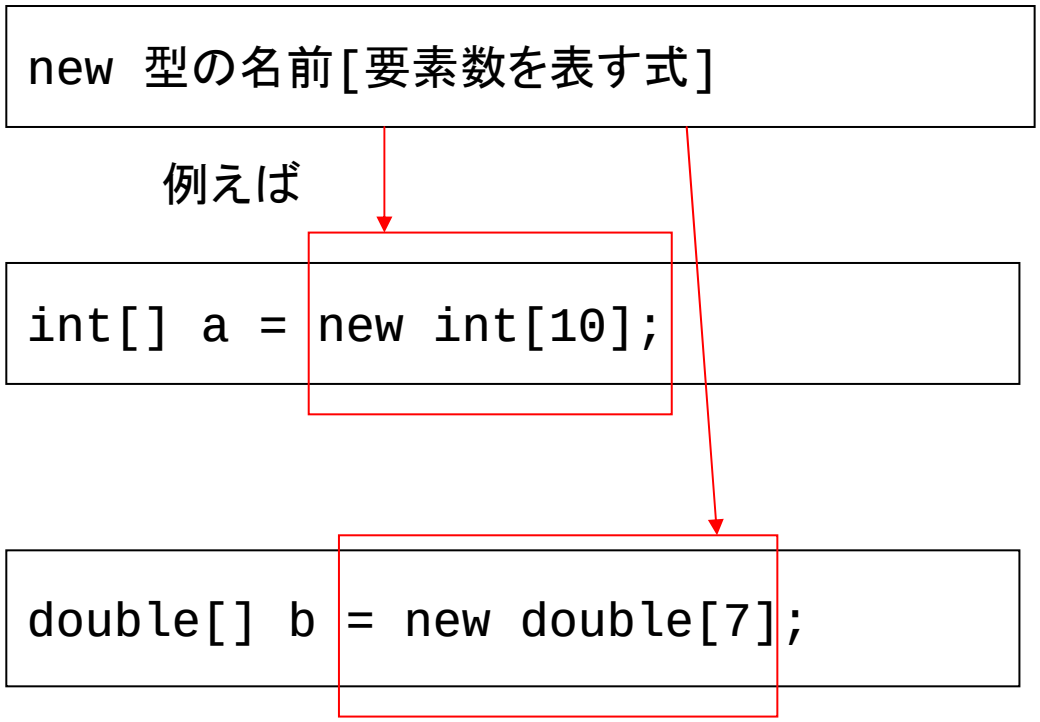
変数の宣言と一緒なので省略形で書いてよい。

```
double[] b = {1.0, 2.0};
```

初期値は定数でなく、式で与えてもよい

```
double[] c = { Math.sqrt(2.0), Math.sqrt(2.0) * 2, Math.sqrt(3.0) };
```

初期値を与えずに配列を生成する式



要素数の指定は変数でも式でもよい

```
BufferedReader reader =
  new BufferedReader(new InputStreamReader(System.in));
System.out.print("配列の大きさを入力:");
int size = Integer.parseInt(reader.readLine());
double[] c = new double[size];
```

配列の大きさは負になってはならない

配列要素の参照

```
int[] a = {1, 2, 3};
a[0] = 10;
int b = a[2];
```

b には 3 が格納された

```
int[] a = {1, 2, 3};
int b = 1;
int c = a[b];
```

c には 2 が格納された

```
int[] a = {1, 2, 3};
int c = a[a[0]];
```

c には 2 が格納された

配列の長さ

```
int[] a = {10, 20, 30};
for (int i = 0; i < a.length; i++) {
  System.out.println(a[i]);
}
```

実行結果

```
10
20
30
```

擬似コードで書くと

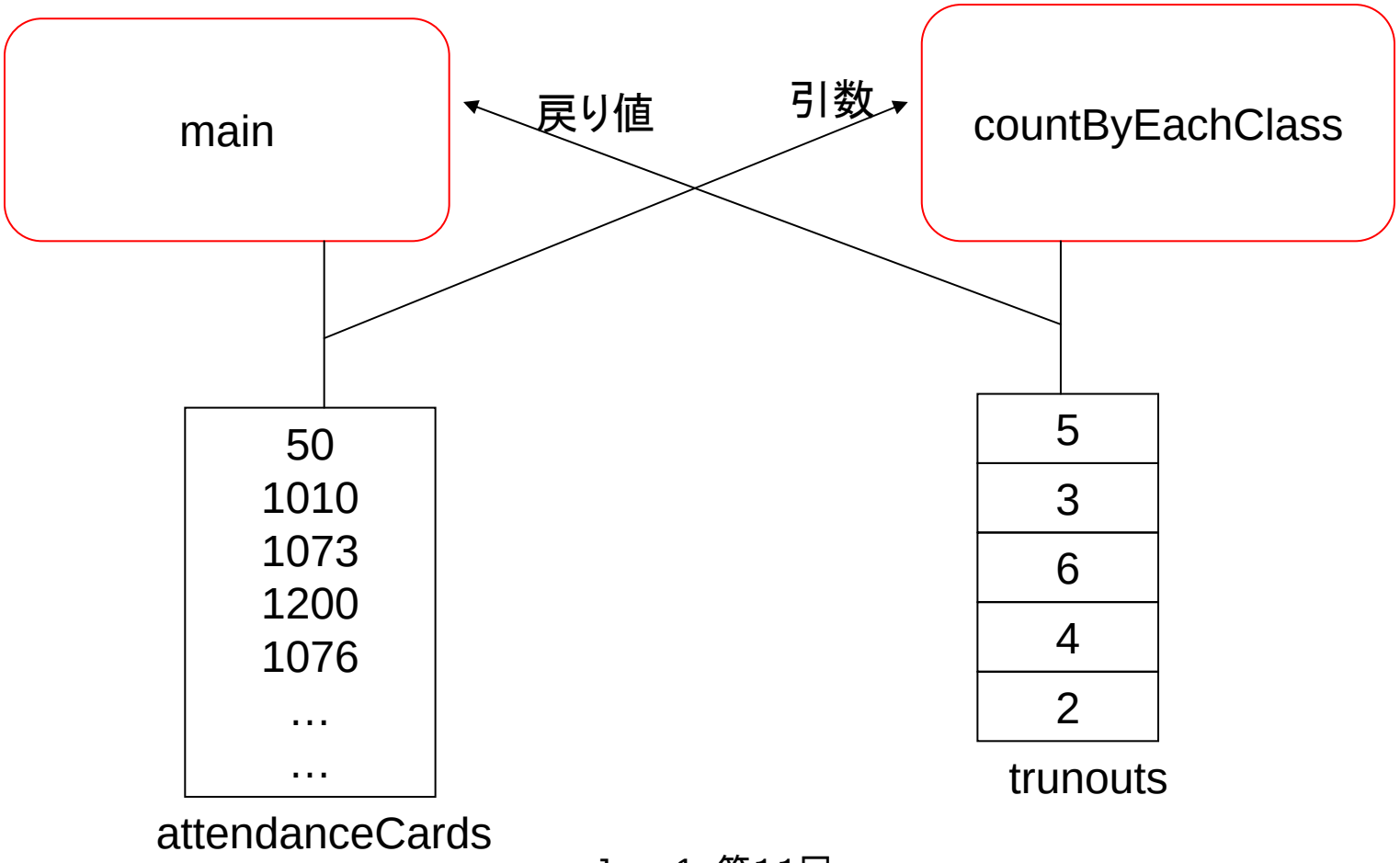
最後の要素のインデックスは

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

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「java20XX」の中の「test」というフォルダに「j1.lesson11.xml」という名前のファイルが作成される。
    - このファイルには今週使用するテスト一式が記述されている。
- j1.lesson11 というパッケージを作成する
- 演習資料にあるAttendance, ArrayReverse 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。

Attendance


main
戻り値
引数
countByEachClass
50
1010
1073
1200
1076
…
…
attendanceCards
5
3
6
4
2
trunouts

countByEachClassの解説

```
public static int[] countByEachClass(int[] attendanceCards) {
  int[] turnouts = new int[5];
  for (int i = 0; i < attendanceCards.length; i++) {
    switch (attendanceCards[i] / 100) {
      case  0:
        turnouts[0]++;
        break;
      case  1:
        turnouts[1]++;
        break;
      case 10:
        turnouts[2]++;
        break;
      case 11:
        turnouts[3]++;
        break;
      default:
        turnouts[4]++;
        break;
    }
  }
  return turnouts;
}
```

変数turnoutsには新しく生成された配列のIDが格納される

```
public static int[] countByEachClass(int[] attendanceCards) {
  int[] turnouts = new int[5];
  for (int i = 0; i < attendanceCards.length; i++) {
    switch (attendanceCards[i] / 100) {
      case  0:
        turnouts[0]++;
        break;
      case  1:
        turnouts[1]++;
        break;
      case 10:
        turnouts[2]++;
        break;
      case 11:
        turnouts[3]++;
        break;
      default:
        turnouts[4]++;
        break;
    }
  }
  return turnouts;
}
```

i番目の出席カードの下位2桁を削り落とす

```
public static int[] countByEachClass(int[] attendanceCards) {
  int[] turnouts = new int[5];
  for (int i = 0; i < attendanceCards.length; i++) {
    switch (attendanceCards[i] / 100) {
      case  0:
        turnouts[0]++;
        break;
      case  1:
        turnouts[1]++;
        break;
      case 10:
        turnouts[2]++;
        break;
      case 11:
        turnouts[3]++;
        break;
      default:
        turnouts[4]++;
        break;
    }
  }
  return turnouts;
}
```

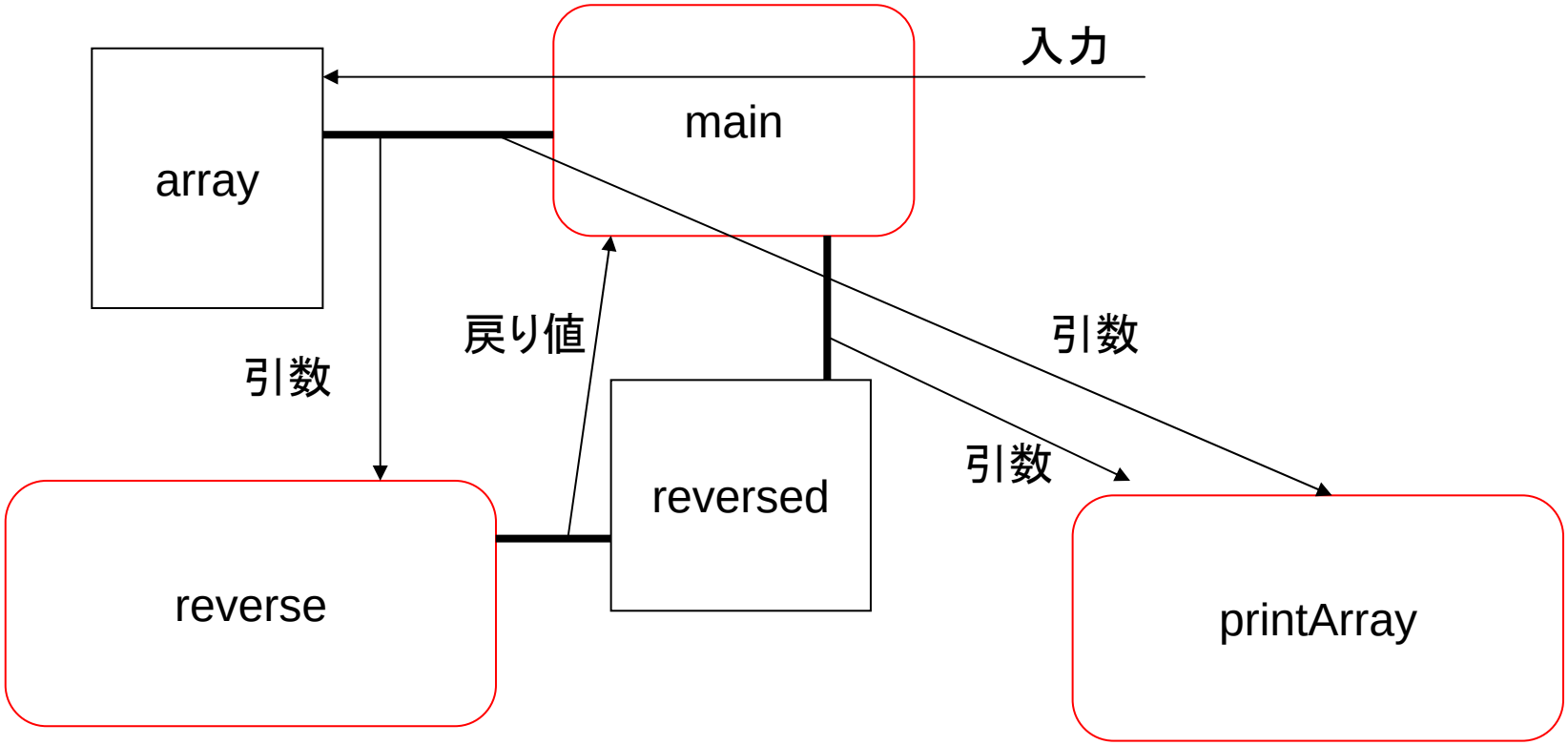
i番目の出席カードの上位2桁が

```
public static int[] countByEachClass(int[] attendanceCards) {
  int[] turnouts = new int[5];
  for (int i = 0; i < attendanceCards.length; i++) {
    switch (attendanceCards[i] / 100) {
      case  0:
        turnouts[0]++;
        break;
      case  1:
        turnouts[1]++;
        break;
      case 10:
        turnouts[2]++;
        break;
      case 11:
        turnouts[3]++;
        break;
      default:
        turnouts[4]++;
        break;
    }
  }
  return turnouts;
}
```

turnoutsに格納されている配列のIDを呼び出し側に返す

```
public static int[] countByEachClass(int[] attendanceCards) {
  int[] turnouts = new int[5];
  for (int i = 0; i < attendanceCards.length; i++) {
    switch (attendanceCards[i] / 100) {
      case  0:
        turnouts[0]++;
        break;
      case  1:
        turnouts[1]++;
        break;
      case 10:
        turnouts[2]++;
        break;
      case 11:
        turnouts[3]++;
        break;
      default:
        turnouts[4]++;
        break;
    }
  }
  return turnouts;
}
```

ArrayReverseの3つのメソッドの連携



ArrayReverseの解説

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
        newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

配列を生成する

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

配列の長さだけ入力を繰り返す

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

reverseが返す配列のIDを
reversedに受け取る

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

reverseをみてみよう

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

配列aの長さと同じ長さをもつ新しい配列を生成する

```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

対称の位置にある値をコピー

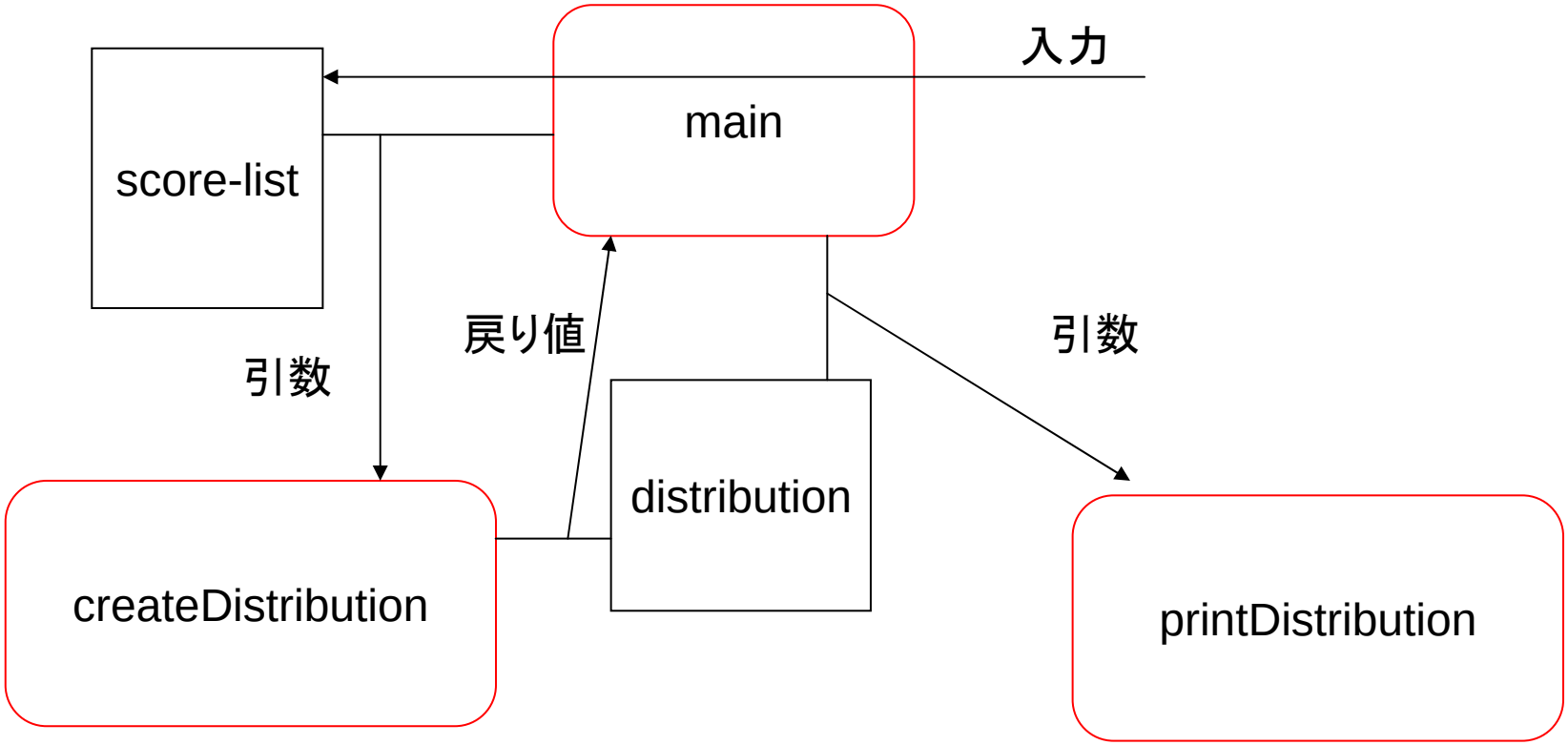
```
public class ArrayReverse {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    int[] array = new int[5];
    for (int i = 0; i < array.length; i++) {
      System.out.print("値を入力:");
      array[i] = Integer.parseInt(reader.readLine());
    }

    int[] reversed = reverse(array);
    printArray(array);
    System.out.println("");
    printArray(reversed);
  }

  public static int[] reverse(int[] a) {
    int[] newA = new int[a.length];
    for (int i = 0; i < newA.length; i++)
      newA[i] = a[newA.length - 1 - i];
    return newA;
  }

  public static void printArray(int[] a) { ...省略...}
}
```

結局、副作用はない

課題

各自のペースで今回の課題をやってください。

課題1101 Distributionの3つのメソッドの連携



```
===== 課題1101 createDistribution メソッド =======

createDistribution ( score-list )  // score-list は点数の配列
  得点分布表となる長さ 11 の配列 distribution をつくり 0 で初期化。

  for i を 0 から score-list の長さ - 1 まで
    もし score-list[i]が適正な値だったら // 0 以上 100 以下
      score-list[i] を 10 で割った結果(商)を変数 k に代入
      distribution[k] に 1 を加える

  distribution を返す

===== 課題1102 search メソッド =======

search ( values, target)  // values は整数の配列, target は整数

  for i を 0 から values の長さ - 1 まで
    もし values[i] と target が等しかったら
        i を返す  // i 番目に見つかった

  -1 を返す // 見つからなかった
```