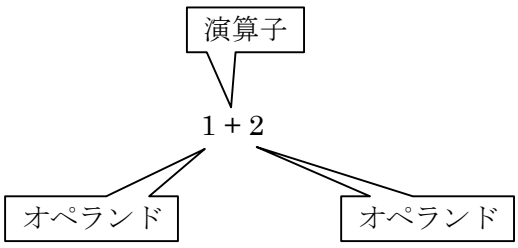
Ｊａｖａプログラミング Ｉ

# ４回目　演　算　子

## ●　演算子とオペランド

| | | |
|---|---|---|
| 演算子 | 演算の種類 | 例えば、＋、－、＊、／など |
| オペランド | 演算の対象 | 例えば、5（値）、num（変数）など |
| 式 | 演算子とオペランドの組み合わせにより構成される数式<br>例えば、 | |



**ソースコード例**

ソースファイル名：Sample4_1.java

```
// 変数を用いた式
class Sample4_1
{
    public static void main(String[] args)
    {
        // 変数宣言と初期化
        int num1 = 15;
        int num2 = 3;

        // num1 と num2 の四則計算
        int add = num1 + num2;
        int sub = num1 - num2;
        int mul = num1*num2;
        int div = num1/num2;

        // 計算結果の出力
        System.out.println(num1 + "と" + num2 + "の四則計算：");
        System.out.println("和 = "+add);
        System.out.println("差 = "+sub);
        System.out.println("積 = "+mul);
        System.out.println("商 = "+div);
    }
}
```

**実行画面**

```
>java Sample4_1
15 と 3 の四則計算：
和 = 18
差 = 12
積 = 45
商 = 5
 -- Press any key to exit (Input "c" to continue) --
```

## ● 演算子の種類

演算子は、算術演算子、ビット論理演算子、シフト演算子、インクリメント・デクリメント演算子、関係演算子、論理演算子、条件演算子、代入演算子など多彩な分類をもつ。

主な演算子

| | | | |
|---|---|---|---|
| （算術演算子） | | （インクリメント・デクリメント演算子） | |
| + | 加算（文字列連結）※１ | ++ | インクリメント |
| - | 減算 | -- | デクリメント |
| * | 乗算 | （関係演算子） | |
| / | 除算 | > | より大きい |
| % | 剰余 | >= | 以上 |
| （ビット論理演算子） | | < | 未満 |
| & | ビット論理積 | <= | 以下 |
| \| | ビット論理和 | == | 等しい |
| ^ | ビット排他的論理和 | != | 等しくない |
| （シフト演算子） | | （論理演算子） | |
| << | 左シフト | ! | 論理否定（単項） |
| >> | 右シフト | && | 論理積 |
| >>> | 符号なし右シフト | \|\| | 論理和 |
| （その他） | | （条件演算子） | |
| + | プラス（単項）※２ | ?: | 条件 |
| - | マイナス（単項） | （代入演算子） | |
| ~ | 補数（単項） | = | |

※１　演算子＋は、加算と文字列連結の２つの役割をもつ。２つの役割は演算子＋のオペランドの種類によりいずれかに定まる。

| | |
|---|---|
| 加算 | 両方のオペランドが数値のとき |
| 文字列連結 | いずれかまたは両方のオペランドが文字列のとき |

例えば、次のコードを実行すると、右のように出力される。

```
System.out.println("ABC"+"DEF");    →    ABCDEF
System.out.println("ABC"+50);       →    ABC50
System.out.println(20+50);          →    70
```

※２　演算子はオペランドの数により単項演算子または二項演算子と呼ばれる。

| | | |
|---|---|---|
| ・単項演算子 | オペランドが１つ | 例えば、++（インクリメント） |
| ・二項演算子 | オペランドが２つ | 例えば、+、-、*、/（四則計算） |

**ソースコード例**

ソースファイル名：Sample4_2.java

```
// 剰余付き割り算プログラム
import java.io.*;

class Sample4_2
{
    public static void main(String[] args) throws IOException
    {
        // キーボード準備
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        // 整数用の変数
        String str1,str2;
        int num1, num2;

        int div; // 商
        int mod; // 余り

        // 整数の入力
        System.out.println("##剰余付き割り算##¥n２つの整数を入力してください。");
        System.out.println("１つ目の整数を入力してください。");
        str1 = br.readLine();
        num1 = Integer.parseInt(str1);
        System.out.println("２つ目の整数を入力してください。");
        str2= br.readLine();
        num2 = Integer.parseInt(str2);

        // 剰余の計算
        mod = num1 % num2; // %は余りを計算する演算子

        // 商の計算
        div = (num1-mod) / num2;
        // ※ここで括弧（）は演算の優先度を表す
        //    詳細は次回の講義で説明する

        // 結果の出力
        System.out.println(num1 + "/" + num2 + "=" + div + "　余り" + mod);
    }
}
```

**実行画面**

```
>java Sample4_2
##剰余付き割り算##
２つの整数を入力してください。
１つ目の整数を入力してください。
11
２つ目の整数を入力してください。
4
11/4=2　余り 3
 -- Press any key to exit (Input "c" to continue) --
```

## ● インクリメント・デクリメント演算子

インクリメント演算子　　++　　値を 1 増やす

a++;　　後置インクリメント　　変数 a を処理した後、a の値を 1 増やす
++a;　　前置インクリメント　　変数 a の値を 1 増やした後、a を処理する

デクリメント演算子　　--　　値を 1 減らす

a--;　　後置デクリメント　　変数 a を処理した後、a の値を 1 減らす
--a;　　前置デクリメント　　変数 a の値を 1 減らした後、a を処理する

### ソースコード例

ソースファイル名：Sample4_3_1.java

```
// 前置・後置インクリメント
class Sample4_3_1
{
    public static void main(String[] args)
    {
        // 変数の宣言と初期化
        int a1 = 0, a2 = 0;
        int b = 0, c = 0;

        System.out.println("a1="+ a1);
        System.out.println("a2="+ a2);

        // 前置・後置インクリメントの違い
        b = a1++;
        c = ++a2;

        System.out.println("後置インクリメント(b=a1++;) b="+ b);
        System.out.println("前置インクリメント(c=++a2;) c="+ c);
        System.out.println("a1="+ a1);
        System.out.println("a2="+ a2);
    }
}
```

**実行画面**

```
>java Sample4_3_1
a1=0
a2=0
後置インクリメント(b=a1++;) b=0
前置インクリメント(c=++a2;) c=1
a1=1
a2=1
 -- Press any key to exit (Input "c" to continue) --
```

**参考：次の場合、前置・後置の違いがどのように演算結果に表れるでしょうか？**
ソースファイル名：Sample4_3_2.java

```
// 前置・後置インクリメント（２）
class Sample4_3_2
{
    public static void main(String[] args)
    {
        // 変数の宣言と初期化
        int a1 = 0, a2 = 0;
        int b = 0, c = 0;
        int num = 5;

        System.out.println("a1="+ a1);
        System.out.println("a2="+ a2);

        // 前置・後置インクリメントの違い
        b = num + a1++;
        c = num + ++a2;

        System.out.println("後置インクリメント(b = num + a1++;) b="+ b);
        System.out.println("前置インクリメント(c = num + ++a2;) c="+ c);
        System.out.println("a1="+ a1);
        System.out.println("a2="+ a2);
    }
}
```

● **代入演算子**

代入演算子　　=　　右辺の値を左辺の変数に代入

複合的な代入演算子　　■=　　演算■と代入を同時に実行

| | | |
|---|---|---|
| a+=b; | 加算代入 | a = a + b; |
| a-=b; | 減算代入 | a = a – b; |
| a*=b; | 乗算代入 | a = a * b; |
| a/=b; | 除算代入 | a = a / b; |
| a%=b; | 剰余代入 | a = a % b; |
| a&=b; | 論理積代入 | a = a & b; |
| a\|=b; | 論理和代入 | a = a \| b; |
| a^=b; | 排他的論理和代入 | a = a ^ b; |
| a<<=b; | 左シフト代入 | a = a << b; |
| a>>=b; | 右シフト代入 | a = a >> b; |
| a>>>=b; | 符号なし右シフト代入 | a = a >>> b; |

**ソースコード例**

ソースファイル名：Sample4_4.java

```
// 複合的な代入演算子を用いた総計処理
import java.io.*;

class Sample4_4
{
    public static void main(String[] args) throws IOException
    {
        String str; // 整数入力用の変数
        int sum=0; // 総計用の変数、ゼロで初期化

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

        // キーボードからの入力を促すメッセージ
        System.out.println("３つの整数の総計を求めます。");
        System.out.println("１つ目の整数を入力してください。");
        str = br.readLine();
        sum += Integer.parseInt(str);
        System.out.println("２つ目の整数を入力してください。");
        str = br.readLine();
        sum += Integer.parseInt(str);
        System.out.println("３つ目の整数を入力してください。");
        str = br.readLine();
        sum += Integer.parseInt(str);

        // 総計の表示
        System.out.println("総計は" + sum + "です。");
    }
}
```

**実行画面**

```
>java Sample4_4
３つの整数の総計を求めます。
１つ目の整数を入力してください。
2
２つ目の整数を入力してください。
3
３つ目の整数を入力してください。
5
総計は 10 です。
 -- Press any key to exit (Input "c" to continue) --
```

● シフト演算子

シフト演算子　　<<、>>、>>>　　指定ビットだけずらす

| | | |
|---|---|---|
| a << b; | 左シフト演算子 | b ビット分 a のビット列を左へずらし、右を 0 で埋める |
| a >> b; | 右シフト演算子 | b ビット分 a のビット列を右へずらし、<br>a が正の場合は 0 で、負の場合は 1 で左を埋める |
| a >>> b; | 符号なし右シフト演算子 | b ビット分 a のビット列を右へずらし 0 で左を埋める |

ソースコード例

ソースファイル名：Sample4_5.java

```
// シフト演算の働き
class Sample4_5
{
    public static void main(String[] args)
    {
        int i;
        short num=5;

        // ビット列出力
        System.out.print("シフト前　");
        for(i=0;i<16;i++)
            System.out.print(0x0001&(num>>(15-i)));
        System.out.println("(b)");

        // シフト演算
        num <<= 2;

        // ビット列出力
        System.out.print("シフト後　");
        for(i=0;i<16;i++)
            System.out.print(0x0001&(num>>(15-i)));
        System.out.println("(b)");
    }
}
```

実行画面

```
>java Sample4_5
シフト前　0000000000000101(b)
シフト後　0000000000010100(b)
 -- Press any key to exit (Input "c" to continue) --
```