# ８回目　for 文

## ●　繰り返し文１　for 文

for 文　①　初期化の式を処理する。
②　繰返し条件を処理する。
A. ②が true のとき、ブロック内を実行して、更新の式を処理する。
B. ②が false のとき、ステップ④へ。
③　ステップ②へ。
④　繰り返しを終了する。

・初期化の式は最初に一度だけ実行される
・繰り返し条件は boolean 型であり、関係演算子で表現される式などを記述
・常に繰り返し条件はブロック内を実行する前に処理される（前判定ループ）

```
for( 初期化の式 ; 繰り返し条件 ; 更新の式 ) // ← セミコロン無し！！
{
    文;
     :
}  // ← ブロック{}内の文が１つの場合、このブロック記号{}は省略できる
```

**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。”);
    }
}
```

**実行画面**

```
>java Sample8_1
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。
 -- Press any key to exit (Input "c" to continue) --
```

**ソースコード例**

ソースファイル名：Sample8_2.java

```
// 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;
		double sum=0;　// 合計の計算用

		// 変数 di を 0.5 ずつ増やし、1.0 から 3.0 になるまで繰り返す
		System.out.println("変数 sum:0.0（初期値）変数 di:1.0～3.0（0.5 刻み繰返し）");
		System.out.println("sum + di --> sum");
		for(di=1.0; di<=3.0; di+=0.5)
		{
			System.out.print(sum+" + "+di+" --> ");
			sum += di;　// sum = sum + di; と同じ
			System.out.println(sum);
		}
		System.out.println("1.0 から 3.0 まで 0.5 刻みでの合計は"+sum+"です。");
	}
}
```

**実行画面**

```
>java Sample8_2
変数 sum:0.0（初期値）変数 di:1.0～3.0（0.5 刻み繰返し）
sum + di --> sum
0.0 + 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
1.0 から 3.0 まで 0.5 刻みでの合計は 10.0 です。
 -- Press any key to exit (Input "c" to continue) --
```

**○　初期化の式、条件の式、更新の式を省略したら？**

初期化の式　　→　初期化ではなにも実行されない
繰り返し条件　→　常に true になる
更新の式　　→　更新ではなにも実行されない

たとえば、

```
for( ; ; )
{
   ・・・
}
```

は無限ループとなる。

**○　初期化の式と更新の式では、カンマで区切って２つ以上の式を記述できる**

```
// 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;

        // 変数の宣言と初期化
        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");
    }
}
```

カンマで区切り
複数の文を記述

○　**初期化の式に変数の宣言（同一の型のみ複数）を含めることもできる**
**宣言された変数のスコープは for 文のブロック内**

```
// 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++) // 宣言と初期化を行う
        {
            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。");
    }
}
```

変数の宣言と
初期化ができる

○　**スコープ**

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
スコープの開始：変数の宣言
スコープの終了：それが属するブロックの終わり

同じスコープ(ネストも含む)内で同名の変数は宣言できない

```
// 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(true)
        {
            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
    }
}
```

**〇次のように for 文を記述するとどうなる？**

```
// 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 {} 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");

        // 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。");
    }
}
```

for 文のブロック { } がない場合は、次の 1 行が for 文の繰り返しで実行される文となる。

繰り返しで実行される文がない for 文となる。次に続くブロックは for 文による繰り返しに含まれず、順次に実行される通常の文となる。

**実行画面**

```
>java Ext8_4
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。
 -- Press any key to exit (Input "c" to continue) --
```

● **for 文の入れ子（ネスト）構造**

for 文のブロック内に for 文をさらに入れた構造であり、多重の繰り返しを処理できる。

```
for( 初期化の式１ ; 繰り返し条件２ ; 更新の式３ )
{
    for( 初期化の式A ; 繰り返し条件B ; 更新の式C )
    {
        文;
        :
    }
}
```

**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 1 度繰り返す度に変数 j を 0 から 2 まで繰り返す。
            {
                System.out.println("i は"+i+" : j は"+j);
            }
        }
    }
}
```

**実行画面**

```
>java Sample8_3
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
 -- Press any key to exit (Input "c" to continue) --
```

**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;

		// 九九を計算して表として出力する
		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.printf(“ %2d”,i*j);
			}
			// 1 段毎に改行を入れる
			System.out.println();
		}
	}
}
```

**実行画面**

```
>java Sample8_4
   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
 -- Press any key to exit (Input "c" to continue) --
```