7回目　switch 文と論理演算子

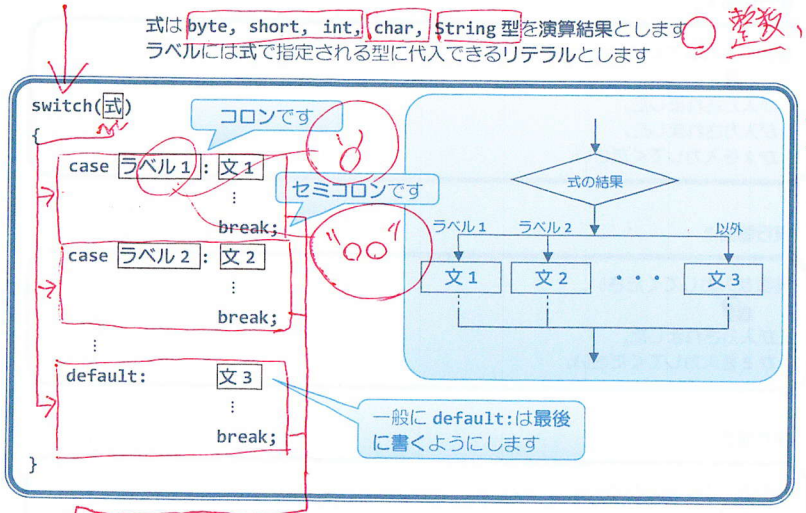
今日の講義で学ぶ内容

- 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, String 型を演算結果とします
ラベルには式で指定される型に代入できるリテラルとします

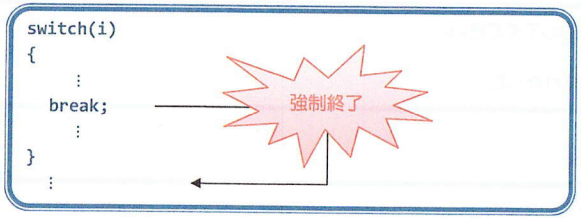


ラベルは重複しないように注意しましょう
ラベルが重複する場合は「case ラベルが重複しています。」のコンパイルエラーになります

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

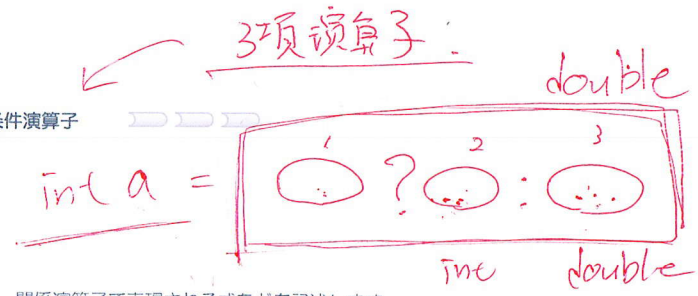
```
あなたは男性ですか?
Ｙか Ｎを入力してください。
y
あなたは男性ですね。
```


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

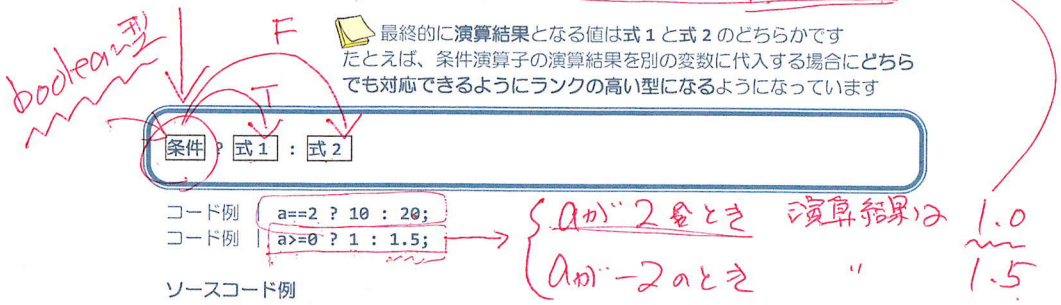
を処理します

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

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。