オブジェクト指向と
ゲームプログラミング

○　Java 入門編 － 第５回　分岐、繰り返し　○

分岐

プログラムの流れを何らかの条件によって分岐させるには、if-else文やswitch-case文を使います。

・if-else文
if-else文は条件を判定し、判定の結果によって実行する処理を分けることができます。

もし～ならば「処理１」を実行し、
そうでなければ「処理２」を実行する

という分岐をさせることができます。書式は以下のようになっています。

```
if(条件式 または boolean値) {
    条件式を満たしたときに実行する処理
} else {
    条件式を満たさないときに実行する処理
}
```

else文以降は、必要ない場合は省略することができます。条件式は、trueかfalseの値か評価結果がboolean型となる式を記述します。C／C++のような、

```
int    value = 0;
if(!value)
   処理１
```

という記述はできません。Javaではboolean値、またはboolean型となる式としなければならないので、

```
int    value = 0;
if(value != 0)
   処理１
```

となります。

条件式に複数の条件を記述したい場合は、論理積演算子"&&"や論理和演算子"||"を使用します。これらの条件式は左から右の順番に評価されます。論理積の方が論理和よりも優先順位が高くなります。

・else-if文
if-else文を連ねて条件分岐を多重に行う場合、else-if文を使用することができます。たとえば、

```
if(条件式１) {
    条件式１を満たしたときに実行する処理
} else if(条件式２) {
    条件式２を満たしたときに実行する処理
} else if(条件式３) {
    条件式３を満たしたときに実行する処理
} else {
    条件式３を満たさないときに実行する処理
}
```

という記述ができます。

・switch-case文
switch-case文は変数の値によって分岐させ、実行する処理を分けることができます。

変数の値が、
定数Ａなら処理１を実行し、
定数Ｂなら処理２を実行し、
定数Ｃなら処理３を実行し、
定数Ｄなら処理４を実行し、
上記のどれでもなければ処理５を実行する

という分岐をさせることができます。書式は以下のようになっています。

```
switch(変数) {
case 定数式１:
    処理１
    break;

case 定数式２:
    処理２
    break;

  (中略)

case 定数式ｘ:
    処理ｘ
    break;

default:
    すべての定数式を満たさなかったときの処理
    break;
}
```

case文とdefault文の最後は":"（コロン）で終了します。case文の分岐条件に記述できるのは整数型の定数式のみです。default文は必要のないときは省略することができます。また、case文の最後にあるbreakは、switch文を抜けるために必要です。breakがないと、breakが見つかるまで、またはswitch文の最後まで実行されます。

```
switch(value) {
case 1:
case 2:
case 3:
    処理１
    break;
case 4:
    処理２
    break;
default:
    処理３
    break;
}
```

このような場合は、変数valueの値が1、2、3の場合は処理１が、4の場合は処理２が、それ以外の場合は処理３が実行されます。

switch-case文では整数型の値による分岐しかできません。String型や小数型の値による分岐、複雑な条件で分岐先が多い場合は、if-else文を連ねて分岐させます。

繰り返し

一定の条件で処理を反復させる制御文として、for文、while文、do-while文があります。

・for文
for文は、条件が真の間処理を反復させることができます。

```
for(初期設定処理; ループ継続条件; ループカウンタ処理) {
    処理本文
}
```

「初期設定処理」は、ループに入る前に一度だけ必ず実行されます。ここでは、ループカウンタの宣言と初期化を行います。","（カンマ）で区切ることにより、複数の変数を宣言し、初期化することができます。

「ループ継続条件」は、ループを継続させるための条件を記述します。条件は、boolean値または評価結果がboolean型となる式でなければなりません。ここがtrueの間、ループが継続されます。ループを終了させるための条件ではないことに注意してください。この条件は、初期設定処理の後にも判定が行われるので、条件によってはループに入らない場合があります。

「ループカウンタ処理」は、処理本文の後に実行され、ループカウンタの増減を行う処理を記述します。","（カンマ）で区切ることにより、複数のカウンタを処理することができます。

「初期設定処理」「ループ継続条件」「ループカウンタ処理」は、必要のないものは省略することができます。

以下のようにfor文を記述すると、ループカウンタiが0から9までの計10回、処理本文が実行されます。

```
for(int i = 0; i < 10; i++) {
    処理本文
}
```

・while文

while文は、ループに入る前に条件判定を行い、条件が真の間処理を反復します。書式は以下のようになっています。

```
while(ループ継続条件) {
    処理本文
}
```

・do-while文

do-while文は、ループの最後に条件判定を行い、真の間処理を反復します。do-while文の書式は以下のようになっています。while文の最後に、";"（セミコロン）が必要なことに注意してください。

```
do {
    処理本文
} while(ループ継続条件);
```

・breakによるループの中断

ループは、ループ継続条件が偽になるまで続けられますが、breakで強制的にループから抜けることもできます。

breakは、ループやswitch文の{}で囲まれたブロックの外側に抜け出すことができます。多重ループの場合は、breakを実行したループからのみ抜け出すことができます。多重ループの外側や指定したループから抜け出すには、ラベルつきbreakを使用します。

```
for(int i = 0, j = 0; i < 10000; i++) {
    処理本文１

    if(j > 65535)
        break;

    処理本文２
}
```

この場合は、ループ継続条件（i < 10000）が偽になるか、jが65535より大きくなるとループが終了します。

```
int   k;
for(int i = 0; i < 10000; i++) {
    for(int j = 0; j < 2000; j++) {
        処理本文

        if(k == 0)
            break;
    }
    // breakはここに抜けます。
}
```

上記のような場合、breakは内側のループから抜けることになります。

```
int   k;
loop1: for(int i = 0; i < 10000; i++) {
    for(int j = 0; j < 2000; j++) {
        処理本文

        if(k == 0)
            break loop1;     // ラベルloop1のループから抜けます
```

```
        }
    }
    // breakはここに抜けます。
```

上記のような場合、breakはラベルloop1のループから抜けることになります。

・continueによる処理のスキップ

ループ内の処理の一部をcontinueでスキップすることができます。for文の場合は、continueが実行されると、そのループに関するループカウンタの増減処理が行われ、ループ継続条件の判定が行われます。while文とdo-while文でcontinueが実行されると、そのループに関する継続条件の判定が行われます。continueでもラベル指定が使用できます。ラベル指定があると、そのラベルがついているループまでスキップします。

```
    for(int i = 0, j = 0; i < 10000; i++) {
        処理本文１

        if(j > 65535)
            continue;

        処理本文２
    }
```

この場合は、jが65535より大きくなるとcontinueが実行され、処理本文２がスキップされます。

```
    int   k;
    for(int i = 0; i < 10000; i++) {
        for(int j = 0; j < 2000; j++) {
            if(k == 0)
                continue;

            処理本文
        }
    }
```

上記のような場合は、continueは内側のループの「処理本文」をスキップします。

```
    int   k;
    loop1: for(int i = 0; i < 10000; i++) {
        for(int j = 0; j < 2000; j++) {
            if(k == 0)
                continue loop1;    // ラベルloop1に抜ける

            処理本文
        }
    }
```

上記のような場合、continueは内側のループをスキップし、ラベルloop1のループに制御を移します。

・無限ループ

for文やwhile文を使って、無限に反復する「無限ループ」を確立することができます。それぞれ以下のようになります。

```
    for(;;) {
        処理本文
    }

    while(true) {
        処理本文
    }
```

無限ループからは、breakで脱出することができます。