7-4.演算子

代入演算子を組み合わせた処理

代入演算子には、変数に値を代入する`=`や、変数の中身に加算する加算代入`+=`などがあります。

```
int a = 3; //変数に3を代入する
a += 5 //変数の値(3)に5を加算する
System.out.println(a); //aの中身を出力

//出力結果
8
```

では、次の場合はどうなるでしょうか。

```
int a = 3;
int b = a += 5
System.out.println(a + b);
```

変数bに入るのは、5が加算される前のa(3)、加算された後のa(8)、どちらでしょうか？

代入演算子の基本ルールとして、代入演算子を挟んだ左右の処理がそれぞれ完全に終わってから代入される、ということを覚えておきましょう。

`int b = a += 5`
この場合、`a += 5`が先に実行され、aの中身は8になります。
その後、aの中身がbに代入されるので、bは8が代入されます。

aとbはどちらも8になり、`a + b`は16となります。

```
//出力結果
16
```

インクリメントの前置と後置

変数に1を加算する`++`や、1を減算する`--`という演算子がありました。
これらを、 **インクリメント演算子** と呼びます。

インクリメント演算子は、`a++`のように、変数の後ろに付けるものをよく見かけます。
しかし、インクリメント演算子は`++a`のように前に付けることも出来ます。

```
int a = 10;
a++; //aに1を加算
System.out.println(a);

++a; //aに1を加算
System.out.println(a);

//出力結果
11
12
```

上記コードを見ると、全く同じような処理に見えます。
しかし、他の演算子と組み合わさる時には注意が必要です。

```
int a = 10;
int b = a++;
int c = ++a;

System.out.println(a);
System.out.println(b);
System.out.println(c);
```

変数に代入する時、インクリメント演算子を後置にすると、現在の変数の中身が代入されてから加算されます。
前置にすると、変数の中身が加算されてから代入されます。

`int b = a++;`は、この時点のaの中身`10`がbに代入されます。
その後、aに1が足され`11`になります。
`int c = ++a;`は、まずaに1が足され`12`になります。

?

その後、cにaが代入されます。
よって、上記コードの出力結果は以下となります。

```
//出力結果
12
10
12
```

覚え方ですが、PCがコードを読み込む時、厳密に前から1つずつ確認しながら判別しているイメージを持っておくと良いです。

```
int b = a++;
//変数aが見つかったからそれを代入。++が見つかったから、1を加算。

int c = ++a;
//++が見つかったから、1を加算。変数aが見つかったからそれを代入。
```

> 練習問題

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = a++ + a + a-- - a-- + ++a;
        System.out.println(b);
    }
}
```

上記コードをコンパイルした表示結果を選んで下さい。

A. 7
B. 32
C. 33
D. 43
E. コンパイルエラーが発生する
F. 実行時に例外がスローされる

【解説】
`int b`の行を読んでいきます。
始めに`a++`があります。
後置なので、計算には今のaの値`10`が使われます。
その後aに1を加算します。

次に`a`があります。これは今のaの値なので`11`になります。

次に`a--`があります。
後置なので、計算には今のaの値`11`が使われます。
その後aから1を減算します。

次に`a--`があります。
後置なので、計算には今のaの値`10`が使われます。
その後aから1を減算します。

次に`++a`があります。
前置なので、まずaに1を足して`10`となり、それが計算に使われます。

よってまとめると、
`int b = 10 + 11 + 11 - 10 + 10;`
となり、`B. 32`が正解です。

論理演算子

条件文の条件式などで使用される論理演算子には`&&`や`||`がありますが、それぞれの記号を1つだけ使った論理演算子も存在します。

演算子	意味
a & b	aとbの両方がtrueであればtrue
a && b	aとbの両方がtrueであればtrue
a | b	aもしくはbのいずれかがtrueであればtrue

演算子	意味
a || b	aもしくはbのいずれかがtrueであればtrue

一見どちらも同じような処理に見えますが、細かい点で少し処理の違いがあります。
上記の表のように条件aとbを記述した時、
記号1個の場合、aとbはどちらも評価や処理が行われます。
記号2個の場合、左のaがfalseだった場合、bの方は何も処理されません。

サンプルコードを見ていきます。

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 10;
        if (10 < a && 10 < ++b) {
            a++;
        }
        System.out.println(a + b);
    }
}

//出力結果
20
```

if文の条件式で`&&`を使用し、左右に条件を記述しています。
`10 < a`の評価は`false`になります。
右側の条件に、`10 < ++b(bに1を加算し、それが10より大きいかどうか)`がありますが、左側の条件がfalseの時点でこちらは処理が行われません。
よって出力結果は、aもbも10のままで、加算した`20`となります。

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 10;
        if (10 < a & 10 < ++b) {
            a++;
        }
        System.out.println(a + b);
    }
}

//出力結果
21
```

記号1つの場合、左側の評価に関わらず、両方とも必ず処理や読み込みが行われます。
`10 < a`は`false`ですが、`10 < ++b`も処理が行われます。
こちらも`false`となりますが、`++b`の処理でbの中身が1加算される処理が行われています。
こちらの出力結果は、aの10とbの11を加算した`21`となります。

このような仕様を使ったひっかけ問題も出ますので、覚えておきましょう。

演算子の優先順位

```
System.out.println(100 % 20 * 30 + 100 / 20);
```

上記のような演算子での計算式があった場合、 **前から一つずつ計算するのではなく、優先順位があるので気をつけましょう。**

試験対策で覚えておくのは、 **通常の算数の計算式と同じく、掛け算や割り算は、足し算や引き算よりも優先して行う** という点で大丈夫です。
また、余りを出す剰余算`%`は、掛け算や割り算と同じく優先するグループです。

上記の例の計算順は以下のとおりとなります。

1.`100 % 20` = 0
2. `0 * 30` = 0
3. `100 / 20` = 5
4. `0 + 5` = 5

また、計算にカッコを付ければ、その部分は足し算や引き算であっても優先して計算を行わせることができます。
これも普通の算数と同じ仕組みですね。

```
System.out.println(100 % 20 * 30 + (50 + 50) / 20);
```

課題

満点を取れるまで小テストを受験して下さい。

制限時間: 30 分

評定方法: 最高評点

合格点: 100 / 100

受験件数: 110