

6回目 if文とif else文

今日の講義で学ぶ内容

- 関係演算子
- if文とif-else文
- if文の入れ子

関係演算子

関係演算子 ==, !=, >, >=, <, <= 2つのオペランド間の関係を評価して、真(true)または偽(false)を判断します

演算結果は boolean 型です

boolean 型の変数には論理値リテラルの true と false を代入できます

たとえば、
 • 加算演算子では 1+2 の演算結果は 3 で数値ですが、
 • 関係演算子では 1==1 の演算結果は true で論理値です

関係演算子とその意味

a == b	b が a に等しいとき true となります それ以外では false となります
a != b	b が a に等しくないとき true となります それ以外では false となります
a > b	b より a が大きいとき true となります それ以外では false となります
a >= b	b より a が大きいとか等しいとき true となります それ以外では false となります
a < b	b より a が小さいとき true となります それ以外では false となります
a <= b	b より a が小さいとか等しいとき true となります それ以外では false となります

2項 true/false
 9 == 9

逆 Not
 a < b
 a >= b
 a <= b
 a > b
 a != b

"==" は関係演算子で等価を表し、"=" は代入演算子で代入を表します

3つの関係演算子 != >= <= のイコールはすべて右側に書きます
 左右を逆にしたコードをコンパイルすると、「式の開始が不正です」や「型の開始が不正です」などのコンパイルエラーになりますので注意しましょう

ソースコード例

ソースファイル名: Sample6_1.java

```
// 関係演算子
class Sample6_1
{
    public static void main(String[] args)
    {
        boolean b0, b1, b2, b3;
        int i1=1, i2=10;
        double d1=2.5, d2=5.0;

        // 3は5より小さいので false
        b0 = (5 < 3);

        // i2はi1より大きいので true
        b1 = (i1 < i2);

        // d2*2とi2は等しいので true
        b2 = (i2==(d2*2));

        // すべての括弧を外して b2 = i2==d2*2; としても演算子の優先順位より
        // 乗算->関係演算->代入の順番で演算が行われるため問題ありませんが
        // このように不明確な場合は括弧で優先順位を明示したほうが無難でしょう

        // d1とd2は等しくないため true、それと false は等しくないため false
        b3 = ((d1!=d2)==false);

        System.out.println("b0="+b0+", b1="+b1+", b2="+b2+", b3="+b3);
    }
}
```

関係演算子と型変換
 関係演算子を用いた式を評価する場合も算術演算子と同様にオペランドの型の拡大変換が行われます
 ただし、演算結果は boolean 型です

関係演算子と代入演算子
 代入演算子"="はイコールが1つで、関係演算子"=="はイコールが2つです
 間違えないようにしましょう

0.0 == 10.0

実行画面

b0=false, b1=true, b2=true, b3=false

条件判断文 1 if 文

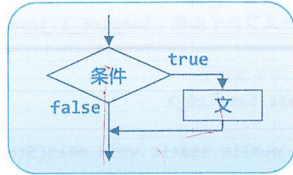
if 文

条件が true の場合、文を処理します
 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
 例えば、`a < b`、`a != 5` など

true / false

if(条件) 文

例 | if(a < 3) System.out.println(a);



文が記述できる場所には複数の文のまとまりとなる `{ }` (ブロック) を記述できます
 ※文は ; (セミicolon) で終わる個々の単一の処理や命令のことです (2 回目の講義を参照)

if 文は次のように記述することもできます

if(条件) { 文1 文2 ... } 2文以上...

コード上の改行やスペース、タブ、改頁は処理に影響を与えません
 ただし、字句を区切る役割は持ちますので、
 ・キーワードや識別子、文字列リテラルなどまとまりある単語の途中に入れたり、
 たとえば、`double` → `doub le`
 ・キーワードや識別子を空白なして続けて書くとコンパイルエラーになります
 たとえば、`int a;` → `inta;`

改行やスペース、タブ、改頁のことをホワイトスペース (whitespace) といいます

ホワイトスペースを上手に用いて、見やすくなるようにコードを工夫しましょう
 たとえば、if 文は次のように書くと分かりやすいでしょう

```

if(条件)
{
    文1
    文2
    :
}
  
```

インデント

ソースコード例

ソースファイル名 : Sample6_2.java

```

// if 文
import java.io.*;

class Sample6_2
{
    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 i;
        System.out.println("整数を入力してください。");
        i = Integer.parseInt(br.readLine());
        // 入力された値を if 文で判断し、1 であればブロック内を処理する
        if(i == 1)
        {
            System.out.println("1 が入力されました。");
            System.out.println("1 が選択されました。");
        }

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

キーボード準備

ブロック毎にインデント

実行の確認
 文の条件の全て
 パターンと確認
 できるうちに!!

この場合
 1が...

実行画面 1

整数を入力してください。
 1
 1 が入力されました。
 1 が選択されました。
 処理を終了します。

実行画面 2

整数を入力してください。
 2
 処理を終了します。

次にように if 文を記述するとどうなるでしょうか?

// if 文のよくあるミス

```
class Sample6_2_1
{
    public static void main(String[] args)
    {
        int i=0;

        // if 文のブロック { } を忘れたら?
        if(i==1)
        System.out.println("1が入力されました。");
        System.out.println("1が選択されました。");
        System.out.println("処理を終了します。¥n");

        // if 文ブロック前に ; (セミicolon) を入れてしまったら?
        if(i==2);
        {
            System.out.println("2が入力されました。");
            System.out.println("2が選択されました。");
        }
        System.out.println("処理を終了します。");
    }
}
```

if 文のブロック { } がない場合
次の 1 文が if 文の条件が真のときに
実行する文と解釈されます

単独のセミicolon

文はセミicolonでおわる処理
です。単独のセミicolonは処理
のない空の文です

条件が真のときに実行する文
が空の文と解釈されます
次に続くブロックは if 文と
は関係のない通常の文です

実行画面

1 が選択されました。
処理を終了します。

2 が入力されました。
2 が選択されました。
処理を終了します。

このようなミスは、Java の文法的には間違いではないためコンパイラエラーとして発見
されることはありません。このようなエラーを論理エラーといいます。

論理エラーを防ぐために、読みやすく分かり易いコードを書くことを心がけましょう!!

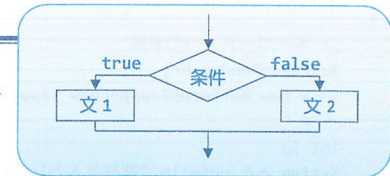
if-else 文

条件が true の場合、文 1 を処理し、
条件が false の場合、文 2 を処理します

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
例えば、a<b、a!=5 など

```
if(条件) 文1 else 文2
```

コード例 | if(a<=0)a=0; else a=1;



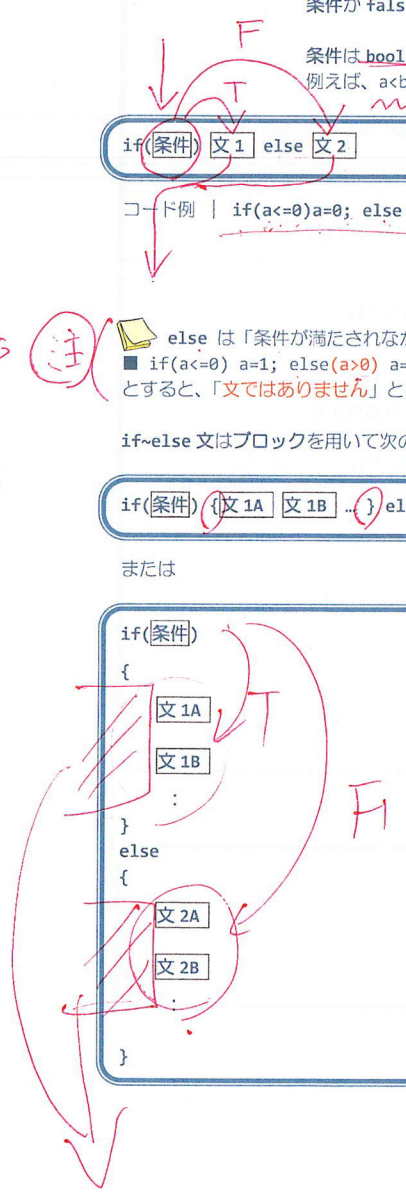
else は「条件が満たされなかった場合」という意味ですので、else の直後に
■ if(a<=0) a=1; else(a>0) a=0;
とすると、「文ではありません」というコンパイルエラーになりますので注意しましょう

if-else 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

```
if(条件) { 文1A 文1B ... } else { 文2A 文2B ... }
```

または

```
if(条件)
{
    文1A
    文1B
    :
}
else
{
    文2A
    文2B
    :
}
}
```



ソースコード例

ソースファイル名: Sample6_3.java

```
// if ~ else 文
import java.io.*;

class Sample6_3
{
    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 i;
        System.out.println("整数を入力してください。");
        i = Integer.parseInt(br.readLine());

        // 入力された値を if ~ else 文で判断し、
        // if(i==1) // 入力値が1であればこのブロックを処理する
        {
            System.out.println("1が入力されました。");
            System.out.println("1が選択されました。");
        }
        // else // 入力値が1以外であればこのブロックを処理する
        {
            System.out.println("1以外が入力されました。");
            System.out.println("1を入力してください。");
        }

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

実行画面 1

整数を入力してください。

1
1が入力されました。
1が選択されました。
処理を終了します。

実行画面 2

整数を入力してください。

2
1以外が入力されました。
1を入力してください。
処理を終了します。

if~else 文の入れ子

if 文や if~else 文は、1つの文です
if 文や if~else 文を他の if 文や if~else 文に入れることができます

```
if(条件) if文1 else if文2
```

if 文や if~else 文は 1つの文ですのでブロックを書かなくてもエラーにはなりません
しかし、次のように if と else の対応が一意的ではなくなる場合があります

```
if (条件A) if (条件B) 文1 else 文2
```

このような場合、

『Javaは逐次的に else を else と対応のない直前の if と対応付けます』

ユーザの解釈

外側に if~else 文
内側に if 文

```
if (条件A) if (条件B) 文1 else 文2
```

Javaの解釈

外側に if 文
内側に if~else 文

誤解を招きますので次のようにブロックを入れて明示しましょう

```
if (条件A) { if (条件B) 文1 } else 文2
```

このような場合は、最初からブロックを付けて明示すると誤解もなくなり良いでしょう

```
if(条件) { if文1 } else { if文2 }
```

ソースコード例
 ソースファイル名 : Sample6_4.java

```
// if ~ else 文の入れ子
import java.io.*;

class Sample6_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 i;
        System.out.println("整数を入力してください。");
        i = Integer.parseInt(br.readLine());

        // 入力された値を if ~ else 文の入れ子で判断し、
        if(i==1) // 入力値が1であればこのブロックを処理する
        {
            System.out.println("1が入力されました。");
            System.out.println("1が選択されました。");
        }
        else // 入力値1以外で、
        {
            if(i==2) // 入力値が2であればこのブロックを処理する
            {
                System.out.println("2が入力されました。");
                System.out.println("2が選択されました。");
            }
            else // 2でなければこのブロックを処理する
            {
                System.out.println("1または2を入力してください。");
            }
        }

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

$i == 1$
 ①
 $i == 2$
 ②
 $i == 3, 4, \dots$
 ③

実行画面 1


整数を入力してください。
 1 
 1が入力されました。
 1が選択されました。
 処理を終了します。

実行画面 2

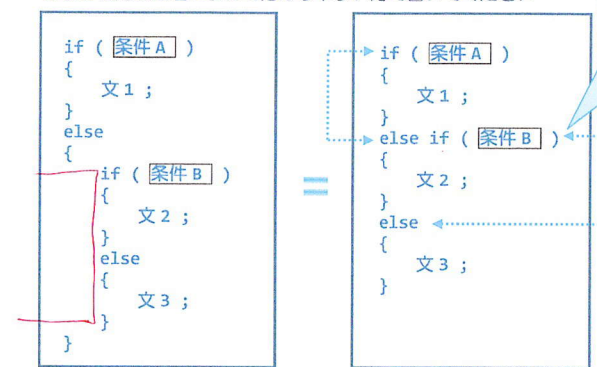
整数を入力してください。
 2 
 2が入力されました。
 2が選択されました。
 処理を終了します。

実行画面 3

整数を入力してください。
 3 
 1または2を入力してください。
 処理を終了します。

 慣れてきたら、Sample6_4.java で出てきた左側のような if~else 文の入れ子を右側のように書くとより読みやすくなり、良いでしょう

どちらも同じ処理ですので分かりやすい方で書いてください



Java の解釈による if と else の対応
 左側と同じ処理をしているのがわかりますね

■ 今日の講義のまとめ ■

- 関係演算子は、オペランドのどちらが大きい小さいか、またそれらが等しいかどうかを演算します。演算結果は、真の場合には true、偽の場合には false です。
- if 文は条件判断を行います。条件が真の場合には、指定された文が実行されます。
- ホワイトスペースとは、改行やスペース、タブ、改項のことです。これらは処理に影響を与えませんので、上手に用いて見やすいコードを心がけましょう。
- if~else 文は、条件が真の場合に実行される文と、条件が偽の場合に実行される文の両方を指定できます。
- if 文や if~else 文を他の if 文などにいれて、入れ子にすることができます。

