プログラミング

- 第3回
  - メソッド（第8章）
  - 配列（第9章）

プログラミング

- 第３回
  - メソッド（第８章）
  - 配列（第９章）

メソッド（第8章）

- 例

```
System.out.println(“Hello”) ;
line = reader.readLine() ;
n = Integer.parseInt(“100”) ;
…
```

- メソッド
  - メソッドの宣言と呼び出し
  - メソッドの引数と戻り値

メソッド（第8章）

例題1 (Discount1.java)

```
public class Discount1 {
    public static void main(String[] args) {
        int p, q;
        p = 10000;
        q = p / 2;
        System.out.println("元の値段が" + p + "円なら、半額だと" + q + "円になります。");
    }
}
```

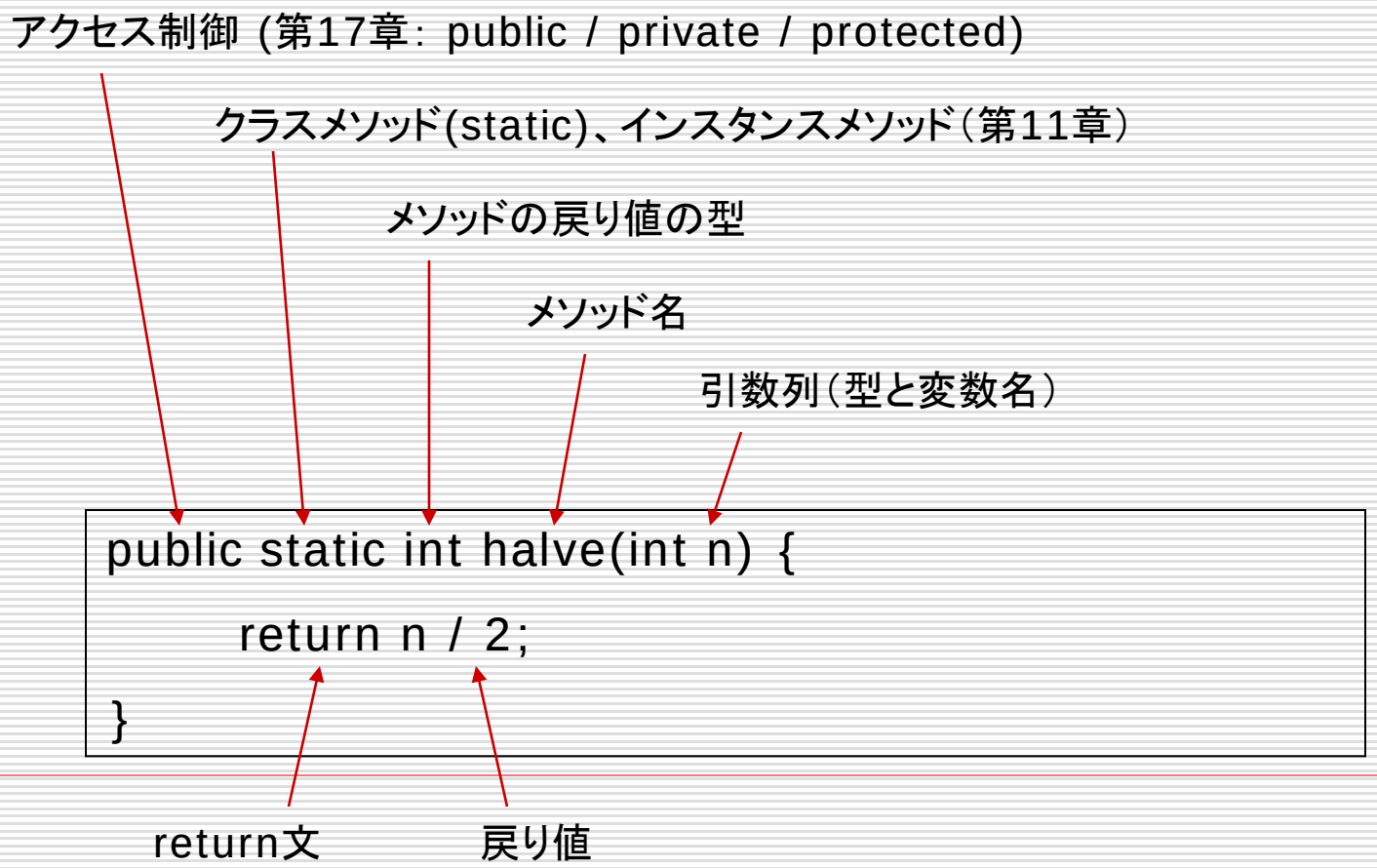
メソッド（第8章）

例題2 (Discount2.java)

```
public class Discount2 {
    public static void main(String[] args) {          // mainメソッド
        int p, q;
        p = 10000;
        q = halve(p);
        System.out.println("元の値段が" + p + "円なら、半額だと" + q + "円になります。");
    }
    public static int halve(int n) {                  // halveメソッド
        return n / 2;
    }
}
```

メソッド（第8章）

メソッド



メソッド（第8章）

例題3 (Graph1.java)

```
public class Graph1 {
    public static void main(String[] args) {          // mainメソッド
        printGraph(10);
    }
    public static void printGraph(int x) {            // printGraphメソッド
        for (int i = 0; i < x; i++) {
            System.out.print("*");
        }
        System.out.println("");
    }
}
```

戻り値が無い場合

局所変数 i のスコープ

return文なし（戻り値なし）

メソッド（第8章）

例題4 (Power1.java)

```
public class Power1 {
    public static void main(String[] args) {          // mainメソッド
        System.out.println(getPower(8, 2));
    }
    /* x の n 乗の計算 */
    public static int getPower(int x, int n) {        // getPowerメソッド
        int y = 1;
        for (int i = 0; i < n; i++) {
            y = y * x;
        }
        return y;
    }
}
```

引数列（型と変数名）

メソッド（第8章）

- Systemクラス
  - APIリファレンス

フィールドの概要	
static PrintStream	err 「標準」エラー出力ストリームです。
static InputStream	in 「標準」入力ストリームです。
static PrintStream	out 「標準」出力ストリームです。

http://java.sun.com/javase/ja/6/docs/ja/api/

プログラミング

- 第3回
  - メソッド（第8章）
  - **配列（第9章）**

配列（第９章）

- 配列とは
  - 変数に番号をつけて並べたもの
  - 「参照型」のひとつ

配列（第9章）

例題1 (Heikin1.java)

```
public class Heikin1 {
    public static void main(String[] args) {
        int kokugo, suugaku, eigo;
        double heikin;

        kokugo = 63;
        suugaku = 90;
        eigo = 75;
        heikin = (kokugo + suugaku + eigo) / 3.0;

        System.out.println("国語は" + kokugo + "点");
        System.out.println("数学は" + suugaku + "点");
        System.out.println("英語は" + eigo + "点");
        System.out.println("平均点は" + heikin + "点");
    }
}
```

変数の宣言

代入

参照

配列（第9章）

例題2 (Heikin2.java)

```
public class Heikin2 {
    public static void main(String[] args) {
        int[] ten;
        double heikin;

        ten = new int[3];
        ten[0] = 63;
        ten[1] = 90;
        ten[2] = 75;
        heikin = (ten[0] + ten[1] + ten[2]) / 3.0;

        System.out.println("国語は" + ten[0] + "点");
        System.out.println("数学は" + ten[1] + "点");
        System.out.println("英語は" + ten[2] + "点");
        System.out.println("平均点は" + heikin + "点");
    }
}
```

宣言

確保、代入

参照

配列（第９章）

- 配列の宣言
  - 配列用の変数の宣言

```
int[] ten;
```

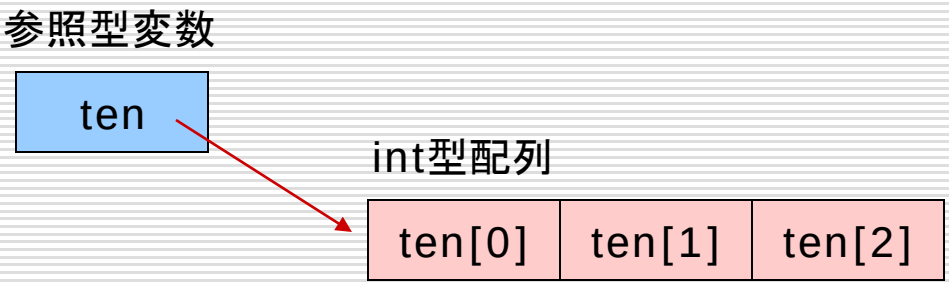
参照型変数

ten

配列（第９章）

- 配列の確保
  - new 型名 [要素の個数] ;
  - int型変数3個分の領域をメモリ上に確保

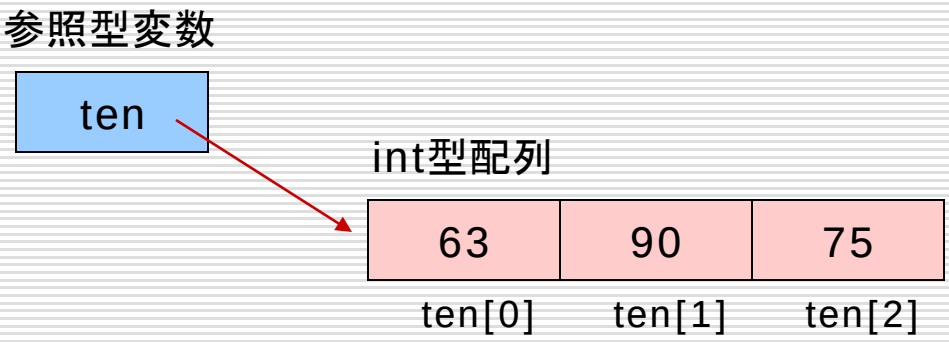
```
ten = new int[3];
```



配列（第９章）

- 配列要素への代入
  - 配列要素は0からカウント (C言語と同じ)

```
ten[0] = 63;
ten[1] = 90;
ten[2] = 75;
```

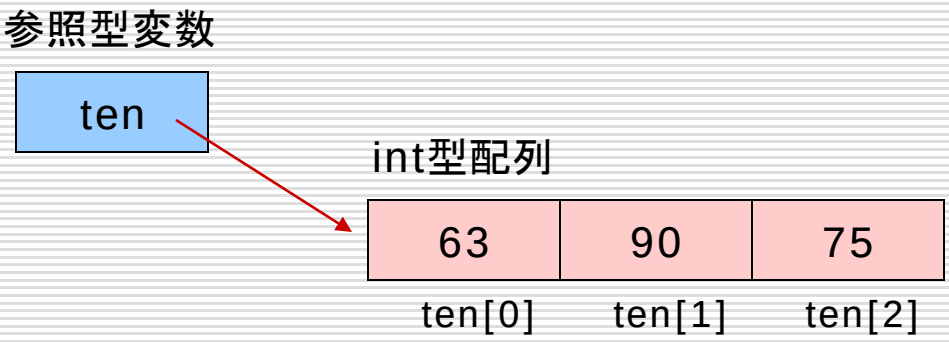


配列（第９章）

配列要素の参照

- 配列名[添字]

```
heikin = (ten[0] + ten[1] + ten[2]) / 3.0;
System.out.println("国語は" + ten[0] + "点");
```



配列（第９章）

- 配列の使い方
  - 宣言　　int[] ten;
  - 確保　　ten = new int[3];
  - 代入　　ten[0] = 63;
  - 参照　　System.out.println(ten[0]);

配列（第9章）

例題3 (Heikin3.java)

```
public class Heikin3 {
   public static void main(String[] args) {
      int[] ten;
      int sum;
      double heikin;

      ten = new int[3];
      ten[0] = 63;
      ten[1] = 90;
      ten[2] = 75;
      sum = 0;
      for (int i = 0; i < 3; i++) {
         sum = sum + ten[i];
      }
      heikin = sum / 3.0;

      System.out.println("国語は" + ten[0] + "点");
      System.out.println("数学は" + ten[1] + "点");
      System.out.println("英語は" + ten[2] + "点");
      System.out.println("平均点は" + heikin + "点");
   }
}
```

宣言

確保、代入

参照

配列（第９章）

- 変数による配列要素の参照

```
sum = 0 ;
for (int i = 0; i < 3; i++) {
        sum = sum + ten[i];
 }
```

```
sum = sum + ten[0] ;
sum = sum + ten[1] ;
sum = sum + ten[2] ;
```

```
sum = ten[0] + ten[1] + ten[2] ;
```

同じ

配列（第9章）

例題4 (Heikin4.java)

```
public class Heikin4 {
   public static void main(String[] args) {
      int[] ten;
      int sum;
      double heikin;

      ten = new int[5];
      ten[0] = 63;
      ten[1] = 90;
      ten[2] = 75;
      ten[3] = 45;
      ten[4] = 81;
      sum = 0;
      for (int i = 0; i < ten.length; i++) {
         sum = sum + ten[i];
      }
      heikin = (double)sum / ten.length;

      System.out.println("国語は" + ten[0] + "点");
      System.out.println("数学は" + ten[1] + "点");
      System.out.println("英語は" + ten[2] + "点");
      System.out.println("理科は" + ten[3] + "点");
      System.out.println("社会は" + ten[4] + "点");
      System.out.println("平均点は" + heikin + "点");
   }
}
```

宣言

確保、代入

参照

配列（第９章）

- 配列の長さ
  - 配列名.length

```
sum = 0;
for (int i = 0; i < ten.length; i++) {
    sum = sum + ten[i];
}
heikin = (double)sum / ten.length;
```

キャスト演算子（型変換）

配列（第9章）

□ 例題5 (Heikin5.java)

```
public class Heikin5 {
    public static void main(String[] args) {
        int[] ten = { 63, 90, 75, 45, 81 };
        int sum = 0;

        for (int i = 0; i < ten.length; i++) {
            sum += ten[i];
        }
        double heikin = (double)sum / ten.length;

        System.out.println("国語は" + ten[0] + "点");
        System.out.println("数学は" + ten[1] + "点");
        System.out.println("英語は" + ten[2] + "点");
        System.out.println("理科は" + ten[3] + "点");
        System.out.println("社会は" + ten[4] + "点");
        System.out.println("平均点は" + heikin + "点");
    }
}
```

宣言、初期化（兼確保、代入）

参照

配列（第９章）

配列の初期化

配列の宣言＆初期化（確保＆代入）

```
int[] ten = { 63, 90, 75, 45, 81 };
```

```
int[] ten;                  // 宣言
ten = new int[5];           // 確保
ten[0] = 63;                // 代入
ten[1] = 90;
ten[2] = 75;
ten[3] = 45;
ten[4] = 81;
```

同じ

配列（第9章）

例題6 (Heikin6.java)

```
public class Heikin6 {
    public static void main(String[] args) {
        int[][] tens = {
            {  63,  90,  75,  45,  81 },
            {  85, 100,  95,  80,  90 },
            { 100, 100, 100, 100, 100 },
        };
        for (int i = 0; i < tens.length; i++) {
            int sum = 0;
            for (int j = 0; j < tens[i].length; j++) {
                System.out.print("¥t" + tens[i][j]);
                sum += tens[i][j];
            }
            System.out.println("¥t| " + (double)sum / tens[i].length);
        }
    }
}
```

宣言、初期化

参照

配列（第９章）

- 二次元配列
  - 二次元配列の宣言

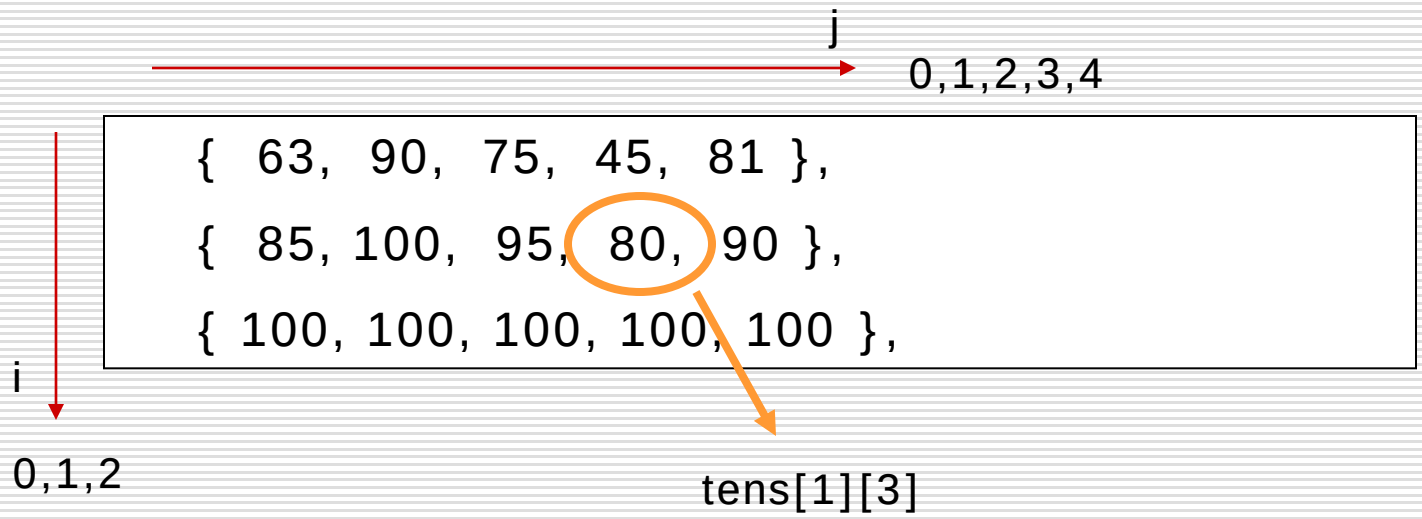
```
int[][] tens;
```

  - 二次元配列の宣言＆初期化（確保＆代入）

```
int[][] tens = {
    {  63,  90,  75,  45,  81 },
    {  85, 100,  95,  80,  90 },
    { 100, 100, 100, 100, 100 },
};
```

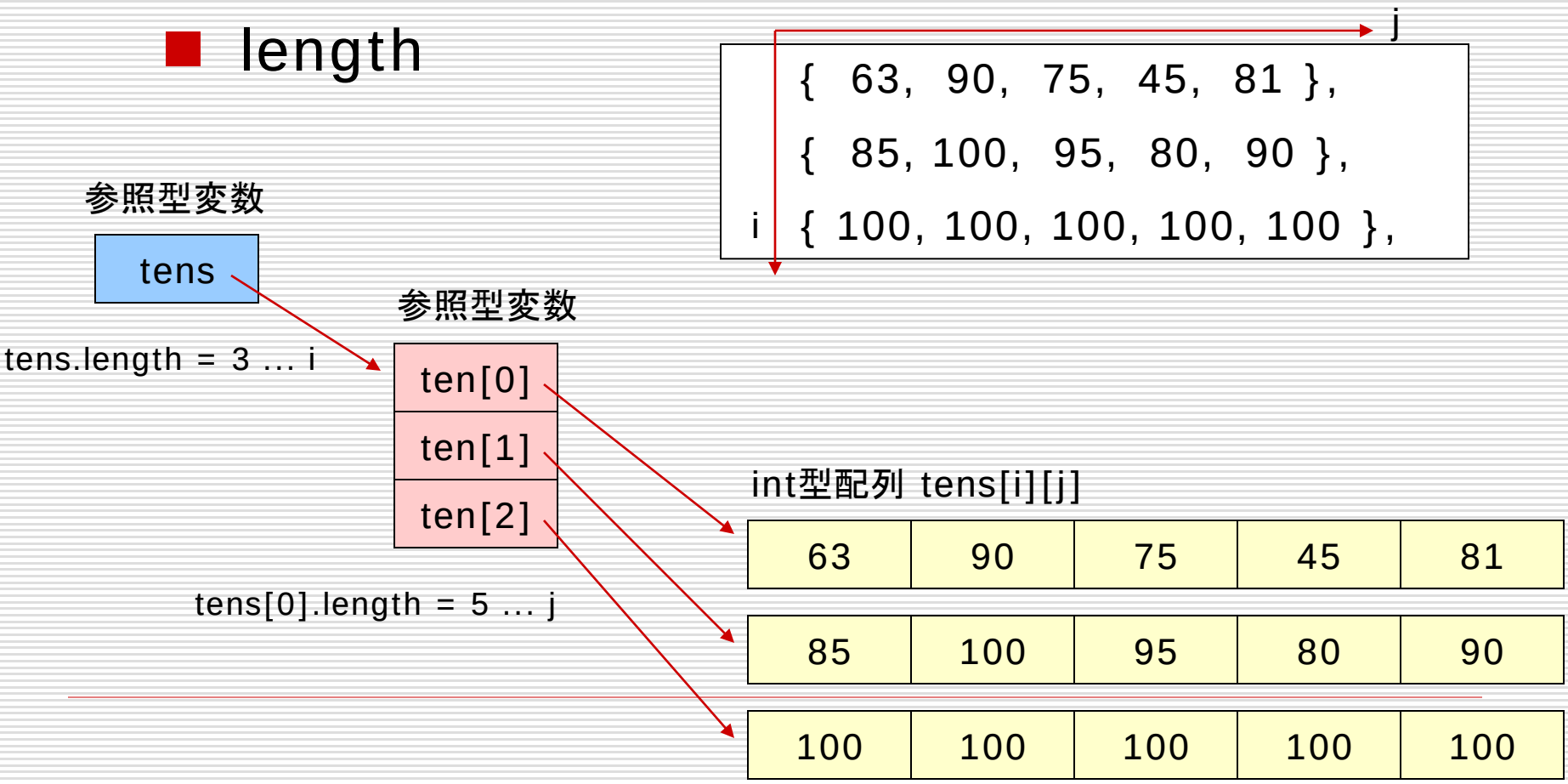
配列（第９章）

- 二次元配列
  - 二次元配列の参照: tens[i][j]



配列（第9章）

- 二次元配列
  - length



配列（第９章）

- 二次元配列
  - 要素数は異なってもよい

```
int[][] arr = {
    {   3,  14,  159,  26 },
    {  53,   5 },
    { 897, 93, 238 },
};
```

配列（第9章）

例題7 (ShowArgs.java)

```
public class ShowArgs {
    public static void main(String[] args) {
        System.out.println("args.length の値は " + args.length);
        for (int i = 0; i < args.length; i++) {
            System.out.println("args[" + i + "] の値は " + args[i] + " です。");
        }
    }
}
```

配列（第９章）

コマンド引数

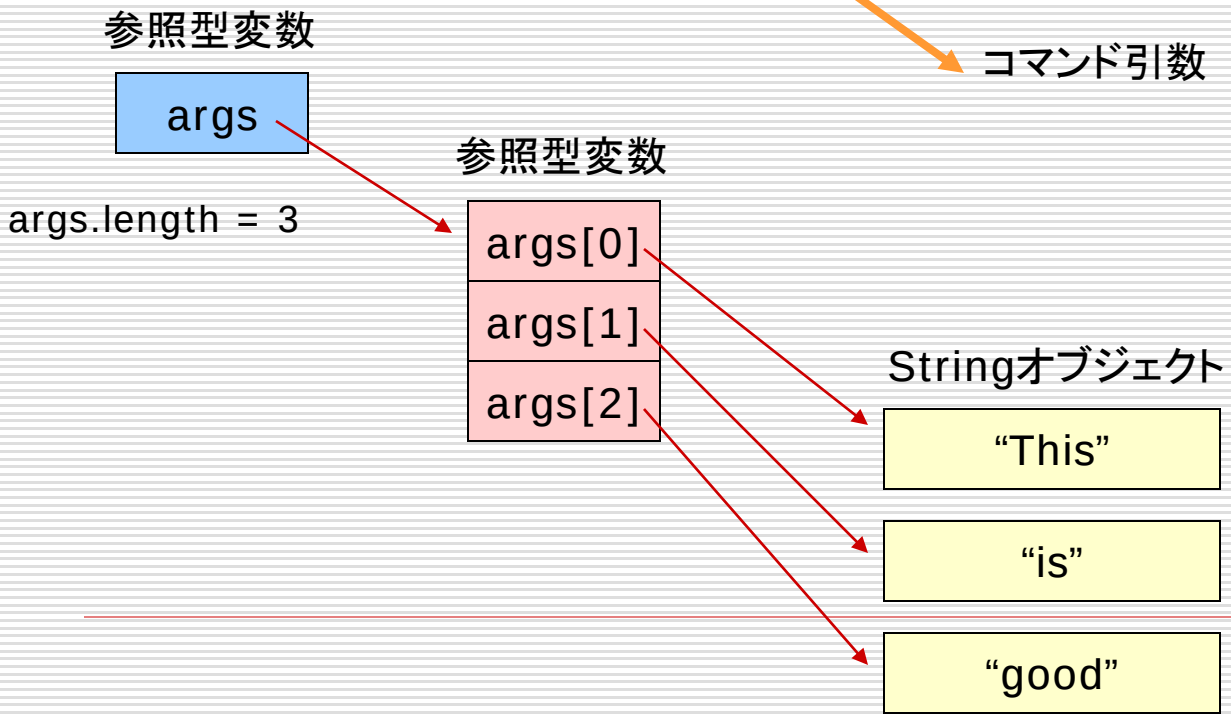
```
public class ShowArgs {
    public static void main(String[] args) {
        System.out.println("args.length の値は " + args.length);
        for (int i = 0; i < args.length; i++) {
            System.out.println("args[" + i + "] の値は " + args[i] + " です。");
        }
    }
}
```

- args：文字列配列(引数列)
- args.length：引数の個数

配列（第９章）

□ コマンド引数

```
public static void main(String[] args) { ... }
> java ShowArgs This is good
```



配列（第９章）

- 例題8(1) (Kamoku.java)

クラスの宣言

```
public class Kamoku {
    String namae;     // 科目名
    int tensuu;       // 点数

    // 科目の作成
    public Kamoku(String namae, int tensuu) {
        this.namae = namae;
        this.tensuu = tensuu;
    }

    // 科目の文字列表現
    public String toString() {
        return namae + "は" + tensuu + "点";
    }
}
```

フィールド

コンストラクタ

メソッド

配列（第9章）

☐ 例題8(2) (Heikin7.java)

インスタンス作成

```
public class Heikin7 {
    public static void main(String[] args) {
        Kamoku[] kamoku = {
            new Kamoku("国語", 63),
            new Kamoku("数学", 90),
            new Kamoku("英語", 75),
            new Kamoku("理科", 45),
            new Kamoku("社会", 81),
        };
        int sum = 0;
        for (int i = 0; i < kamoku.length; i++) {
            System.out.println(kamoku[i]);
            sum += kamoku[i].tensuu;
        }
        double heikin = (double)sum / kamoku.length;
        System.out.println("平均点は" + heikin + "点");
    }
}
```

インスタンスの作成＆初期化

コンストラクタ実行

配列（第９章）

- toString メソッド
  - System.out.println(インスタンス)で文字列表示させるためのメソッド

```
System.out.println(kamoku[i]);
```

```
public class Kamoku {
    ...
    public String toString() {
        return namae + "は" + tensuu + "点";
    }
}
```

配列（第９章）

- メモリの確保
  - 「newを使ってメモリを確保する」

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