プログラミングA　（山内クラス）　第１０回

――――　今日の目標　――

教科書第７章「配列」の7.7　（多次元配列）　の応用問題を学ぶ。　次の問に答えられるようにする。

―　プログラムの構造を自分で最初から考えてみよう

―　多次元配列を使ってプログラムを考えよう

――

１．　２次元配列を書き出す

［課題１］　3×3 の整数の配列 `g` を用意します。初期値を　`g = {{1, 0, 0}, {0. 1. -1}, {0, -1, 1}}` としてください。

`g` を画面上にプリントするプログラムを書いてください。但し、全体が3×3のマス目になるようにします。行ごとに、１列ずつ（右側に）書き足していけばうまく書けます。書き足すやり方は、`System.out.println()` の代わりに、プリントの後に改行しない `System.out.print()` を使うと、やりやすいでしょう。教科書p20のSample2に `System.out.print()` の説明があります。別の方法としては、印刷する文字列を変数（たとえば `outstr`）に保持することにして、`outstr = outstr + "o";` とか `outstr = outstr + " ";` とかやりながら１行分の出力文字列を `outstr` に作って、１行分が完成したらそれを `System.out.println(outstr)` として書き出す方法も考えられます。

マス目の境界には線（横の線はマイナス、縦の線は縦棒 `|` を入れてください。配列 `g` の値が `0` のマスには空白を、`1` のマスには文字 `o`（小文字のオゥ）を、`-1` のマスには文字 `x`（小文字のエックス）を書いてください。全体の幅が揃って、下の図のようになるように工夫してください。（下の図では、マス目の境界線の交点には文字 `+` を入れ、また、横方向に幅が詰まりすぎるので、文字と境界線の間に空白を１つ入れてあります。この空白は面倒ならサボってよいです。）

```
o |   |
- + - + -
  | o | x
- + - + -
  | x | o
```

`g` の初期値を、`g = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}` や、`g = {{-1, -1, 1}, {0, 1, 1}, {1, 0, -1}}` に変えて、正しく表示されるか確かめてください。

2.　横３つ並び、縦３つ並び、斜め３つ並びのチェック

横３つ並びは、横に `o` が３つ並んだり、横に３つ `x` が 並んでいる状態です。１行目でも２行目でも３行目でも、どれでも構いません。

縦３つ並びは、縦に `o` が３つ並んだり、縦に３つ `x` が 並んでいる状態です。１列目でも２列目でも３列目でも、どれでも構いません。

斜め３つ並びは、斜めに `o` が３つ並んだり、斜めに `x` が並んでいる状態です。これは対角線上しか起こりません。

```
o | o | o      |   |        |   |       o |   |        | o |       o |   |        |   | o
- + - + -    - + - + -    - + - + -    - + - + -    - + - + -    - + - + -    - + - + -
  |   |      o | o | o      |   |       o |   |        | o |        | o |        | o |
- + - + -    - + - + -    - + - + -    - + - + -    - + - + -    - + - + -    - + - + -
  |   |        |   |      o | o | o    o |   |        | o |        |   | o    o |   |
```

3×3のマス目上の、どこかに`o`の３つ並びがあれば`o`の勝ち、どこかに`x`の３つ並びがあれば`x`の勝ち、とします。　つまり「３目並べ」です。

［課題２］　ある`g`のデータ（3×3）があるとき、その`g`の状態が、`o`の勝ちであるか、`x`の勝ちであるか、どちらでもないか、の判定をしてください。

次の`g`の初期データに対して、判定をした結果を出力してください。

```
g = {{0, -1, 0}, {0, 1, 1}, {1, 0, -1}}
```

```
g = {{-1, -1, 1}, {0, 1, 1}, {1, 0, -1}}
```

3.　打てるかどうかのチェック

既に`o`や`x`が置かれている場所に、重ねて`o`や`x`を置くことはできません。ある場所に置きたいと思った時に、置いてもよいかどうか（置けるかどうか）を判定してください。

［課題３］　たとえば、`g`の値を `g = {{0, -1, 0}, {0, 1, 1}, {1, 0, -1}}` に初期化しておいて、

１）　位置 (x, y) = (0, 0)（右上のマス）に置けるでしょうか？　　まだ空白なので置けますね。

２）　位置 (x, y) = (0, 1)（上段の真ん中のマス）に置けるでしょうか？　　既に`x`が置いてある（-1）ので、置けません。

4.　コンピュータに打たせるのに、ランダムな位置を選ぶ

さて、次に考えたいのは、人間とコンピュータが交互に石を置く（打つ）ことです。

コンピュータの番になった時に、コンピュータはどこに打つべきか？　戦略を考えるのはまだ難しいので、ランダムにマスを選ぶことにします。

ただ、３で考えたように、すでに埋まっている所には打つことはできません。　これを回避する方法を考えます。

こんな方法はどうでしょうか？

とにかくランダムに１カ所選ぶ。つまり「サイコロを振って０から２までのランダムな数を作る」という操作を２回繰り返して、縦と横の位置にします。

0から2までのランダムな数`x`を作るには、

```
r = Math.random();
if (r<(1.0/3.0)) {
    x = 0;
} else if (r<(2.0/3.0)) {
    x = 1;
} else {
    x = 2;
}
```

などとすればよいでしょう。同じような処理で`y`も作ります。　なお、もっと凝った方法（random()の結果を３倍して`int`に変換するとか）を考えたい人は自分でどうぞ。

たとえば、２回振った結果が（１、０）であれば、場所（１、０）を、次に置くマスの候補にします。もし、そのマスが既に埋まっていたら、もう一度　≪「サイコロを振って０から２までのランダムな数を作る」という操作を２回繰り返す≫　をやります。これで得られたマスがまだ埋まっていたら、もう一度≪「サイコロを振って０から２までのランダムな数を作る」という操作を２回繰り返す≫　を繰り返します。空いているマスが得られるまでこれを繰り返して、次に打つ場所を決めます。言い換えると、`while`文を使って、ランダムにマスの座標`(x, y)`を生成して、その`(x, y)`が「打てるところ」

になるまで、繰り返します。「打てるところ」だったらループを終了し、その`(x, y)`を「コンピュータが次に打つ場所」として使います。

［課題４］　それでは、`g`の値を`g = {{0, -1, 0}, {0, 1, 1}, {1, 0, -1}}`に初期化しておいて、次に打てるランダムな場所（マス）を探して、表示してください。

5.　人間が打つ場所を入力する

人間とコンピュータが交互に打つことを考えます。人間を`o`（`g`の中では`1`）として、コンピュータを`x`（`g`の中では`-1`）としましょう。

これをするためには、人間が打ちたい場所（マス）の位置を、キーボードから入力する必要があります。教科書p66～67に従って、「２つの整数を入力する」というやり方を使いましょう。２つの入力した整数の値を、`myx`と`myy`としておきましょう。

［課題５］　上記のように入力し、それが打ってよい場所であるかどうかの判断をしてみましょう。課題３の所で考えたのと同じように、`g`の値を`g = {{0, -1, 0}, {0, 1, 1}, {1, 0, -1}}`に初期化しておいて、マスの位置`(myx, myy)`が置けるかどうかの判定を表示してください。

6.　人間が、置いてよい場所を指定するまで、入力を繰り返させる

５で置けないマスを入力したときは、もう一度やり直すようにしたいです。つまり、`while`を使って、置けるところが入力されるまで、繰り返して入力することを考えます。考え方は４のところと同じです。、`while`文でループし、入力した位置`(myx, myy)`が打ってよい（置いてよい）場所になるまで繰り返し、ループを脱出したときには、`(myx, myy)`が、次に人間が打つ場所になります。

［課題６］　それでは、`g`の値を`g = {{0, -1, 0}, {0, 1, 1}, {1, 0, -1}}`に初期化しておいて、人間に次に打ちたい場所を入力させて、それが打てる場所（マス）になるまで繰り返すように、プログラムしてください。

7.　全体を組み立てる

人間とコンピュータが交互に打つ仕組の、いちばん外側の構造を考えてみます。まず盤面`g`を空っぽに初期化しておいて、`while`文でループします。このループは、`int`型の変数`owari`が値`1`になるまで繰り返します。但し、変数`owari`は、最初に初期値`0`にしておき、終りの条件（３つ並びが見つかった）時に`1`にすることにします。

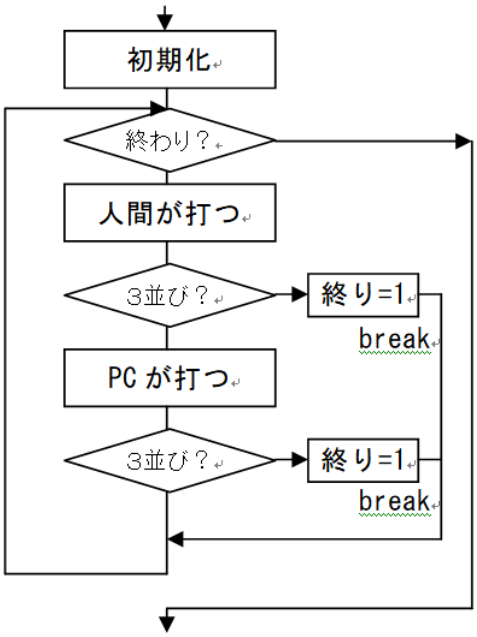
`while`ループの中身は、図のように考えます。まず人間が打ちます。つまり先手は人間です。課題６で見たように、この中に内側の`while`を置いて、人間が置けないところに打ったらやり直させます。うまく空いているところに打ったら、`g`のその座標`(myx, myy)`の位置に`1`（人間＝`o`とする）を書込みます。ここで書込んだ結果の`g`を表示しましょう。

更に、結果の`g`上に、３並びがあるかどうか、課題２のやり方を使ってチェックします。３並びがあれば、人間の勝ちです。「`o`の勝ち」というメッセージを出力してから、`owari`を`1`に書き換えるとともに、`while`ループの残りの部分をすっ飛ばすために`break`を入れておきます。３並びがなければ、次へ続けます。

次は、コンピュータが打つ番です。コンピュータが打つ手続きは、課題４で考えたとおりです。もう一度この部分の手順を要約すると、ランダムに整数を２つ作って、それを`(x, y)`として、それが打てる場所かどうかを判定して、打てる場所が得られるまで（内側の）`while`ループで繰り返します。打てる場所`(x, y)`が得られたら、`g`上で`(x, y)`の指す場所に、値`-1`（コンピュータ側＝`x`とします）を置きます。ここで書込んだ結果の`g`を表示しましょう。

更に、結果の`g`上に、３並びがあるかどうか、課題２のやり方を使ってチェックします。３並びがあればコンピュータの勝ちです。「`x`の勝ち」というメッセージを出力してから、`owari`を`1`に書き換えるとともに、`while`ループの残りの部分をすっ飛ばすために`break`を入れておきます。３並びがなければ、次へ続けます。

次といっても、あとはループの先頭へ戻るだけなので、ループのボディ部分をここまでとします。



8.　　時間が余った人向けの、改良点　（順番は特に意味はありません。好きなものからどうぞ）

(1) いつ盤面を表示するのがよいか、もう一度考えてみてください。

(2) 人間が途中でやめたくなった時、打つ手の入力値をたとえば `-1` と入れると終わりになる、というような工夫をしてください。

(3) 五目並べに拡張してください。

(4) コンピュータ側が、現在はランダムなので、下手な手を選ぶことがあります。人間同士が闘っているように、相手をやりにくくする（たとえば３並びの完成を妨害する）手を打つためには、ランダムにマスを選ぶのではなくて、`o` が３並びの完成に必要そうな場所（＝２つ既に並んでいて３つ目がまだ空いている所）を見つけて、妨害して打つ必要があります。どうやったらできるか、考えてみてください。

注　３目並べも５目並べも、先手必勝

実は、３/５目並べは先手必勝です。人間が先手にしてあるので、コンピュータはどう頑張っても負けます。参考まで。