● JKad24D「2 進数で表示しよう！③」

リスト１は getBinString メソッドのコードである。引数で受け取った整数を 8 ビットの 2 進数文字列として返す。ただし、負の数には対応していない。負の数のときも 2 進数に変換するように getBinString メソッドを修正し、入力された整数を 2 進数表示する処理を作成せよ。

リスト１：getBinString メソッド

```
public static String getBinString(int n) {
    String strBinary = "";
    for (int i = 0; i < 8; i++) {
        strBinary = (n % 2) + strBinary;
        n /= 2;
    }
    return strBinary;
}
```

リスト１のコードで実行したとき

```
整数を入力してください>-99
-99  0-1-1000-1-1
```

課題完成時の画面①

```
整数を入力してください>99
99   01100011
```

課題完成時の画面②

```
整数を入力してください>-99
-99  10011101
```

● JKad24C「シフト演算！」

入力した整数を 2 進数で表示し、右の表の演算を実施した結果を表示する処理を作成せよ。なお、2 進数表示は JKad24D の getBinString メソッドを使って OK（JKad24D.getBinString(整数)で呼び出すことが可能）。

0	<< (左シフト)
1	>> (右シフト)
2	~ (NOT 演算、ビット反転)

課題完成時の画面①（<<演算）

```
整数を入力してください>151
151  10010111
何の演算をしますか？ (0：<<、1：>>、2：~) >0
<<   10010111
------------
     00101110
```

課題完成時の画面②（>>演算）

```
整数を入力してください>151
151  10010111
何の演算をしますか？ (0：<<、1：>>、2：~) >1
>>   10010111
------------
     01001011
```

課題完成時の画面③（NOT 演算）

```
整数を入力してください>151
151  10010111
何の演算をしますか？ (0：<<、1：>>、2：~) >2
~    10010111
------------
     01101000
```

● JKad24B「ビット演算！」

入力した 2 つの整数を 2 進数で表示し、右の表の演算を実施した結果を表示する処理を作成せよ。

0	& (AND 演算)
1	| (OR 演算)
2	^ (XOR 演算)

課題完成時の画面①（AND 演算）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何の演算をしますか？ (0 : AND、1 : OR、2 : XOR) >0
     00111100
AND  00111010
------------
     00111000
```

課題完成時の画面②（OR 演算）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何の演算をしますか？ (0 : AND、1 : OR、2 : XOR) >1
     00111100
OR   00111010
------------
     00111110
```

課題完成時の画面③（~演算）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何の演算をしますか？ (0 : AND、1 : OR、2 : XOR) >2
     00111100
XOR  00111010
------------
     00000110
```

● JKad24X「2 進数のかけ算！」※スペースの都合でここにありますがこれは課題 X です！

2 進数のかけ算アルゴリズム（←検索）を使ってかけ算を行う mul メソッドを作成し、「*演算子」によるかけ算と mul メソッドによるかけ算の結果が同じになることを調べよ。

書式	仕様
public static int mul(int n1, int n2)	2 進数のかけ算アルゴリズムを使って n1 と n2 のかけ算を行い、結果を返す。

課題完成時の画面①

```
0～255 の整数 1 を入力してください>60
0～255 の整数 2 を入力してください>58
*によるかけ算：  3480
2 進数のかけ算：  3480
```

課題完成時の画面②

```
0～255 の整数 1 を入力してください>255
0～255 の整数 2 を入力してください>100
*によるかけ算：  25500
2 進数のかけ算：  25500
```

● JKad24A「勇者の状態！」

勇者の状態悪化と回復処理を作成せよ。状態は int 型変数 1 つで管理するものとし、それぞれ以下のビットに対応する。ビットが 0 のとき正常（GOOD）、1 のとき悪化（BAD）とする。

状態	ビット	10 進表示	回復用のビット	10 進表示
毒	0b00000001			
眠り	0b00000010			
沈黙	0b00000100			
麻痺	0b00001000			
石化	0b00010000			
混乱	0b00100000			
病気	0b01000000			
呪い	0b10000000			

表の空欄は各自で埋めること。

課題完成時の画面

```
勇者の状態：
毒：GOOD 眠り：GOOD　沈黙：GOOD　麻痺：GOOD　石化：GOOD　混乱：GOOD　病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞1
状態コードを入力してください＞2                                    ← 眠り

勇者の状態：
毒：GOOD 眠り：BAD　 沈黙：GOOD　麻痺：GOOD　石化：GOOD　混乱：GOOD　病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞1
状態コードを入力してください＞8                                    ← 麻痺

勇者の状態：
毒：GOOD 眠り：BAD　 沈黙：GOOD　麻痺：BAD　 石化：GOOD　混乱：GOOD　病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞1
状態コードを入力してください＞48                                   ← 石化＋混乱

勇者の状態：
毒：GOOD 眠り：BAD　 沈黙：GOOD　麻痺：BAD　 石化：BAD　 混乱：BAD　 病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞2
状態コードを入力してください＞253                                  ← 眠りの回復

勇者の状態：
毒：GOOD 眠り：GOOD　沈黙：GOOD　麻痺：BAD　 石化：BAD　 混乱：BAD　 病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞2
状態コードを入力してください＞0                                    ← 全回復

勇者の状態：
毒：GOOD 眠り：GOOD　沈黙：GOOD　麻痺：GOOD　石化：GOOD　混乱：GOOD　病気：GOOD　呪い：GOOD
どうしますか？（1：悪化させる、2:回復させる、-1：終了）＞-1
終了します！
```

● JKad24S「トリガー検出！」

入力した 2 つの整数を 2 進数で表示し、変化したビット（もしくは変化しなかったビット）を検出する処理を作成せよ。なお、XOR 演算の使用は NG とする。

0	0→1 へ変化したビット
1	1→0 へ変化したビット
2	変化しなかったビット

ヒント：(もしまだ習っていなかったら)「主加法標準形」を検索すること。

課題完成時の画面①（0→1 へ変化したビットの検出）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何を検出しますか？（0：0→1、1：1→0、2：変化なし）>0
     00111100
0→100111010
------------
     00000010
```

整数 1	整数 2	0→1	必要な最小項
0	0	0	
0	1	1	
1	0	0	
1	1	0	

課題完成時の画面②（1→0 へ変化したビットの検出）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何を検出しますか？（0：0→1、1：1→0、2：変化なし）>1
     00111100
1→000111010
------------
     00000100
```

整数 1	整数 2	1→0	必要な最小項
0	0		
0	1		
1	0		
1	1		

課題完成時の画面③（変化しなかったビットの検出）

```
整数 1 を入力してください>60
整数 2 を入力してください>58
60   00111100
58   00111010
何を検出しますか？（0：0→1、1：1→0、2：変化なし）>2
     00111100
X→X00111010
------------
     11111001
```

整数 1	整数 2	変化なし	必要な最小項
0	0		
0	1		
1	0		
1	1		

※XOR 演算の使用は NG