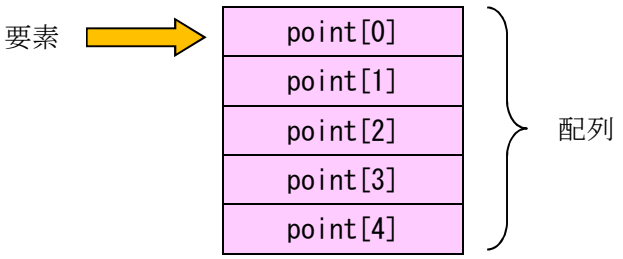
2018 年 5 月 24 日（木）実施

## 配列

同種のデータ型を有する複数のデータ（要素）を番号付けして，ひとまとまりの対象として扱うものを**配列**と呼ぶ。




配列の取り扱いに関して，次のような特徴がある。

1. プログラム中で用いる**配列変数**（配列の本体を参照する参照型の変数）は必ず**宣言**しておく。

   例）int[] point;　/* int 型の配列変数 point を宣言 */

2. 配列の本体を生成し，配列変数に代入して参照させる。

   例）point = new int[5];　/* int 型の 5 個の要素を持つ配列を生成 ⇒ 要素数は配列が生成された時点で定まる */

3. 配列変数を宣言する際の初期化で，配列の本体を生成することが出来る。

   例）int[] point =new int[5];

4. 配列の生成時に、各要素には初期値（int 型の場合は 0，char 型の場合は¥u0000，boolean 型の場合は false，参照型の場合は null）が設定される。

5. 配列変数を宣言する際に，中括弧{ }の間に初期値を与えて初期化することが出来る。

   例）int[] point = { 98, 76, 85, 67, 59 }; ⇒ この場合には，new で配列を生成する必要はない。

6. 配列の要素は 0 番から[要素数]－1 番までの添え字を用いて表される。

### char 型の配列

Java 言語では，char 型の**配列**は文字列の扱いとはならない。String クラスの変数を宣言して初期化するには次の様な書き方をすれば良かった。

例）　String str = "CUC";

char 型の配列を用いて上の例と同等な宣言を行うには

```
char[] charray = {'C', 'U', 'C'};
String str = new String(charray);
```

とする。

## メソッド

Java 言語のプログラムでは，ひとまとまりの処理をメソッドの単位で記述する。メソッドはあるクラスの機能として，そのクラスに属している。これまでのプログラムでは本体として最初に実行される main メソッドの他に，元々用意されている様々なクラスのメソッドが登場している。このことから分かる様に，main メソッド以外のメソッドは他のメソッドから呼び出されて実行される。

### メソッドの定義

メソッドの定義の一般形は次の様になる。

```
[ 修飾子 ] 戻り値のデータ型 メソッド名 ([引数指定-1 [, 引数指定-2・・・] ・・・]) {
    処理
    [ return 式; ]
  }
```

ここで，[ ]内は省略可能であり，修飾子がないメソッド，引数指定がないメソッド，return 文のないメソッドもある。戻り値のデータ型は，return 文で戻されるデータの型となり，return 文のない場合には **void** と表される。なお，引数指定とは，メソッド内で利用する仮引数をそのデータ型と共に指定するものである。

例 1）
```
public int wa (int x, int y) {
        return x+y;
    }
```

＊ 修飾子 public は，このメソッドがどのクラスからも利用出来ることを指定する。

例 2）
```
private void dispsum (int x, int y) {
        System.out.println(x +"と"+ y +"との和は"+ (x+y) +"です。");
    }
```

＊ 修飾子 private は，このメソッドが所属するクラス内でしか利用できないことを指定する。

### 例題１　（int 型のデータによる配列）

次のプログラムは，標準入力装置から入力された整数データの平均値を求め，表示させるものである。これを入力して，実行せよ。なお，データは 5 個分入力する必要がある。ここで，クラス名は Sample6_1，ソースファイル名は Sample6_1.java とする。

```
import java.util.Scanner;


public class Sample6_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM=5;
        int sum=0, i;
        double ave;
        int[] data = new int[NUM];
        Scanner sc = new Scanner(System.in);

        for (i=0; i<NUM; i++) {
            System.out.print((i+1) + "番目の整数データを入力してください： ");
            data[i] = sc.nextInt();

            sum = sum + data[i];
        }

        ave = (double)sum / NUM;

        for (i=0; i<NUM; i++)
            System.out.println(" "+(i+1) + "番目のデータ： " + data[i]);

        System.out.println(" データの合計：   " + sum);
        System.out.println(" データの平均値： " + ave);
    }

}
```

【解説】

1. 配列の要素の添え字は 0 番から始まるため，ループ変数 i は 0 で初期化している。
2. 配列の要素の添え字は[要素数]－1 番までなので，for 文の継続条件は i が NUM 未満となる。
3. System.out.print 及び System.out.println で(i+1)としているのは，i の初期値が 0 となっているが，画面表示では 1 番目から始めたいからであり，括弧で括っているのは，数値としての足し算を行った上で他の文字列と結合するためである。（括弧で括らないと，01 番目，11 番目の様に表示される。）

例題２　（String 型のデータによる配列）

次のプログラムは，標準入力装置から入力された氏名を配列によって格納し，編集して表示させるものである。これを入力して，実行せよ。なお，氏名は 5 人分入力する必要がある。ここで，クラス名は Sample6_2，ソースファイル名は Sample6_2.java とする。

```
import java.util.Scanner;


public class Sample6_2 {
```

```
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM=5;
        int i;
        String[] str = new String[NUM];
        Scanner sc = new Scanner(System.in);

        for (i=0; i<NUM; i++) {
            System.out.print((i+1) + "人目の氏名を入力してください： ");
            str[i] = sc.nextLine();
        }

        for (i=0; i<NUM; i++)
            System.out.println(" Name (No."+(i+1) + ") : " + str[i]);
    }

}
```

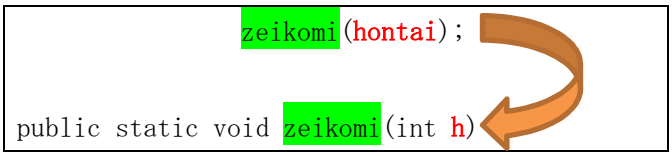
【解説】nextLine()は１行分の文字列を読み込む。これまで用いていた next()の場合は，文字列に空白やタブが含まれるとそれらを区切り記号とみなし，その前までの文字列を読み込む。

例題３（int 型のデータを引数とするメソッド）

次のプログラムは，標準入力装置から本体価格を円単位で入力すると，税込み価格を求めて結果を表示させるものである。これを入力して，実行せよ。ここで，クラス名は Sample6_3，ソースファイル名は Sample6_3.java とする。

```
import java.util.Scanner;


public class Sample6_3 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        int hontai;
        Scanner sc = new Scanner(System.in);

        System.out.print("本体価格を円単位で入力してください：");
        hontai = sc.nextInt();

        zeikomi(hontai);
    }

    public static void zeikomi(int h) {
        System.out.printf("税込み価格は%d 円です。%n", (int)(h*1.08));
    }
}
```

【解説】zeikomi メソッドの定義にある引数 h を**仮引数**，main メソッド内で zeikomi メソッドを呼び出しに用いる引数 hontai を実引数と呼ぶ。メソッドの呼び出しの際には**実引数**の値を仮引数に受け渡して用いる。

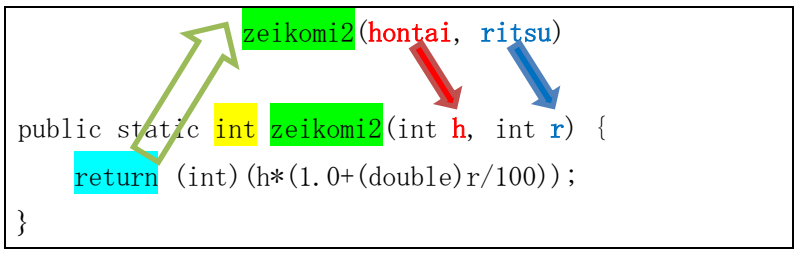



例題４（int 型の２つのデータを引数とし，int 型の値を返すメソッド）

次のプログラムは，例題３のプログラムを拡張して，税率を標準入力装置から入力出来る様にしたものである。これを入力して，税率を入力する際には 10 として実行せよ。ここで，クラス名は Sample6_4，ソースファイル名は Sample6_4.java とする。

```
import java.util.Scanner;


public class Sample6_4 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        int hontai, ritsu;
        Scanner sc = new Scanner(System.in);

        System.out.print("本体価格を円単位で入力してください：");
        hontai = sc.nextInt();

        System.out.print("税率を％単位で入力してください：");
        ritsu = sc.nextInt();

        System.out.printf("税込み価格は%d 円です。%n", zeikomi2(hontai, ritsu));
    }

    public static int zeikomi2(int h, int r) {
        return (int)(h*(1.0+(double)r/100));
    }

}
```

＊ ％（per cent または percent；パーセント）とは，100 のうちいくつを占めるかという比率であり，百分率と訳される。なお，per は／，cent は 100 を意味する。




【解説】main メソッド内で zeikomi2 メソッドを呼び出すと，２つの実引数 hontai，ritsu がそれぞれ zeikomi2 メソッドの仮引数 h，r に受け渡され，メソッド内で計算された税込み価格を return 文により，int 型の数値として呼び出した箇所に返す。

演習

次のプログラムは，標準入力装置から２つの整数を入力すると，合計を求めて結果を表示させるものである。このプログラムリストの空欄に適切な語句を埋めて完成させたプログラムを入力し，実行せよ。ここで，クラス名は Ex6，ソースプログラム名は Ex6.java とする。

```
import java.util.Scanner;


public class Ex6 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 2;
        int[] x = new int[NUM];

        for (int i=0; i<NUM; i++)
            x[i] = 1)    (i);

        System.out.printf("%d + %d => %d%n", x[0], x[1], sum(x[0], x[1]));
    }

    private static int inx(int j) {
        Scanner sc = new Scanner(System.in);

        System.out.print((j+1) +"つ目の整数を入力してください：");

        2)    sc.nextInt();
    }

    private static int sum(int a, int b) {
        3)    a+b;
    }

}
```

【解説】

1. main メソッド内で 1)　　メソッドを呼び出すと，標準入力装置から入力された整数が返され，x[i]に代入される。
2. main メソッド内で sum メソッドを呼び出すと，２つの実引数 x[0]，x[1]がそれぞれ sum メソッドの仮引数 a，b に受け渡され，合計が返される。

提出物：

1）例題 1，例題 2，例題 3，例題 4 及び演習のプログラムのコンソールへの出力結果をコピーして貼り付けたテキストファイル res6.txt をメールに添付する。
2）演習のソースプログラムのファイル Ex6.java をメールに添付する。
3）第５回の授業の復習の内容を埋めたファイル Review_5th.txt をメールに添付する。

＊ メールの件名は『プログラミング１　第６回課題』（鍵括弧は要らない）とする。