

7回目 switch 文と論理演算子

今日の講義で学ぶ内容

- switch 文
- 論理演算子
- 条件演算子

条件判断文 3 switch 文

switch 文 式が case のラベルと一致する場所から直後の break; まで処理します
 どれも一致しない場合、default: から直後の break; まで処理します

式は byte, short, int, char 型 (文字または整数) を演算結果とします
 ラベルには 整数リテラル、文字リテラル を指定します

```

switch(式)
{
    case ラベル1: 文1
        :
        break;
    case ラベル2: 文2
        :
        break;
    :
    default: 文3
        :
        break;
}
    
```

コロンです

セミコロンです

一般に default: は最後に書くようにします

式の結果

実数 × 文字列入
 文 → 2分岐
 switch文
 9分岐

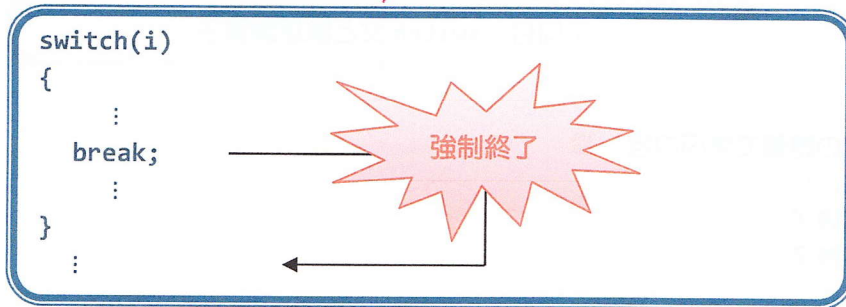
ラベルは重複しないように注意しましょう
 ラベルが重複する場合は「**case ラベルが重複しています。**」のコンパイルエラーになります

default: は指定しないか、または1つ指定するかであり複数指定することはできません
 default: を省略するとどのラベルとも一致しない場合、何もせず switch 文を抜けます

switch 文の式にはこの他列挙型やラッパクラス、またラベルには定数などの定数式を書くことができより柔軟なプログラムが可能です。Javaプログラミング II で解説します。

break 文

switch ブロック内の実行中の処理を強制的に終了し、ブロックから抜けます



ソースコード例

ソースファイル名 : Sample7_1.java

```
// 入力値の判定
import java.io.*;

class Sample7_1
{
    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));

        // キーボード入力
        System.out.println("整数を入力してください。");
        int i;
        i=Integer.parseInt(br.readLine());
        switch(i) // 変数 i により処理を分岐
        {
            case 1: // i が 1 のとき、
                System.out.println("1 が入力されました。");
                break;
            case 2: // i が 2 のとき、
                System.out.println("2 が入力されました。");
                break;
            default: // i が 1 でも 2 でもないとき、
                System.out.println("1 か 2 を入力してください。");
                break;
        }
    }
}
```

X ① ~ ③ キーボード準備

③

②

整数値

一致

"1" ← キーボードからは整数を表す文字列

可能性 3パターン

3通りの実行と確認

抜ける

↓ 続行

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。

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。

実行画面 3

整数を入力してください。

3 

1か2を入力してください。

ソースコード例

ソースファイル名 : Sample7_2.java

// 入力文字の判定

```
class Sample7_2
```

```
{
```

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

```
        char c='b';
```

```
        switch(c) // 変数 c により処理を分岐
```

```
        {
```

```
            case 'a': // cが'a'のとき、
                System.out.println("a です");
                break;
```

```
            case 'b': // cが'b'のとき、
                System.out.println("b です");
                break;
```

```
            default: // cが'a'でも'b'でもないとき、
                System.out.println("a でも b でもありません");
                break;
```

```
        }
```

```
    }
```

CASEの後 文の後
↓ ↓
コン エンコン

-文字で分岐を
行うときは、
ラベルも
文字列ラベルに
すること!!

-教

↓ 続行

実行画面

b です

? switch 文で break; を省略したらどうなる?

- 続けて次のラベルからの処理を行います
- 以降、最初に出会う break; まで来たら switch ブロックを抜けます
- すべての break; を書かない場合 switch ブロックの最後まで来るとブロックを抜けます

Sample7_1.java の break; をすべて取り除いた場合の実行画面です

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。

2 が入力されました。

1 か 2 を入力してください。

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。

1 か 2 を入力してください。

実行画面 3

整数を入力してください。

3 

1 か 2 を入力してください。

論理演算子

<
>
==

論理演算子

!, &&, ||

オペランド間の論理的な関係

- ~ではない
- かつ
- または

を評価して真(true)または偽(false)を判断します

○ × ○

↑
単項 2項

オペランドの数

! は単項演算子です

&& と || は 2 項演算子です

オペランドは boolean 型です

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

論理演算子とその意味

ここで、変数 a と b を boolean 型とします

論理否定

「~ではない」

! ○

a	!a
true	false
false	true

たとえば、

!(3 < 5) → false
! (3 < 5) → false
f

関係演算子と一緒に

関係演算子の演算結果は boolean 型です

論理演算子のオペランドに 関係演算子を用いた式 を書くことが多いです

論理積

「かつ」

○ && ○

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

..... Aが真かつ Bが真

↓
全体が真

たとえば、

(1 == 0) && (1 < 2) → false
(1 == 0) && (1 < 2) → false
f t

論理和

「または」

||

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

..... Aが真 または Bが真

↓
全体が真

たとえば、

(1 == 0) || (1 < 2) → true
(1 == 0) || (1 < 2) → true
f t

ソースコード例

ソースファイル名 : Sample7_3.java

```
// 論理演算子の真理値表
class Sample7_3
{
    public static void main(String[] args)
    {
        System.out.println("!true = "+ (!true));
        System.out.println("!false = " + (!false));
        System.out.println("true && true = "+ (true && true));
        System.out.println("true && false = "+ (true && false));
        System.out.println("false && true = "+ (false && true));
        System.out.println("false && false = "+ (false && false));
        System.out.println("true || true = "+ (true || true));
        System.out.println("true || false = "+ (true || false));
        System.out.println("false || true = "+ (false || true));
        System.out.println("false || false = "+ (false || false));
    }
}
```

実行画面

```
!true = false
!false = true
true && true = true
true && false = false
false && true = false
false && false = false
true || true = true
true || false = true
false || true = true
false || false = false
```

ソースコード例

ソースファイル名 : Sample7_4.java

// 大文字・小文字の処理

```
import java.io.*;
```

```
class Sample7_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));
```

```
        System.out.println("あなたは男性ですか? YかNを入力してください。");
```

```
        // キーボードから1文字を入力
```

```
        char c = br.readLine().charAt(0);
```

```
        if(c == 'Y' || c == 'y') // Yまたはyのとき、
```

```
            System.out.println("あなたは男性ですね。");
```

```
        else
```

```
        {
```

```
            if(c == 'N' || c == 'n') // Nまたはnのとき、
```

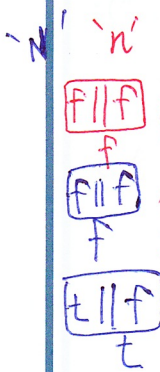
```
                System.out.println("あなたは女性ですね。");
```

```
            else
```

```
                System.out.println("YかNを入力してください。");
```

```
        }
```

```
    }
```



フォーマットを
読み易く



1文字入力と他のキーボード入力

```
// キーボードから文字列を入力
```

```
String str = br.readLine();
```

```
// キーボードから整数を入力
```

```
int i = Integer.parseInt(br.readLine());
```

```
// キーボードから実数を入力
```

```
double d = Double.parseDouble(br.readLine());
```

```
// キーボードから1文字を入力
```

```
char c = br.readLine().charAt(0);
```

キーボードから1文字を読み込む場合 (文字行かとして)

大文字のYまたは小文字のyであれば、if文の条件は真

実行画面

あなたは男性ですか?

YかNを入力してください。

y

あなたは男性ですね。



if 文の条件内の論理演算子 || を switch 文でわかりやすく表現してみましょう

ソースファイル名 : Ext7_1.java

```
// 大文字・小文字の処理 2
import java.io.*;

class Ext7_1
{
    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));

        System.out.println("あなたは男性ですか? ¥nY か N を入力してください。");

        // キーボードから 1 文字を入力
        char c = br.readLine().charAt(0);

        switch(c)
        {
            case 'Y':
            case 'y': // Y または y のとき、
                System.out.println("あなたは男性ですね。");
                break;
            case 'N':
            case 'n': // N または n のとき、
                System.out.println("あなたは女性ですね。");
                break;
            default:
                System.out.println("Y か N を入力してください。");
        }
    }
}
```

break 文を外すと
 case の 1 つだけで → またほ を表現できる
 'Y'、'y' のとき
 同じ == を実行する

条件演算子

条件演算子 `?` : 条件が

- `true` のとき 式 1
- `false` のとき 式 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

演算結果は条件が

- `true` のとき 式 1 の値
- `false` のとき 式 2 の値

です

演算結果の型は式 1 と式 2 の演算結果の型のうち ランクの高い型 です

最終的に演算結果となる値は式 1 と式 2 のどちらかです
たとえば、条件演算子の演算結果を別の変数に代入する場合にどちらでも対応できるようにランクの高い型になるようになっています

$a =$

`条件 ? 式 1 : 式 2`

3つのポイント

コード例 | `a==2 ? 10 : 20;`

コード例 | `a>=0 ? 1 : 1.5;`

ソースコード例

ソースファイル名 : `Sample7_5.java`

// 偶数・奇数の判定

```
class Sample7_5
```

```
{
```

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

```
    {
```

```
        int i = 3; ← 奇
```

```
        // 偶数・奇数の判断
```

```
        String str;
```

```
        str = (i%2==0) ? "偶数" : "奇数";
```

```
        System.out.println("与えられた整数は"+str+"です。");
```

```
    }
```

```
}
```

実行画面

与えられた整数は奇数です。

? 条件演算子と if 文の違いは?

if 文 制御構造の 1 つ → 演算結果をもちません

条件演算子 演算子の 1 つ → 演算結果をもちますので、式の一部に利用できます
たとえば、
`int a=1;`
`int b. = (a==1 ? 10 : 20) + 5;`
とすると、変数 **b** には **15** が代入されます

また、次の条件演算子のコードは

```
ans = 条件 ? 式 1 : 式 2 ;
```

if~else 文を用いて

```
if(条件)  
    ans = 式 1 ;  
else  
    ans = 式 2 ;
```

と同じです

■ 今日の講義のまとめ ■

- switch 文は条件判断文の 1 つです。多分岐の条件判断を見通し良く記述できます。
- 論理演算子は、論理否定「～ではない」、論理積「かつ」、論理和「または」を評価します。論理演算子を用いると、if 文や if ~ else 文でより高度な条件を記述できます。
- 条件演算子は、3 項演算子です。条件演算子を用いた式の演算結果は、与えられた条件が真または偽によりいずれかに決まります。

