

### ● 繰り返し文2 while 文

while 文

① 繰り返し条件の式を処理する。

  A. 式が true のとき、ブロック内を処理する。

  B. 式が false のとき、ステップ③へ。

② ステップ①へ。

③ 繰り返しを終了する。

```
while( 繰り返し条件の式 )
{
  文;
  ;
}
```

※ ブロック{}内の文が1つの場合、そのブロック記号{}は省略できる。

#### ソースコード例

ソースファイル名 : Sample9\_1.java

```
// while 文の実行
class Sample9_1
{
    public static void main(String[] args)
    {
        int i = 1;

        // 変数 i が 5 以下なら繰り返す
        while(i<=5)
        {
            System.out.println(i+"回目を繰り返しています。");
            i++; // 変数 i を 1 増やす (ここがなければ無限に繰り返す)
        }
        System.out.println("繰り返しが終わりました。");
    }
}
```

## 実行画面

```
>java Sample9_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 : Sample9\_2.java

```
//フィボナッチ数列の隣接する項の比率は黄金率(1+√5)/2≈1.61803に収束する
class Sample9_2
{
    public static void main(String[] args)
    {
        int n1=1, n2=1;          // 第1項、第2項
        int n_1, n_2, n_3;       // 第3項以降の計算用
        double rate;             // 隣接する項の比率
        double temp=999.0;       // 隣接する項の比率の保存用
        double diff=999.0;       // 隣接する項の比率の差

        // 処理内容の出力
        System.out.println("<フィボナッチ数列の隣接する項の比率の収束>");

        // 順次求めながら比率の収束を探す
        n_1 = n1;
        n_2 = n2;
        while(diff>0.00001 || diff<-0.00001) // 比率の差が 0.00001 以下になるまで
        {
            rate = (double)n_2/n_1;    // 隣接する項の比率を計算
            diff=temp-rate;           // 隣接する項の比率の差
            temp=rate;                // 次回の比較のために比率を保存
            System.out.println(n_2+"/"+n_1+"t="+rate);
            n_3 = n_1 + n_2;          // 次の項を求める
            n_1 = n_2;
            n_2 = n_3;
        }
    }
}
```

## 実行画面

```
>java Sample9_2
<フィボナッチ数列の隣接する項の比率の収束>
1/1      =1.0
2/1      =2.0
3/2      =1.5
5/3      =1.6666666666666667
8/5      =1.6
13/8     =1.625
21/13    =1.6153846153846154
34/21    =1.619047619047619
55/34    =1.6176470588235294
89/55    =1.61818181818182
144/89   =1.6179775280898876
233/144  =1.6180555555555556
377/233  =1.6180257510729614
610/377  =1.6180371352785146
987/610  =1.618032786885246
-- Press any key to exit (Input "c" to continue) --
```

### ● 繰り返し文3 do while 文

do while 文

- ① ブロック内を処理する。
- ② 繰り返し条件の式を処理する。
  - A. 式が true のとき、ステップ①へ。
  - B. 式が false のとき、ステップ③へ。
- ③ 繰り返しを終了する。

```
do{  
    文;  
    :  
    :  
}while( 繰り返し条件の式 ); // ← セミコロンが必要です。
```

※ ブロック{}内の文が1つの場合、そのブロック記号{}は省略できる。

### ソースコード例

ソースファイル名 : Sample9\_3.java

```
// do while 文の実行  
class Sample9_3  
{  
    public static void main(String[] args)  
    {  
        int i = 1;  
  
        // 変数 i が 5 以下なら繰り返す  
        do{  
            // このブロックは最低でも 1 度は処理される。  
            System.out.println(i+“回目を繰り返しています。”);  
            i++; // 変数 i を 1 増やす (ここがなければ無限に繰り返す)  
        }while(i<=5);  
        System.out.println(“繰り返しが終わりました。”);  
    }  
}
```

## 実行画面

```
>java Sample9_3
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) --
```

### ○ while 文と do while 文の違い

while 文 繰り返し条件の式を初めに処理 → ブロックを1度も処理しない場合がある。  
do while 文 繰り返し条件の式を後で処理 → 最低でも1回はブロックを処理する。

Sample9\_3.java で int 型の変数 i を 7 で初期化した場合の実行画面

```
>java Sample9_3
7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 : Sample9\_4.java

```
// キーボード判定付き入力
import java.io.*;

class Sample9_4
{
    public static void main(String[] args) throws IOException
    {
        // キーボード入力の準備
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        int num;

        // 正しい範囲の値が入力されるまで繰り返す
        do{
            System.out.println("1 から 10 までの整数を入力してください。");
            num = Integer.parseInt(br.readLine());
        }while(num<1 || num >10);

        System.out.println("あなたが入力した値は"+num+"です。");
    }
}
```

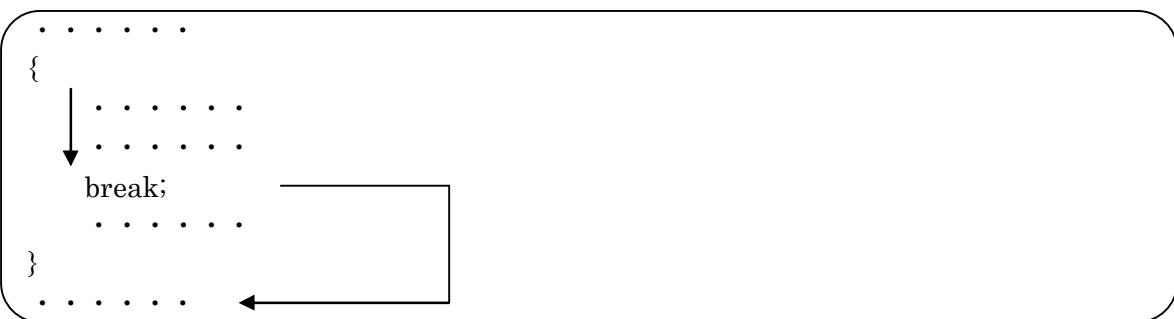
## 実行画面

```
>java Sample9_4
1 から 10 までの整数を入力してください。
-1
1 から 10 までの整数を入力してください。
11
1 から 10 までの整数を入力してください。
7
あなたが入力した値は 7 です。
-- Press any key to exit (Input "c" to continue) --
```

### ● break 文と continue 文

break 文

現在の switch 文ブロックまたは for, while, do while 文ブロックの処理を終了して、そのブロックから抜ける。



#### ソースコード例

ソースファイル名 : Sample9\_5.java

```
// 無限ループから break 文により抜ける
class Sample9_5
{
    public static void main(String[] args)
    {
        int num=3; // 抜け出す繰り返しの回数番目
        int cnt=0; // 繰り返し回数のカウント

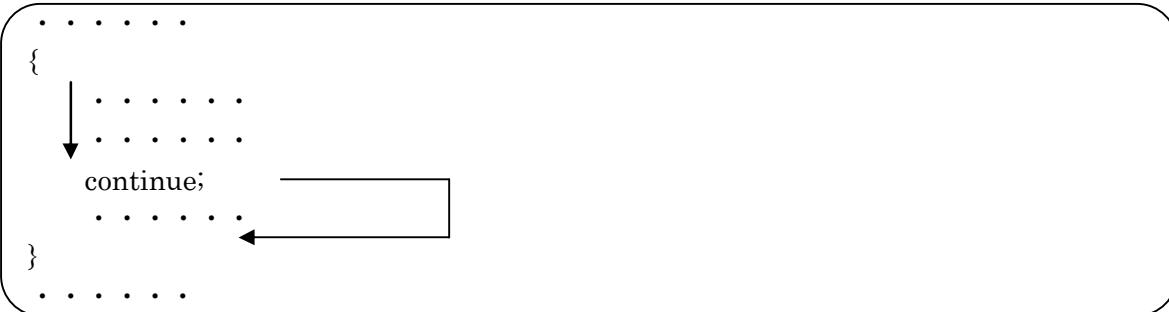
        // 指定された回数番目で break 文により抜ける
        while(true) // 無限ループ
        {
            cnt++;
            System.out.println(cnt+"回目を繰り返しています。");
            if(num==cnt)break;
        }
    }
}
```

#### 実行画面

```
>java Sample9_5
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。
-- Press any key to exit (Input "c" to continue) --
```

continue 文

現在の for, while, do while 文ブロックの処理を終了して、  
そのブロックの終端部にスキップする。



#### ソースコード例

ソースファイル名 : Sample9\_6.java

```
// continue 文により繰り返しをスキップ
class Sample9_6
{
    public static void main(String[] args)
    {
        int i;
        int num=3; // スキップする繰り返しの回数番目

        // 指定された回数番目は continue 文によりスキップ
        for(i=1;i<=6;i++)
        {
            if(num==i)continue;
            System.out.println(i+“回目を繰り返しています。”);
        }
    }
}
```

#### 実行画面

```
>java Sample9_6
1回目を繰り返しています。
2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。
6回目を繰り返しています。
-- Press any key to exit (Input "c" to continue) --
```

○ 入れ子の繰返しの中で **break** 文を使うと？  
break;文からみて最も内側の繰返し文を一つだけ抜ける。

ソースコード例

ソースファイル名 : Sample9\_7.java

```
// 入れ子の繰返しの中で break 文を使う
class Sample9_7
{
    public static void main(String[] args)
    {
        int i;

        // 内側のブロック内で break 文を使うと？
        for(i=0;i<3;i++)
        {
            System.out.println("i="+i);
            while(true)
            {
                break;
            }
        }
    }
}
```

実行画面

```
>java Sample9_7
i=0
i=1
i=2
-- Press any key to exit (Input "c" to continue) --
```