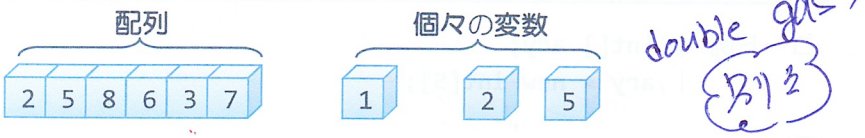
１０回目　配　　列

今日の講義で学ぶ内容

- 配列とその使い方
- 基本型変数と参照型変数
- 拡張 for 文

配　　列

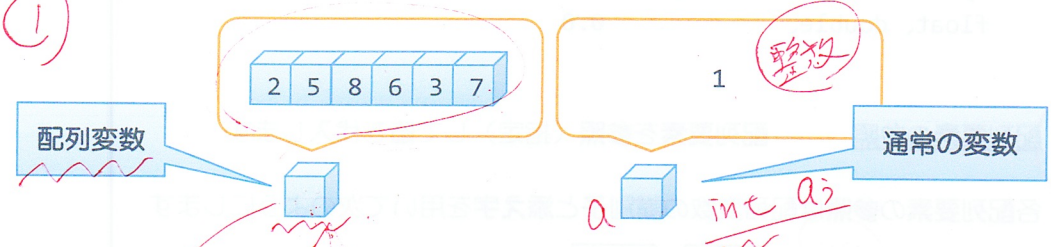
配　　列　　同じ型である複数の変数を一括して管理する機能です
直観的には、



配列の利用

配列の利用手順　　配列変数の宣言 → 配列要素の確保 → 配列要素の参照

配列変数の宣言　　配列変数とは配列を扱う（代入する）変数です



配列変数は通常の変数と同様に型と識別子をもちます

型と識別子を指定して次のように行います

```
型 識別子[];
```

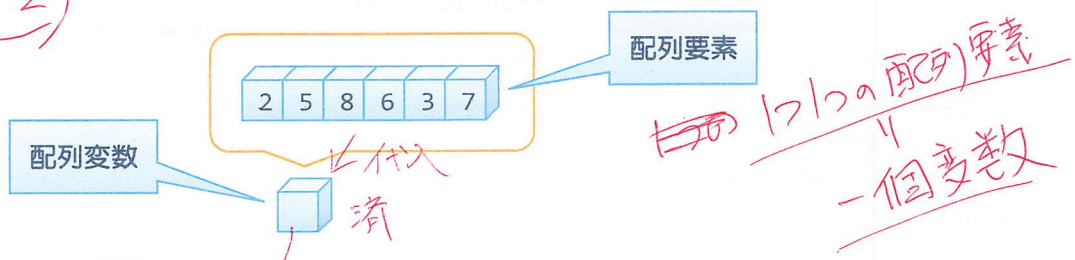
または

```
型[] 識別子;    // Java での標準のスタイル
```

コード例　`int[] ary;`

配列要素の確保

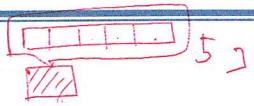
配列要素とは値を格納するための一連の領域です
各配列要素は一つの変数としての機能をもちます



型と配列要素の個数を指定して次のように行います

```
識別子 = new 型 [配列要素の個数];
```

コード例
```
int[] ary;
ary = new int[5];
```



new 演算子は指定された個数の配列要素をコンピュータのメモリ上に確保します

配列要素は確保されたとき予め以下の値が代入されます

（型）	（デフォルト値）
boolean	false
char	0（¥u0000）
byte、short、int、long	0
float、double、	0.0

配列要素の参照

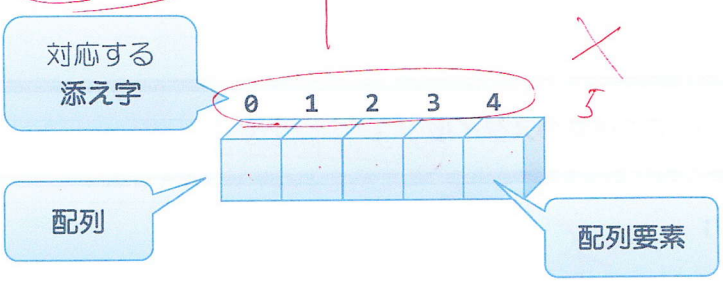
配列要素を参照（指定）して値を代入します

各配列要素の参照は配列変数の識別子と添え字を用いて次のようにします

```
識別子 [添え字]
```

添え字には個々の配列要素の位置を表す 0 以上の整数を指定します
- 最初の配列要素を参照するには、添え字に 0 を指定します
- 最後の配列要素を参照するには、添え字に 配列要素の個数-1 を指定します

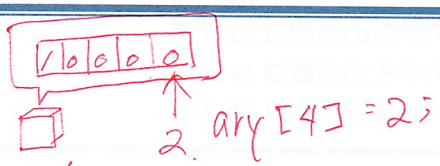
例えば、5 個の配列要素を持つ配列の場合は次のように 0 から 4 までの添え字を指定します



配列要素への値の代入は、各配列要素を参照して次のように行います

識別子 [添え字] = 値;

コード例
```
int[] ary;
ary = new int[5];
ary[0] = 1;
```

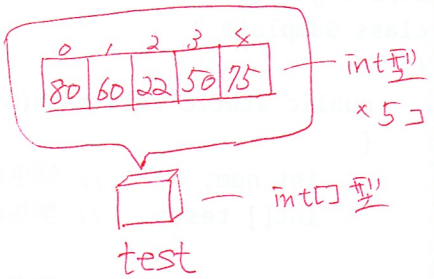


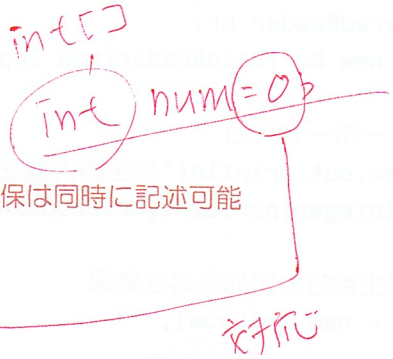
添え字に指定する値は int 型である必要があります

ソースコード例

ソースファイル名：Sample10_1.java

```
// 配列を用いて５人の学生の点数を管理する
class Sample10_1
{
    public static void main(String[] args)
    {
        int i;

        // 配列変数の宣言
        int test[];
        // int[] test; とも記述可能

        // 配列要素の確保
        test = new int[5];

        // 配列変数の宣言と配列要素の確保は同時に記述可能
        // int test[] = new int[5];
        // int[] test = new int[5];

        //各配列要素へ値を代入
        test[0]=80;
        test[1]=60;
        test[2]=22;
        test[3]=50;
        test[4]=75;

        // 各配列要素（添え字は０から４まで）を順番に出力
        for(i=0; i<5; i++)
            System.out.println(i+"番目の学生の点数は"+test[i]+"です。");
    }
}
```





実行画面

```
0番目の学生の点数は80です。
1番目の学生の点数は60です。
2番目の学生の点数は22です。
3番目の学生の点数は50です。
4番目の学生の点数は75です。
```

ソースコード例

ソースファイル名：Sample10_2.java

```
// 配列要素の動的な確保
import java.io.*;
class Sample10_2
{
    public static void main(String[] args) throws IOException
    {
        int num;        // 学生数
        int[] test;     // 学生の点数を保存する配列変数

        // キーボード入力の準備
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        // キーボード入力
        System.out.println("学生の人数を入力してください。");
        num=Integer.parseInt(br.readLine());

        // 学生数分の配列要素を確保
        test = new int[num];

        // キーボードから点数を配列要素へ順番に入力する
        for(int i=0;i<num;i++)
            test[i]= Integer.parseInt(br.readLine());

        // 配列要素に入力されている点数を順番に出力する
        for(int i=0;i<num;i++)
            System.out.println(i+"番目の学生の点数は"+test[i]+"です。");
    }
}
```

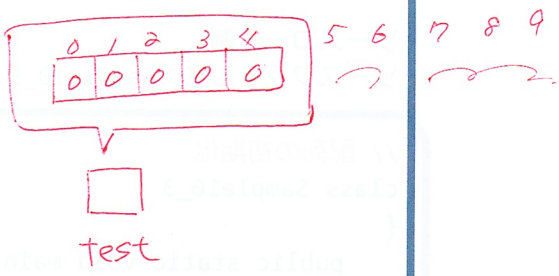
実行画面

```
学生の人数を入力してください。
3
89
75
95
0 番目の学生の点数は 89 です。
1 番目の学生の点数は 75 です。
2 番目の学生の点数は 95 です。
```

配列要素の確保されていない領域へアクセスしたら?

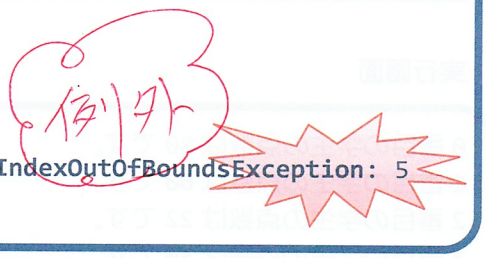
ソースファイル名：Ext10_1.java

```
// 添え字の範囲のミス
class Ext10_1
{
    public static void main(String[] args)
    {
        int i;

        // 配列変数の宣言と 5 個の配列要素の確保
        int[] test = new int[5];

        // test[4]より上の配列要素へアクセス
        for(i=0; i<10; i++)
            System.out.println(i+"番目の配列要素は"+test[i]+"です。");
    }
}
```



実行画面

```
0 番目の配列要素は 0 です。
1 番目の配列要素は 0 です。
2 番目の配列要素は 0 です。
3 番目の配列要素は 0 です。
4 番目の配列要素は 0 です。
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at Ext10_1.main(Ext10_1.java:13)
```



配列の初期化

配列の初期化　　指定された値の列が代入された配列要素をもつ配列変数を宣言します

配列の初期化は、配列変数の宣言時に次のように行います

```
型 識別子[] = {値1, 値2, 値3,…, 値n};
```

または

```
型[] 識別子 = {値1, 値2, 値3,…, 値n};    // Java での標準のスタイル
```

コード例 | `int[] ary={2,5,8,6,3,7};`

配列の初期化により、**配列要素の確保**や個々の**値の代入**を**省く**ことができます

ソースコード例

ソースファイル名：Sample10_3.java

```
// 配列の初期化
class Sample10_3
{
    public static void main(String[] args)
    {
        // 配列の初期化
        int test[]={80,60,22,50,75};
        // int[] test={80,60,22,50,75}; とも記述可能

        // 配列要素を出力
        for(int i=0;i<5;i++)
            System.out.println(i+"番目の学生の点数は"+test[i]+"です。");
    }
}
```

実行画面

```
0番目の学生の点数は80です。
1番目の学生の点数は60です。
2番目の学生の点数は22です。
3番目の学生の点数は50です。
4番目の学生の点数は75です。
```

配列の初期化では配列要素の数を指定しません
配列要素の数は値の列より自動的に計算されます

匿名配列　　**配列変数**を用いない**配列**のことです
通常、配列は配列要素を配列変数に代入して用いますが
匿名配列は配列要素をそのまま用います

匿名配列は、次のようにして作成します

new 型[] {値１, 値２, 値３,…, 値ｎ}

指定された値の列が代入された**配列要素**が**確保**されます

匿名配列は、次のように配列変数に代入してこれまでと同じように使用できます

```
int[] ary;
ary = new int[]{2,5,8,6,3,7};
```

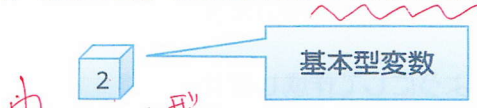
また、メソッド（Java プログラミング 2 で解説）に引数として渡すときよく利用されます

```
obj.method1(new int[]{2,5,8,6,3,7});
```

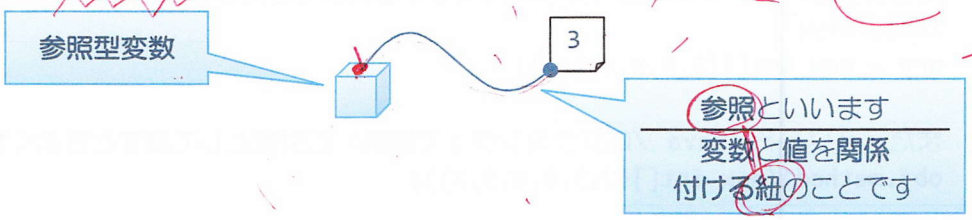
配列変数と参照型変数

変数の種類　　基本型変数と参照型変数の 2 種類があります

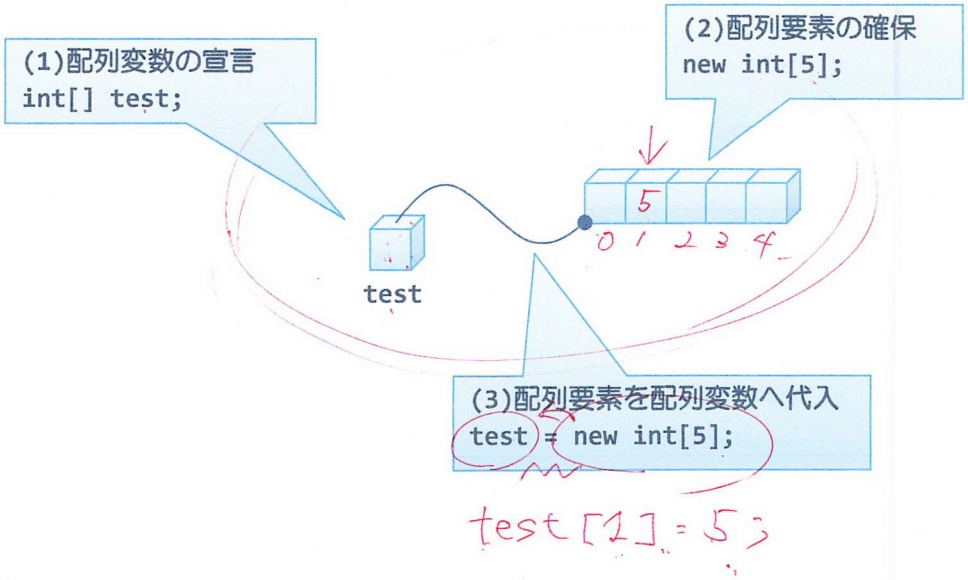
基本型変数　　"値"そのものを代入できます
boolean 型、char 型、byte 型、short 型、
int 型、long 型、float 型、double 型の 8 種類の変数



参照型変数　　"値のある場所"を代入できます
配列変数、クラス型変数のみ



例題 **Sample10_1** での配列変数の宣言と配列要素の確保は次のように理解できます



ソースコード例

ソースファイル名：Sample10_4.java

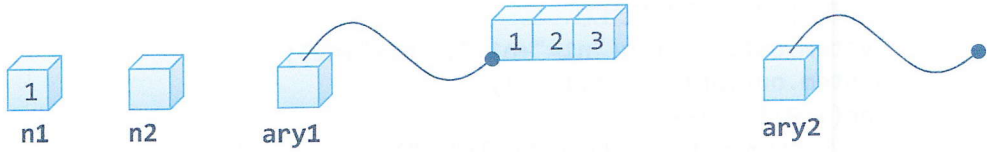
```
// 配列変数へ代入するということは?
class Sample10_4
{
    public static void main(String[] args)
    {
        int i;
        int n1=1, n2;
        int[] ary1={1,2,3}, ary2;

        // int 型変数へ代入
        n2=n1;
        // 配列変数へ代入
        ary2=ary1;

        // 変数と配列要素を出力
        System.out.println("n1="+n1+",  n2="+n2);
        System.out.print("ary1={");
        for(i=0;i<3;i++)
            System.out.print(ary1[i]+"  ");
        System.out.println("}");
        System.out.print("ary2={");
        for(i=0;i<3;i++)
            System.out.print(ary2[i]+"  ");
        System.out.println("}");

        // 一方の int 型変数の値を変更
        System.out.println("n2 := 2;");
        n2=2;
        // 一方の配列要素の値を変更
        System.out.println("ary2[1] := 4;");
        ary2[1]=4;

        // 変数と配列要素を出力
        System.out.println("n1="+n1+",  n2="+n2);
        System.out.print("ary1={");
        for(i=0;i<3;i++)
            System.out.print(ary1[i]+"  ");
        System.out.println("}");
        System.out.print("ary2={");
        for(i=0;i<3;i++)
            System.out.print(ary2[i]+"  ");
        System.out.println("}");
    }
}
```

実行画面

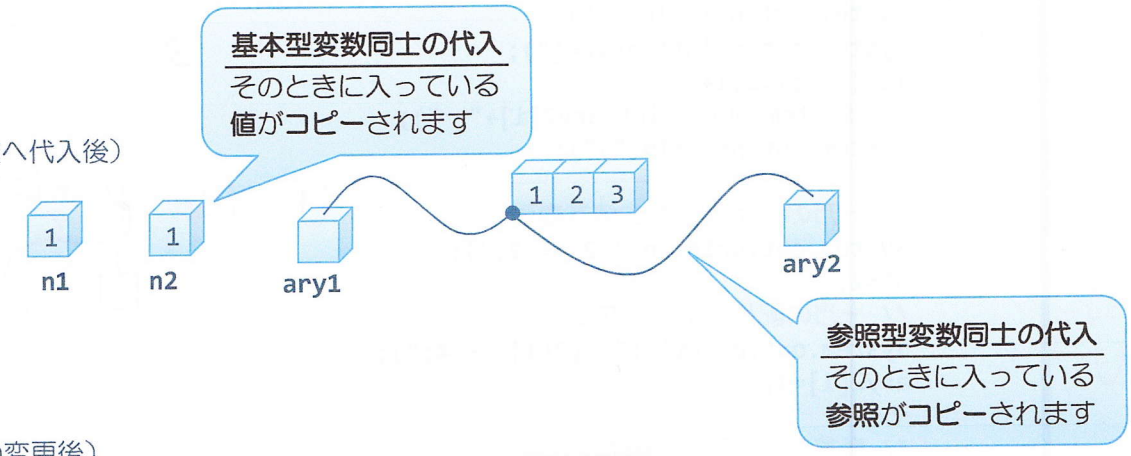
```
n1=1,  n2=1
ary1={1  2  3  }
ary2={1  2  3  }
n2 := 2;
ary2[1] := 4;
n1=1,  n2=2
ary1={1  4  3  }
ary2={1  4  3  }
```

例題 **Sample10_4** の変数の振る舞いは次のように図的に理解できます

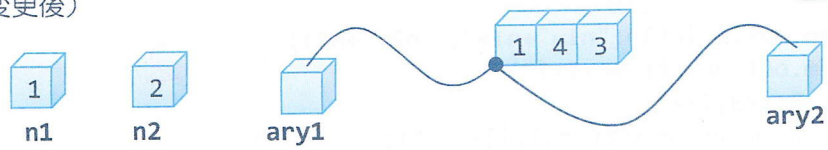
（変数の宣言時）



（変数へ代入後）



（値の変更後）



変数同士の代入では、代入する側の**変数に入っているデータが**代入される側の変数に**コピーされます**

基本型変数の場合には中に入っている**値**がコピーされ、**参照型変数**の場合には中に入っている**参照**がコピーされます

拡張 for 文

Java では配列を処理するための特別な for 文が準備されています

拡張 for 文　　**配列**に格納されている**値**が**順番に**取り出されます
取り出された値が変数に代入され文が処理されて、繰り返します

```
for(型 変数名:配列変数名) 文
```

コード例 ｜ `int[] ary={2,5,8,6,3,7};`
　　｜ `for(int i:ary)System.out.println(i);`

配列の終端に到達するまで for 文は繰り返されます
1 回目の繰り返し：添え字 0 の配列要素の値が変数に代入されます
2 回目の繰り返し：添え字 1 の配列要素の値が変数に代入されます
3 回目の繰り返し：添え字 2 の配列要素の値が変数に代入されます
　　⋮

ソースコード例
ソースファイル名：Ext10_2.java

```
// エンハンスド for 文
class Ext10_2
{
    public static void main(String[] args)
    {
        // 配列の初期化
        int[] test={80,60,22,50,75};

        // 配列要素を出力
        for(int num:test)
            System.out.println(num);
    }
}
```

実行画面

```
80
60
22
50
75
```

今日の講義のまとめ

・配列を用いることにより、同じ型の複数の変数を一括して管理できます。

・配列を利用するには、まず配列変数を宣言します。次に、個数を指定して配列要素を確保します。そして、各配列要素に値を代入します。

・配列の初期化を用いると、値を列挙するだけでその値が代入された配列要素が確保され、配列変数へ代入されます。

・配列の初期化を用いる場合には、配列要素の個数は明示する必要はありません。

・変数には、基本型変数と参照型変数があります。基本型変数には、値そのものを代入できます。参照型変数には、参照（値のある場所）を代入できます。

・拡張 for 文は、配列を用いた処理に適した繰返し文です。