

9回目 while、do while 文

今日の講義で学ぶ内容

- while 文
- do while 文
- break 文と continue 文

繰り返し文 2 while 文

while 文 条件を処理します
条件が true の場合、文を実行して繰り返します
条件が false の場合、while 文を終了します

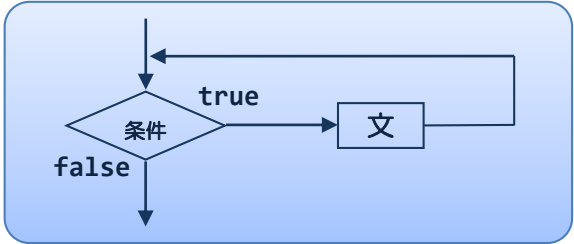
条件は boolean 型で、関係演算子で表現される式などを記述します

```
while(条件) 文
```

コード例 | while(a<5)a++;

📄 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)

📄 前判定ループでは1度も文が実行されない場合があることに注意してください (for 文についても同じです)



while 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

```
while(条件) {文1 文2 ... }
```

または、if 文や for 文の書き方と合せて次のように書くと読みやすく分かりやすいでしょう

```
while(条件)
{
    文1
    文2
    ⋮
}
```

ソースコード例

ソースファイル名 : Sample9_1.java

```
// while 文の実行
public 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

```
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 : Sample9_2.java

```
// 1、2、… と総計を計算したとき、50を超えるのは?  
public class Sample9_2  
{  
    public static void main(String[] args)  
    {  
        int i=1; // 1、2、… と数え上げ用  
        int sum=0; // 総計用  
  
        // 総計が 50 以下なら繰り返します  
        while(sum<=50)  
        {  
            System.out.println(i+"を加算します");  
  
            sum += i; // sum = sum + i; と同じです  
            System.out.println("現在の総計は"+sum+"です");  
  
            i++; // 変数 i を 1 増やす  
        }  
        System.out.println("総計が 50 を超えました");  
    }  
}
```


実行画面

```
1 を加算します  
現在の総計は 1 です  
2 を加算します  
現在の総計は 3 です  
3 を加算します  
現在の総計は 6 です  
4 を加算します  
現在の総計は 10 です  
    :  
9 を加算します  
現在の総計は 45 です  
10 を加算します  
現在の総計は 55 です  
総計が 50 を超えました
```

? for 文と while 文はどのように使い分ければいいのか?

for 文 「〇〇から△△まで □□を繰り返す」
というように繰り返し回数が予め分かる場合に適しています

while 文 「〇〇を満たす間はずっと □□を繰り返す」
というように繰り返し回数が予め分からない場合に適しています

 たとえば、Sample9_2.java では
「総数が 50 以下の間はずっと...」
となります

この場合、繰り返し回数は予め分からないので **while 文** を使うと便利です

 一般に、**for 文** で書ける繰り返し文は **while 文** で書けますし、
その逆もできますので、使い分けにそんなに悩む必要はありません

? 次のように while 文を記述するとどうなるでしょうか?

```
// while 文のよくあるミス
public class Ext9_1
{
    public static void main(String[] args)
    {
        int i=1;

        // while 文のブロック { } を忘れたら?
        while(i<=5)
            System.out.println(i+"回目を繰り返しています。");
            i++;

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

        // while 文ブロック前に ; (セミコロン) を入れてしまったら?
        while(i<=5);
        {
            System.out.println(i+"回目を繰り返しています。");
            i++;
        }
        System.out.println("繰り返しが終わりました。");
    }
}
```

while 文のブロック { } がない場合
次の 1 文が while 文の繰り返しで実行する文と解釈されます

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

繰り返しで実行する文が空の while 文と解釈されます
次に続くブロックは while 文の繰り返しには含まれず、常に実行される通常の文です

繰り返し文 3 do while 文


do while 文 文を実行し、条件を処理します
条件が true の場合、繰り返します
条件が false の場合、do while 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

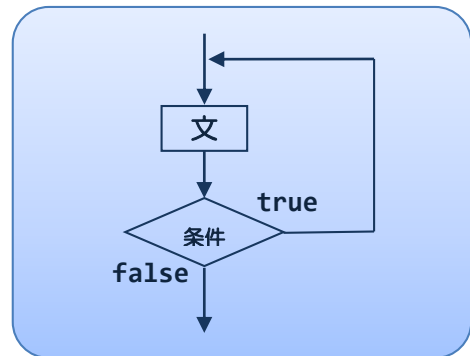
```
do 文 while(条件);
```

条件の括弧の後ろに
セミコロンが必要です

コード例 | `do a++; while(a<5);`

 条件は常に文を実行した後に処理されます
(後判定ループといいます)

 後判定ループでは少なくとも1度は文が実行されることに注意してください



do while 文はブロックを用いて次のように記述することができます

```
do { 文1 文2 ... } while(条件);
```

または、つぎのように書くと読みやすく分かりやすいでしょう

```
do {  
    文1  
    文2  
    :  
} while(条件);
```

ソースコード例

ソースファイル名 : Sample9_3.java

```
// do while 文の実行
public 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

```
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。
```



while 文と do while 文の大きな違いは何でしょうか?

while 文 → 前判定ループ → 文やブロックを **1 度も処理しない場合**があります
(条件を文の前に処理)

do while 文 → 後判定ループ → 最低でも **1 回は文やブロックを処理**します
(条件を文の後に処理)



while 文と do while 文はともに「**〇〇を満たす間は**ずっと**□□を繰り返す**」と
いうように繰り返す回数が予め分からない場合に適しています



for 文と while 文、do while 文はお互いに書き換えることができますので、使い分け
にそんなに悩む必要はありませんが、それぞれの特徴を押さえておくことでスマートなコードが
書けるようになります

Sample9_1.java で int 型の変数 i を 7 で初期化した場合の実行画面

繰り返しが終わりました。

while 文

繰り返しが一度も実行されない場合があります

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

7 回目を繰り返しています。
繰り返しが終わりました。

do while 文

繰り返しは一度は実行されます

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.*;

public 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

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。

7




あなたが入力した値は 7 です。

break 文と continue 文

break 文

switch 文または for、while、do while 文内の実行中の処理を終了し、その文から抜けます

```
for(...)
{
  ↓
  break;
  :
}
  :
```



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
public class Sample9_5
{
    public static void main(String[] args)
    {
        int num=3;    // 抜け出す繰り返しの回数番号
        int cnt=0;    // 繰り返し回数のカウント

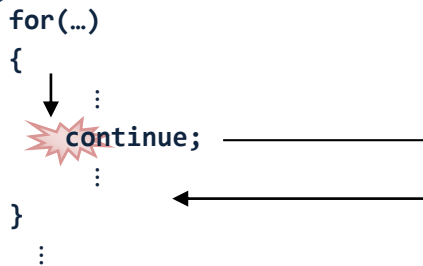
        // 指定された回数番号で break 文により抜ける
        while(true) // 無限ループ
        {
            cnt++;
            System.out.println(cnt+"回目を繰り返しています。");
            if(num==cnt)break;
        }
    }
}
```

実行画面

```
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。
```


continue 文

for、while、do while 文内の実行中の処理を終了し、
繰り返し部分の終端にスキップします



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
public 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号

        // 指定された回数番号は 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

```
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。
```

 繰返しの入れ子（ネスト）構造の中で break 文や continue 文を使うと？

break 文や continue 文からみて最も内側の繰返し文に対して有効になります

break 文 → 最も内側の switch 文や繰返し文を抜けます

continue 文 → 最も内側の繰返し文の繰返し部分の終端にスキップします

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
public 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

```
i=0
i=1
i=2
```

 より外側の繰返し文を break したい場合は
→ ラベル付き break 文

 より外側の繰返し文に対して continue を行いたい場合は
→ ラベル付き continue 文

■ 今日の講義のまとめ ■

- while 文と do while 文は繰り返し処理を記述します。
- while 文は、繰り返し条件と繰り返し対象の文からなります。まず、条件が評価されます。条件が真である間、文が繰り返し実行されます。条件が偽になると while 文は終了します。
- while 文は、for 文と同じ前判定ループです。
- do while 文は、繰り返し条件と繰り返し対象の文からなります。まず、文が実行され、その後で条件が評価されます。条件が真である間、文が繰り返し実行されます。条件が偽になると do while 文は終了します。
- do while 文は、後判定ループです。後判定ループとは、対象となる文を処理した後に条件が評価・判定される繰り返し処理のことです。
- break 文は、switch 文または for、while、do while 文などの実行中の処理を終了し、その文から抜けます。
- continue 文は、for、while、do while 文などの実行中の処理を終了し、繰り返し部分の終端にスキップします。

