程式設計 期末專題

主題：康威生命遊戲

學號：110813037

姓名：余珮綺

指導老師：楊凱翔 教授

1. 摘要

本專題將透過上課所學習到的 java 程式實現能夠讓使用者互動的康威生命遊戲，讓使用者對這項數學遊戲有基本的認識與直觀的感受。

2. 製作動機

在搜尋數學遊戲的過程當中，發現了這個數學問題且是我之前從未聽過的，此外它結合了生物發展的概念讓我想起了曾經在三類組的寶貴時光，所以便決定製作出這款簡單的小互動遊戲。

3. 康威生命遊戲介紹

康威生命遊戲為英國數學家何頓．康威（John Horton Conway）所發明，又稱康威生命棋。在一個二維世界中，每個方格都居住了一個生命，而該生命在下世代的生死由其周圍八個生命決定。

規則如下：

A. 若該細胞圍為存活狀態，而其周圍存活細胞小於 2 個時，該細胞將在下一刻變成死亡狀態。(模擬生命數量過低)

B. 當該細胞為存活狀態，而周圍有 2 個或 3 個存活細胞時，則該戲胞保持原樣。(模擬生命數量適中)

C. 當該細胞為存活狀態，而周圍有大於 3 個細胞存活時，該細胞將在下一刻變成死亡狀態。(模擬生命數量過多)

D. 當該細胞為死亡狀態，而周圍有三個存活細胞時，該細胞將於下一刻變成存活狀態。(模擬繁殖)

而在遊戲進行中，會逐漸發展出各種精緻有形的結構，其結構往往有良好的對稱性，而有的形狀會逐代變化，有的則不會在進一步演化。此時若有別的細胞入侵造成結構的破壞，在不久後仍能漸漸找出其秩序與規律。

4. 製作過程

在搜尋遊戲的過程中，儘管也有搜尋到 java 語法的寫法，但都是直接使用 random 隨機創造這個二維世界，之後讓電腦自動去模擬它接下來所有的生命週期變化，並沒有讓使用者輸入盤面並指定想要看見幾個周期變化或第幾周期時的盤面，缺乏了互動的過程，所以我決定加入這兩項特點以增加此款遊戲時的互動性。

一. 將遊戲規則輸出以讓玩家知道康威生命棋的運作模式

程式碼：

```
System.out.println("康威生命遊戲 Conway's Game of Life");
System.out.println();
System.out.println("俗話(？)說的好，No game no life，歡迎大家來到用生命下棋的世界(誤)");
System.out.println("以下有個規則聽好了：");
System.out.println("每個細胞在當前的狀況下，檢查以自己為中心的周邊共8個細胞，而這些細胞的數量將會決
System.out.println();
System.out.println("以下幾點請注意：");
System.out.println("1. 當周邊低於2個(不含2)存活細胞時，他將在下一個世代變成死亡狀態");
System.out.println("2. 當周邊有2個或3個存活細胞時，他將在下一個世代保持不變");
System.out.println("3. 當周邊有3個存活細胞時，他將在下一個世代變成存活狀態");
System.out.println("4. 當周邊超過3個存活細胞時，他將在下一個世代變成死亡狀態");
System.out.println();
System.out.println("好了，了解這些規則之後，終於輪到你主宰生命的時刻了");
System.out.println("請先輸入盤面的長與寬");
System.out.println("再輸入你構想的盤面(令存活細胞為1，死亡細胞為0)，創造這個世界");
System.out.println("接著請輸入你想要知道幾個世代的變化");
System.out.println("最後請輸入你想要知道第哪個世代的變化");
System.out.println("廢話不多說，我們開始吧！");
System.out.println();
System.out.println();
```

輸出結果：

```
康威生命遊戲 Conway's Game of Life

俗話(？)說的好，No game no life，歡迎大家來到用生命下棋的世界(誤)
以下有個規則聽好了：
每個細胞在當前的狀況下，檢查以自己為中心的周邊共8個細胞，而這些細胞的數量將會決定你這個細胞在某世代的存亡..

以下幾點請注意：
1. 當周邊低於2個(不含2)存活細胞時，他將在下一個世代變成死亡狀態
2. 當周邊有2個或3個存活細胞時，他將在下一個世代保持不變
3. 當周邊有3個存活細胞時，他將在下一個世代變成存活狀態
4. 當周邊超過3個存活細胞時，他將在下一個世代變成死亡狀態

好了，了解這些規則之後，終於輪到你主宰生命的時刻了
請先輸入盤面的長與寬
再輸入你構想的盤面(令存活細胞為1，死亡細胞為0)，創造這個世界
接著請輸入你想要知道幾個世代的變化
最後請輸入你想要知道第哪個世代的變化
廢話不多說，我們開始吧！
```

二. 使用者輸入盤面的長、寬，以及設計盤面

程式碼：

```
System.out.print("請輸入長：");
int length = sc.nextInt();
System.out.print("請輸入寬：");
int width = sc.nextInt();

System.out.println("請輸入盤面：");
int [][] world = new int[length][width];

for(int i=0; i<length; i++) {
    for(int j=0; j<width; j++) {
        System.out.print("您現在輸入的是第"+ (i+1) +"列第" + (j+1) +"行的生命=>");
        world[i][j] = sc.nextInt();

    }
}

System.out.println();
System.out.println("您目前的世界如下：" );
for(int i=0; i<length; i++) {
    for(int j=0; j<width; j++) {
        System.out.print(world[i][j]+" ");
    }
```

輸出結果：

```
請輸入長：5
請輸入寬：7
請輸入盤面：
您現在輸入的是第1列第1行的生命=>0
您現在輸入的是第1列第2行的生命=>0
您現在輸入的是第1列第3行的生命=>1
您現在輸入的是第1列第4行的生命=>0
您現在輸入的是第1列第5行的生命=>0
您現在輸入的是第1列第6行的生命=>0
您現在輸入的是第1列第7行的生命=>0
您現在輸入的是第2列第1行的生命=>1
您現在輸入的是第2列第2行的生命=>0

您現在輸入的是第5列第6行的生命=>0
您現在輸入的是第5列第7行的生命=>0

您目前的世界如下：
0 0 1 0 0 0 0
1 0 1 0 0 1 0
0 1 1 0 0 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
-----------------------------------
```

三. 讓使用者輸入想知道幾個世代，且指定想知道的世代

程式碼：

```
System.out.print("請輸入你想要知道幾個世代：");
int num = sc.nextInt();

for(int k=0; k<num; k++) {
    System.out.println();
    System.out.print("請輸入你想要知道第幾個世代：");
    int gen = sc.nextInt();

    int[][] new_world;
    new_world = new_world(world, length, width);

    for(int g=1; g<gen; g++) {
        new_world = new_world(new_world, length, width);
    }

    System.out.println("以下為您現在的世界：");
    print_new_world(new_world, length, width);
}
```

輸出結果：

```
請輸入你想要知道幾個世代：3

請輸入你想要知道第幾個世代：2
以下為您現在的世界：
0 0 1 0 0 0 0
0 0 0 1 1 1 1
0 1 1 1 1 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0

請輸入你想要知道第幾個世代：1
以下為您現在的世界：
0 1 0 0 0 0 0
0 0 1 1 0 1 1
0 1 1 0 0 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0

請輸入你想要知道第幾個世代：3
以下為您現在的世界：
```

四. 生命棋的運作

建立一個 new_world 的函式，傳入由使用者建立的 world、長與寬。利用多層迴圈去檢視每個生命的周圍其他八個生命的狀態，若為存活的生命，則將該生命的周圍生命存活數 count +1。再依 count 數更新世界的狀態，最後回傳更新後的世界 new_world 至 new_world 函式。

```
public static int[][] new_world(int[][] world, int length, int width) {
    int[][] new_world = new int[length][width];

    //用雙層迴圈方式掃描棋盤每一個點
    for(int i=0; i<length; i++){
        for(int j=0; j<width; j++){
            //以一個算為中心，計算周圍八個點的生命數，用雙層迴圈方式掃描
            int count = 0;
            for(int x=-1; x<=1; x++){
                for(int y=-1; y<=1; y++){
                   //將周圍八個點的生命數(1)加到count裡
                   //(i+x >= 0 && j+y >= 0) && (i+x < length && j+y < width) 為判斷點是否超出棋盤外
                   //!(x == 0 && y == 0) 為排除中心點
                    if((i+x >= 0 && j+y >= 0) && (i+x < length && j+y < width) && !(x == 0 && y == 0)){
                        count += world[i+x][j+y];
                    }

                }
            }

                ,
               //中心點下一次的狀態
                 if(count == 2){ //加總為 2 時等於原樣
                     new_world[i][j] = world[i][j];
                 }
                 else if(count == 3){ //加總為 3 時生出新生命
                     new_world[i][j] = 1;
                 }
                 else{ //其它狀態生命死亡
                     new_world[i][j] = 0 ;
                 }
             }
        }
        return new_world;
    }
```

而在 main 函式中的詢問使用者想知道幾個世代的迴圈當中有以下程式。其先將 new_world 函式中所得出來的第一個 new_world 存入此處建立的二維矩陣，之後再根據使用者想知道的世代數之下，更新世界到使用者指定的世代

```
int[][] new_world;
new_world = new_world(world, length, width);

for(int g=1; g<gen; g++) {
    new_world = new_world(new_world, length, width);
}

System.out.println("以下為您現在的世界：");
print_new_world(new_world, length, width);
```

此為輸出新世界的函示 print_new_world。

```
public static void print_new_world(int[][] new_world, int length, int width) {
    for(int s=0; s<length; s++) {
        for(int r=0; r<width; r++) {
            System.out.print(new_world[s][r]+" ");
        }
        System.out.println();
    }
}
```

5. 結論與心得

再製作這個小遊戲的過程中，首先遇到的就是理解這個遊戲的規則運作。由於該遊戲我在之前從未聽說過，所以花了一段時間去理解它。而接下來遇到的問題就是要如何將腦中所想透過程式化為現實。例如我其實想要使用者輸入的初始世界是直接一整個二維矩陣，而不是像我程式中所寫的，變成讓使用者逐次輸入，然而網路上的相關語法很少所以我還是決定就讓使用者將每個生命逐筆輸入。此外在整個撰寫過程中以下圖片這部分是花費我最多時間去理解的


```
int[][] new_world;
new_world = new_world(world, length, width);

for(int g=1; g<gen; g++) {
    new_world = new_world(new_world, length, width);
}

System.out.println("以下為您現在的世界：");
print_new_world(new_world, length, width);
```

我本來是有撰寫較簡易、未使用函式的版本，但想到不能辜負上課所學還是決定花一點時間重新撰寫一份有用到函式的版本。然而當我呼叫完一次 new_world 函式後，我想將其得出的 new_world 再度傳回 new_world 函式中，所以我寫了

new_world(new_world(world, length, width), length, width)

但發現不管指定要第幾世代，總是只有回傳第一世代的結果。直到後來再檢視一次查詢到的相關範例，靜下心來思考之後才成功的寫出來。雖然可能有更好的寫法，但我依舊很享受思考以及逐漸將程式修改到令自己還算滿意的程度。最後，很感謝老師這學期的教導，簡潔扼要的 ppt 搭配老師詳細的講解，讓我除了學習到一個全新的程式語法外，更學習到撰寫程式的思考邏輯和讀程式的技巧，所以在這門課中著實受益良多，謝謝老師！

(但我也很希望老師可以再多講一些><，我很期待可以學習到更多！)