UNIVERSIDAD POLITECNICA SALESIANA

SEDE CUENCA

CARRERA: *INGENIERIA DE SISTEMAS*

Nombre: Bryam Gabriel Mora Lituma

Materia: Simulacion

Fecha:18/05/2021

Reglas del juego:

Una nueva célula nace cuando:

```
Si una célula está muerta y en la vecindad existen exáctamente tres células vivas se activa
```

Una célula se muere cuando:

```
Sobre población: 4 Células vivas o mas en su vecindad
```

```
Aislamiento: No hay células vivas en su vecindad
```

Cuándo sigue viva

```
Cuando existen 2 o 3 células en su vecindad
```

In [1]:

```
import copy
import random
import itertools
import time
import os
```

In [6]:

```
UNIX = False

clear = lambda: os.system('clear') if UNIX else os.system('cls')


class GameOfLife(object):

    def __init__(self, rows, cols):

        self.rows = rows
        self.cols = cols

        row_life = lambda: [random.randint(0, 1) for n in range(self.cols)]
        self.game = [row_life() for n in range(self.rows)]

        self.life = 1
        self.dead = 1

    def __str__(self):

        table = ''
        for row in self.game:
            for cell in row:
                table += '@ ' if cell else '. '
            table += '\n'

        table += "Viven: {0} Mueren: {1}".format(self.life, self.dead)
        return table

    def evaluate(self, row, col):

        distance = list(set(itertools.permutations([-1, -1, 1, 1, 0], 2)))
        into_table = lambda x, y: (x in range(self.rows) and y in range(self.cols))

        total = 0
        for r, c in distance:
            if into_table(r + row, c + col):
                total += self.game[r + row][c + col]
        return total

    def test(self):

```

```
        gameaux = copy.deepcopy(self.game)
        self.life = 0
        self.dead = 0

        for r in range(self.rows):
            for c in range(self.cols):
                total = self.evaluate(r, c)

                if (total < 2 or total > 3) and gameaux[r][c]:
                    gameaux[r][c] = 0
                    self.dead += 1
                elif total == 3 and not gameaux[r][c]:
                    gameaux[r][c] = 1
                    self.life += 1

        self.game = copy.deepcopy(gameaux)

rows, cols = int(input("Filas>> ")), int(input("Columnas>> "))

game = GameOfLife(rows, cols)

iterations = 0
while game.life > 0 or game.dead > 0:
    try:
        clear()
        game.test()
        print(game)
        time.sleep(1)
        iterations += 1
    except KeyboardInterrupt:
        break
print("Total: ", iterations)
```

```
Filas>> 4
Columnas>> 4
. . . .
@ . . .
. @ . @
. . . @
Viven: 1 Mueren: 5
. . . .
. . . .
. . @ .
```

```
. . @ .
Viven: 2 Mueren: 4
. . . .
. . . .
. . . .
. . . .
Viven: 0 Mueren: 2
. . . .
. . . .
. . . .
. . . .
Viven: 0 Mueren: 0
Total:  4
```

In [ ]:
```
1
```

In [ ]:
```
1
```