

8回目 for 文

今日の講義で学ぶ内容

- for 文
- 変数のスコープ
- for 文の入れ子

繰り返し文 1 for 文

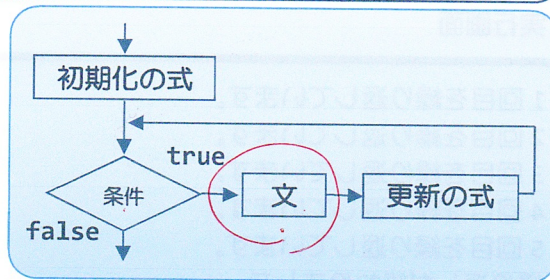
for 文 最初に一度だけ初期化の式を処理します
 条件が true の場合、文を実行し、更新の式を処理して繰り返します
 条件が false の場合、for 文を終了します

条件は boolean 型で、関係演算子で表現される式などを記述します

for(初期化の式 ; 条件 ; 更新の式) 文 ← 1文

セミコロンです

コード例 | for(a=0;a<10;a++)b++;



条件は常に文を実行する前に処理されます (前判定ループといいます)

for 文はブロックを用いて次のように記述することもできます

for(初期化の式 ; 条件 ; 更新の式) { 文1 文2 ... } ← 2文~

または、次のように書くと読みやすく分かりやすいでしょう

for(初期化の式 ; 条件 ; 更新の式)

```

{
  文1
  文2
  :
}
    
```

ソースコード例

ソースファイル名 : Sample8_1.java

```
// for 文の実行
class Sample8_1
{
    public static void main(String[] args)
    {
        int i;
        // 変数 i を1つずつ増やし、1から5になるまで繰り返す
        for(i=1; i<=5; i++)
            System.out.println(i+"回目を繰り返しています。");

        System.out.println("繰り返しが終わりました。");
    }
}
```

Handwritten annotations in red:

- カマコケ (curly braces) → 初期化(1回だけ) ... 繰り返しのスタート
- int i; → ゴール
- i<=5 → 条件
- i++ → 更新
- 繰り返しのスタート
- ゴール
- 中身

1回目 ...
2回目 ...
3回目 ...
4回目 ...
5回目 ...
繰り返し ...

実行画面

```
1回目を繰り返しています。
2回目を繰り返しています。
3回目を繰り返しています。
4回目を繰り返しています。
5回目を繰り返しています。
繰り返しが終わりました。
```

ソースコード例

ソースファイル名 : Sample8_2.java

変数 di

// 1.0 から 3.0 まで 0.5 刻みでの合計を求める

```
class Sample8_2
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        double di; ← 初期値 -
```

```
        double sum=0; // 合計の計算用
```

```
        // 変数 di を 1.0 から 0.5 ずつ増やし 3.0 になるまで繰り返す
```

```
        for(di=1.0; di<=3.0; di+=0.5)
```

```
            sum += di; // sum = (sum + di); と同じ
```

```
            System.out.println("1.0 から 3.0 まで 0.5 刻みでの合計は "+sum+" です。");
```

```
        }
```

```
    }
```

$$1.0 + 1.5 + 2.0 + 2.5 + 3.0 = ?$$

2.5

4.5

7.0

10.0

Sum	di	sum
0	1.0	1.0
1.0	1.5	2.5
2.5	2.0	4.5
4.5	2.5	7.0
7.0	3.0	10.0
10.0	3.5	

実行画面

1.0 から 3.0 まで 0.5 刻みでの合計は 10.0 です。



for 文の初期化の式、条件、更新の式は省略可能です

省略した場合、それぞれ次のような動作をします

- 初期化の式 → 初期化ではなにも実行されません
- 条件 → 常に true になります
- 更新の式 → 更新ではなにも実行されません

たとえば、

```
for( ; ; )
```

```
{
```

```
    :
```

```
}
```

は無限ループです




for 文の初期化の式と更新の式には式文という分類の式を書きます

式文とはセミコロンをその後につけて文とできる式であり、代入演算子、インクリメント・デクリメント演算子を用いた式があります

たとえば、

```
a++;
```

```
b=5;
```

 for 文の初期化の式と更新の式では“,”カンマで区切り 2 つ以上の式を記述できます
カンマで区切られた式は、左から右へ順番に処理されます

// 複数の変数の初期化・更新をおこなう


```
class Ext8_1
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int i, j;
```

 カンマで区切り複数の式を記述できます

// 2つの変数の初期化

```
        for(i=1, j=1; i<=5; i++, j+=2) // カンマで区切る
```

```
            System.out.println(i+" "+j+"="+ (i+j));
```

```
        System.out.println("終わり");
```

```
    }
```

```
}
```

i	j	出処
1	1	1+1=2 ✓
2	3	2+3=5 ✓
3	5	3+5=8 ✓
4	7	4+7=11 ✓
5	9	5+9=14 ✓
6	11	

実行画面

```
1+1=2
```

```
2+3=5
```

```
3+5=8
```

```
4+7=11
```

```
5+9=14
```

```
終わり
```



for 文の初期化の式に変数の宣言を含めることもできます

変数を宣言するのと同じ要領で、1つまたは複数の変数を宣言、初期化することができます

// 変数の宣言と初期化を行う

```
class Ext8_2
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        // 変数の宣言と初期化
```

```
        for(int i=1; i<=5; i++)
```

```
            System.out.println(i+"回目を繰り返しています。");
```

```
        // 同一の型で複数の変数の宣言と初期化を行う
```

```
        for(int i=1, j=2; i+j<=5; i++, j++)
```

```
            System.out.println(i+" "+j+"="+ (i+j));
```

*int i;
double dj;*

```
        // 複数の型の変数の宣言と初期化を行う (エラー)
```

```
//        for(int i=1, double dj=2; i+j<=5; i++, j++)
```

```
//            System.out.println(i+"回目を繰り返しています。");
```

```
    }
```

```
}
```

初期化の式

変数の宣言と
初期化ができます

初期化の式

同一の型の変数の
宣言と初期化がで
きます

異なる型の変数の
宣言と初期化はで
きません

この場合は for 文
に入る前に宣言す
るとよいでしょう

実行画面

1回目を繰り返しています。

2回目を繰り返しています。

3回目を繰り返しています。

4回目を繰り返しています。

5回目を繰り返しています。

1+2=3

2+3=5

変数のスコープとは

その変数を参照可能なコードの上の領域のことです
スコープの開始は、変数の宣言の位置です
スコープの終了は、それが属するブロックの終わりです

直上の

```
// 変数のスコープ
class Ext8_3
{
    public static void main(String[] args)
    {
        int i=10; // main メソッドブロックの最後までがスコープ
        if(i==10)
        {
            int j=10; // if 文ブロックの最後までがスコープ
            System.out.println(i); // OK
            System.out.println(j); // OK
        }
        System.out.println(i); // OK
        System.out.println(j); // コンパイルエラー
    }
}
```

変数 i のスコープ

変数 j のスコープ

double i; X

X

📄 同じスコープ（ネストも含む）内で同名の変数は宣言できません

📄 同じスコープ内に同名の変数が宣言されていると、
「変数ooは△△で定義されています。」
というコンパイルエラーがでます

📄 for 文の初期化の式で宣言される変数のスコープは、
• 初期化の式（その変数以降（右側））
• 条件
• 更新の式
• for 文のブロック
です

たとえば、次の for 文で変数 i のスコープは点線内部です

```
for(int i=1; i<=5; i++)
{
    System.out.println("繰返し番号");
    System.out.println(i);
}
```

? 次のように for 文を記述するとどうなるでしょうか?

// for 文のよくあるミス

```
class Ext8_4
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int i=0;
```

for 文のブロック { } がない場合

次の 1 文が for 文の繰り返して

実行する文と解釈されます

// for 文のブロック { } を忘れたら?

```
for(i=1; i<=5; i++)
```

```
    System.out.println(i+"回目を繰り返しています。");
```

```
    System.out.println("次の繰り返しに進みます。");
```

```
System.out.println("処理を終了します。¥n");
```

単独のセミコロン

文はセミコロンでおわ

る処理です

単独のセミコロンは処

理のない空の文です

// for 文ブロック前に ; (セミコロン) を入れてしまったら?

```
for(i=1; i<=5; i++);
```

```
{
```

```
    System.out.println(i+"回目を繰り返しています。");
```

```
    System.out.println("次の繰り返しに進みます。");
```

```
}
```

```
System.out.println("処理を終了します。");
```

```
}
```

```
}
```

繰り返して実行する

文が空の for 文と解釈

されます

次に続くブロックは

for 文の繰り返しに含

まれず、常に実行され

る通常の文です

実行画面

1 回目を繰り返しています。

2 回目を繰り返しています。

3 回目を繰り返しています。

4 回目を繰り返しています。

5 回目を繰り返しています。

次の繰り返しに進みます。

処理を終了します。

6 回目を繰り返しています。

次の繰り返しに進みます。

処理を終了します。

for 文の入れ子

for 文は、1つの文です

for 文を他の for 文に入れることができ、多重の繰り返しを処理できます

```
for(初期化の式 ; 条件 ; 更新の式) { for 文 }
```

または、次のように書くと多重の繰り返しが分かりやすいでしょう

```
for(初期化の式 1 ; 条件 1 ; 更新の式 1 )
{
    for(初期化の式 2 ; 条件 2 ; 更新の式 2 )
    {
        文
        :
    }
}
```

多重 for 文の動作

外側の for 文が一回繰り返される毎に内側の for 文が処理されます

2~3重の for 文の入れ子はよく使われますので慣れておくといよいでしょう

ソースコード例

ソースファイル名 : Sample8_3.java

// for 文のネスト構造

```
class Sample8_3
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
        int i, j;
```

// 2重の繰り返し

```
    for(i=0;i<5;i++) // 変数 i を 0 から 4 まで繰り返す。
```

```
    {
```

```
        for(j=0;j<3;j++) // 変数 i を繰り返す度に 変数 j を 0 から 2 まで繰り返す。
```

```
        {
```

```
            System.out.println("i は "+i+" : j は "+j);
```

```
        }
```

```
    }
```

```
}
```

(変数)		(表示)	
i	j	i=00: j=00	
0✓	0✓	0	0
0✓	1✓	0	1
0✓	2✓	0	2
0✓	3✓		
1✓	0✓	1	0
1✓	1✓	1	1
1✓	2✓	1	2
1✓	3✓		
2✓	0✓	2	0
		⋮	⋮
		4	
		5	

実行画面

```
iは0:jは0
iは0:jは1
iは0:jは2
iは1:jは0
iは1:jは1
iは1:jは2
iは2:jは0
iは2:jは1
iは2:jは2
iは3:jは0
iは3:jは1
iは3:jは2
iは4:jは0
iは4:jは1
iは4:jは2
```

ソースコード例

ソースファイル名 : Sample8_4.java

// 九九の表

```
class Sample8_4
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
        int i, j;
```

i, j // 九九を計算して表として出力する

スコープ for(i=1; i<=9; i++) // 変数 i を 1 から 9 まで繰り返す。

```
{
```

```
    for(j=1; j<=9; j++) // 変数 j を 1 から 9 まで繰り返す。
```

```
    {
```

```
        // i 段 j 列目の九九を計算
```

```
        System.out.print(i*j+"¥t");
```

```
    }
```

```
    // 1 段毎に改行を入れる
```

```
    System.out.println();
```

```
    }
```

```
}
```

*改行^
出力.*

	1	2	3	4	...	9
1	1	2	3	4	...	9
2	2	4	6	18
3						
4						
...						
9						

実行画面

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

■ 今日の講義のまとめ ■

- for 文は繰返し処理を記述します。
- for 文は、初期化の式と条件、更新の式、繰返し対象の文からなります。初期化の式は最初に一度だけ処理されます。条件が真である間、文と更新の式が繰返し処理されます。条件が偽になると for 文は終了します。
- for 文は、前判定ループです。前判定ループとは、対象となる文を処理する前に条件が評価・判定される繰返し処理のことです。
- 変数のスコープとは、その変数を参照可能なコード上の領域のことです。
- for 文の中に for 文を入れることで多重の繰返しを処理できます。

