７回目　switch 文と論理演算子

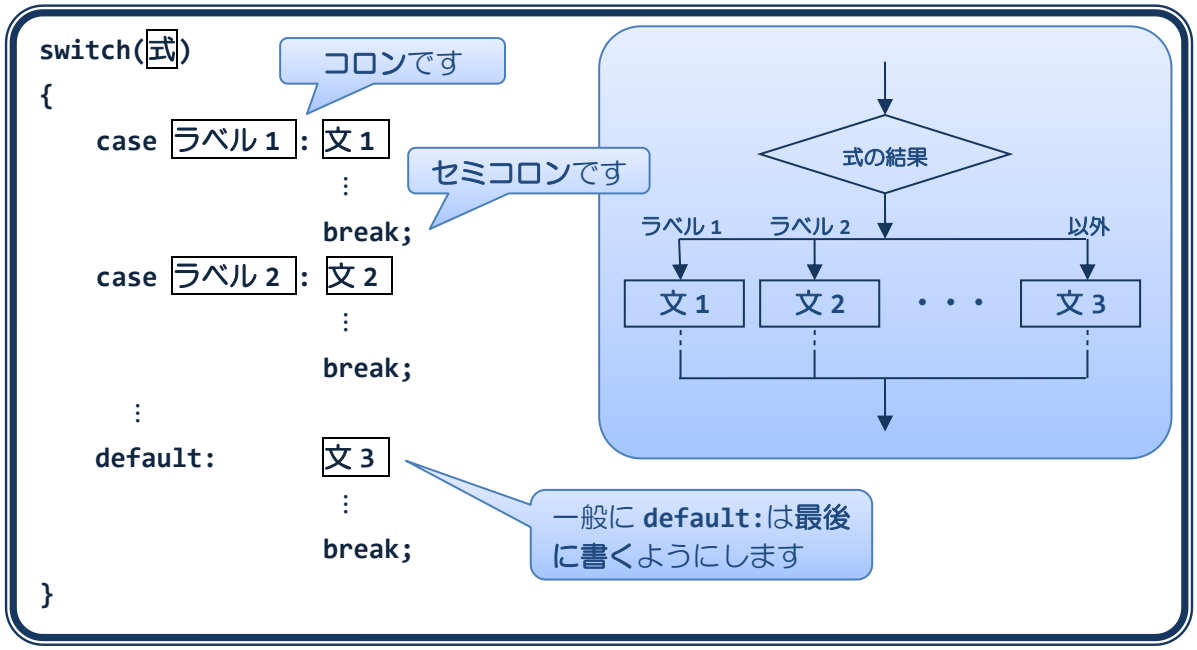
今日の講義で学ぶ内容

- 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
どれにも一致しない場合、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

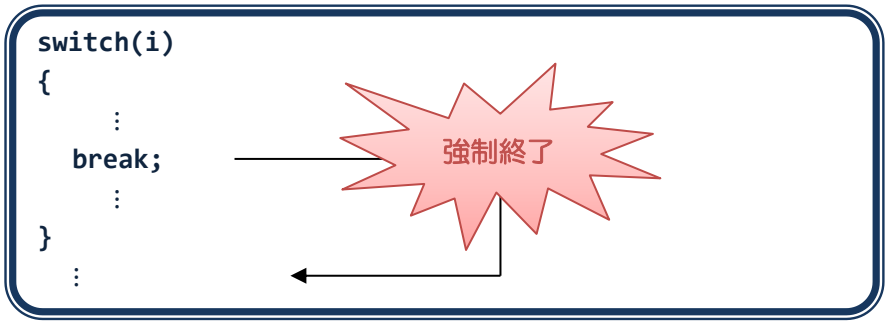


ラベルは**重複しないように**注意しましょう
ラベルが重複する場合は「case ラベルが重複しています。」の**コンパイルエラー**になります

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;
        }
    }
}
```

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;
        }
    }
}
```

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

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

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

論理否定　　**!**
「～ではない」

a	!a
true	false
false	true

たとえば、
!(3 < 5) → false

論理積　　**&&**
「かつ」

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

たとえば、
(1 == 0)&&(1 < 2) → false

論理和　　**||**
「または」

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

たとえば、
(1 == 0)||(1 < 2) → true

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?¥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);

        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 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。");
        }
    }
}
```

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());
// キーボードから一文字を入力
char c = br.readLine().charAt(0);
```

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 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。");
        }
    }
}
```

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

```
条件 ? 式 1 : 式 2
```

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;

        // 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。