UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

ALUMNO:

PAZ MALDONADO CARLOS SAÚL

NOMBRE DEL PROFESOR:

HERNANDEZ CABRERA JESUS

NOMBRE DE LA MATERIA:

ESTRUCTURA DE DATOS

FECHA DE ENTREGA:

17 de septiembre del 2024

TAREA NO. 6.

1. Instrucciones de la tarea.

Criterios de evaluación.
-El juego usa una rejilla rectangular (SE DEBE EMPLEAR LA CLASE Array2d.java) de tamaño infinito, donde cada celda puede tomar uno de
dos valores: hospedar una célula viva o una célula muerta. Las 8 celdas que
rodean una célula son las células vecinas.

El juego se juega por un periodo de tiempo determinado por el número de generaciones totales a calcular.

Se empieza el juego con una configuración inicial de población y se calcula la siguiente generación con base a las siguientes reglas:

1.Si una célula está viva y tiene 2 o 3 vecinos vivos, la célula sobrevive en la siguiente generación.

2. Las células que tienen 1 o 0 vecinos vivos, muere por soledad.

3.Una célula viva que tiene 4 o más vecinos vivos, muere por sobrepoblación.

4.Una célula muerta que tiene exactamente 3 vecinos vivos resulta en un nacimiento en la siguiente generación. El resto de las células muertas permanecen en el mismo estado para la siguiente generación.

NOTA:

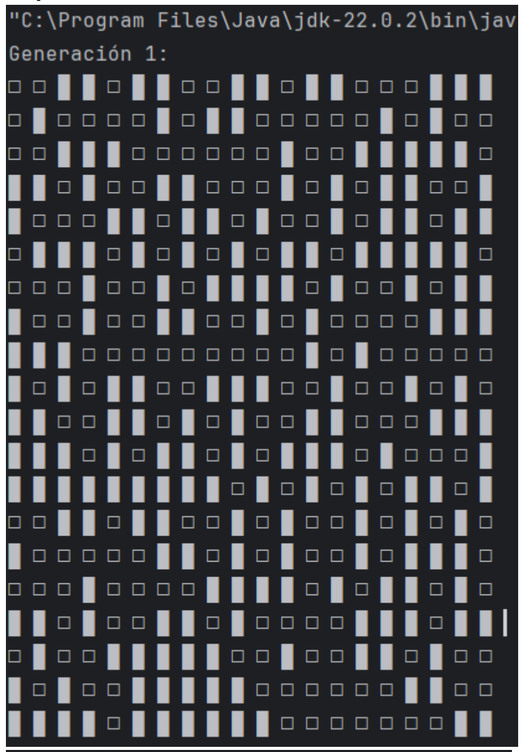
Realice el algoritmo para crear simular el juego de la vida; sin embargo, la cuadricula no es infinita y aunque me falte realizar ese apartado el programa funciona de la siguiente manera:

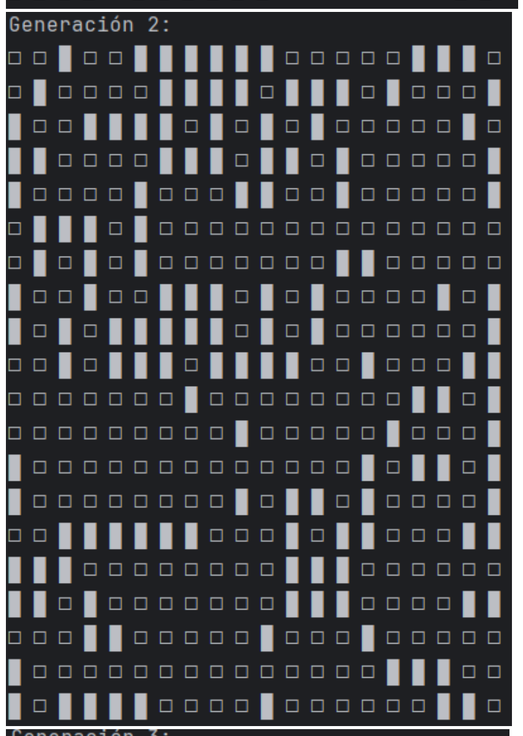
En el método main se puede modificar el tamaño de la cuadrícula y las generaciones que se requiere calcular.

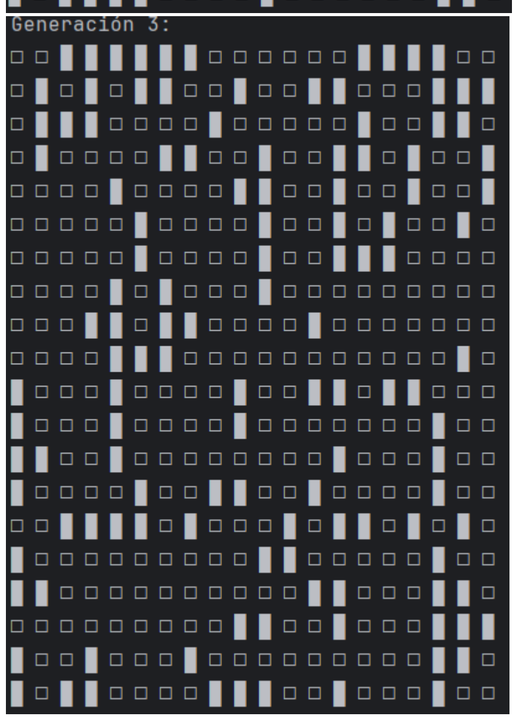
Utilice una clase llamada consola, que permite realizar la impresión de la cuadrícula y comenzar el juego de la vida, con impresión en la terminal.

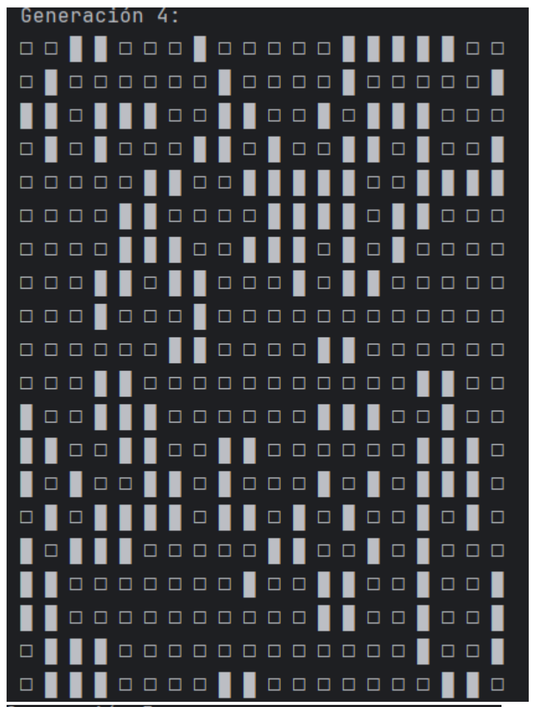
Un problema que existe en el código actual, específicamente en el commit 315b23d… es que los valores de la rendija tienen valores aleatorios cada vez que se ejecuta el juego, por lo que estaré trabajando en la implementación. Si existe algún otro commit posterior a este, probablemente ya haya solucionado el problema, pero, por lo tanto, entrego lo anteriormente escrito.

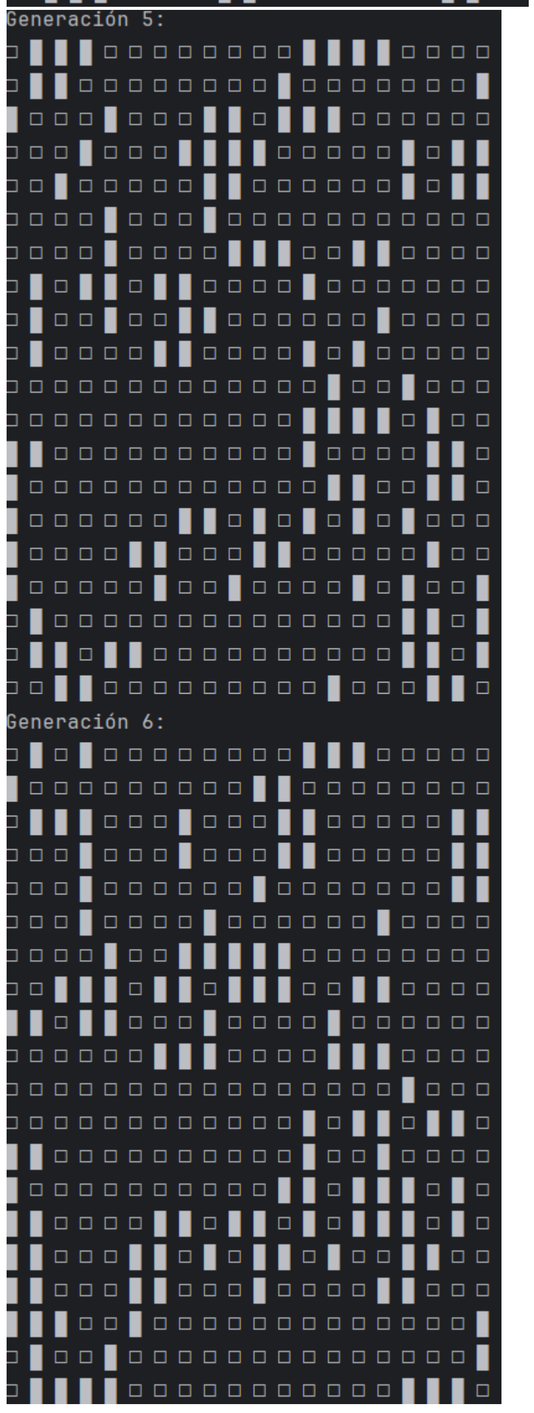
2. **Capturas de la consola.**

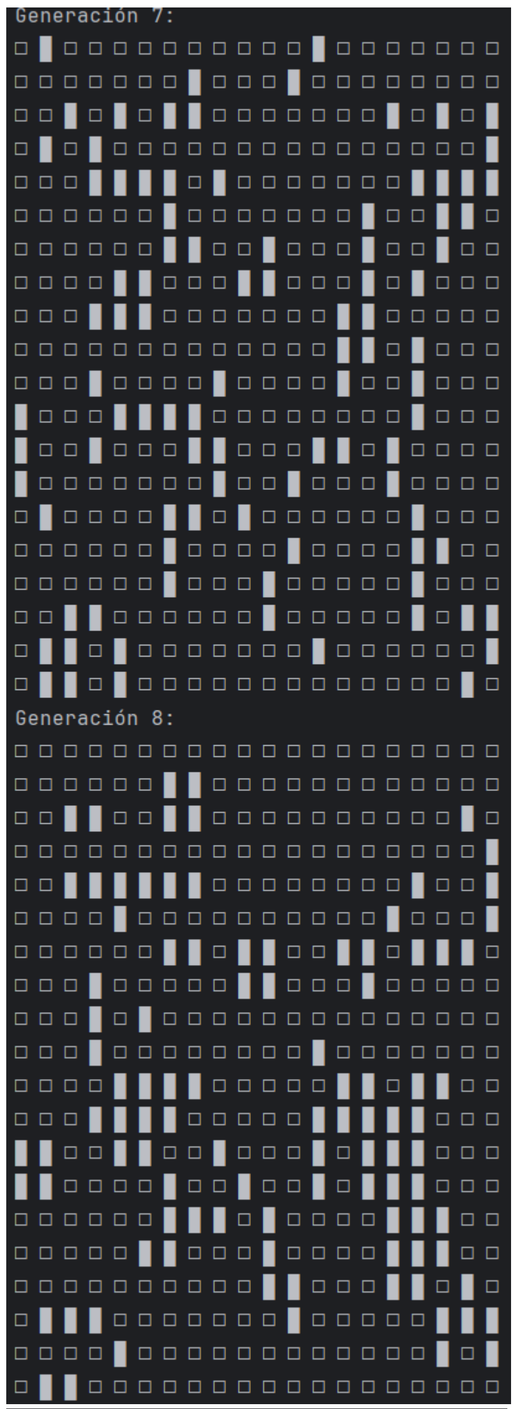


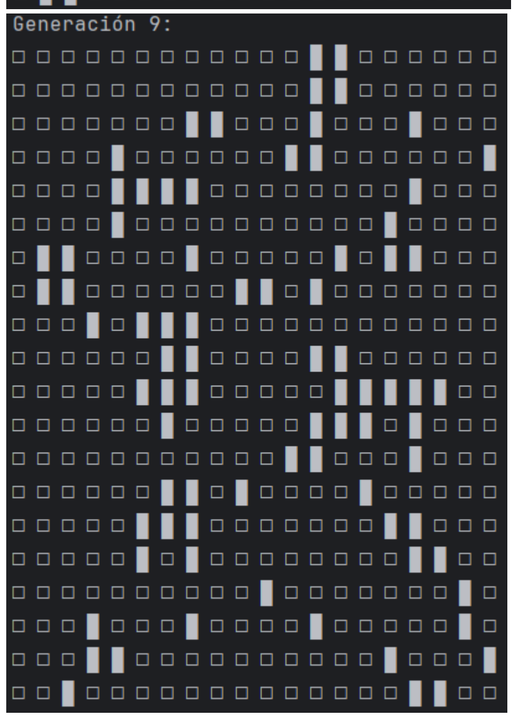


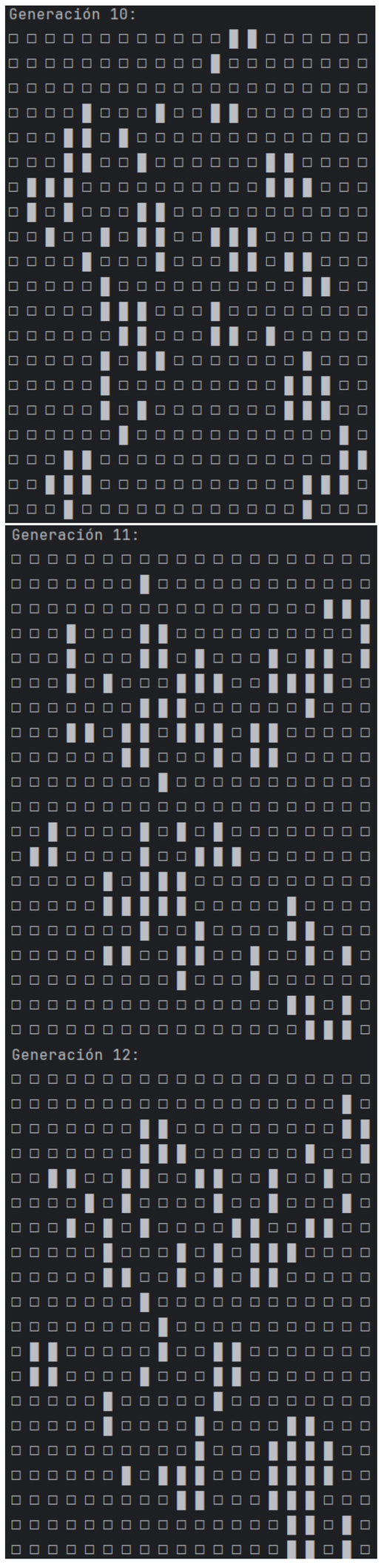




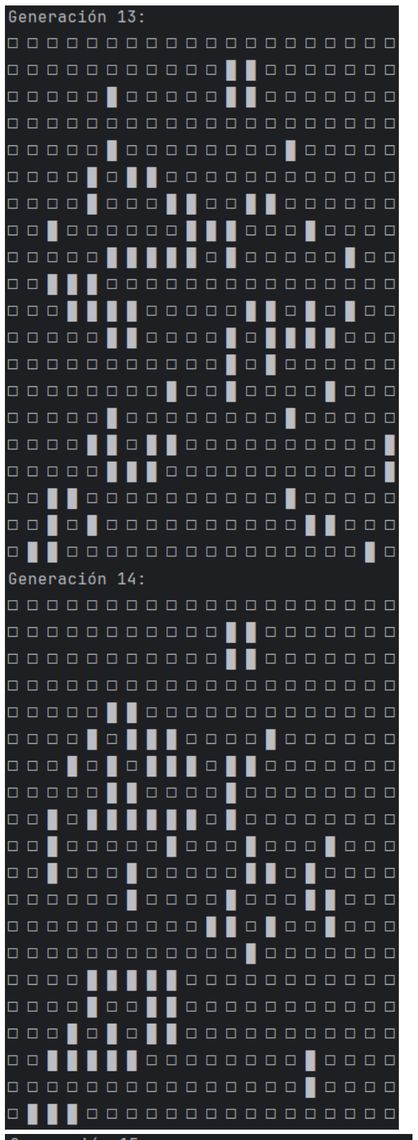






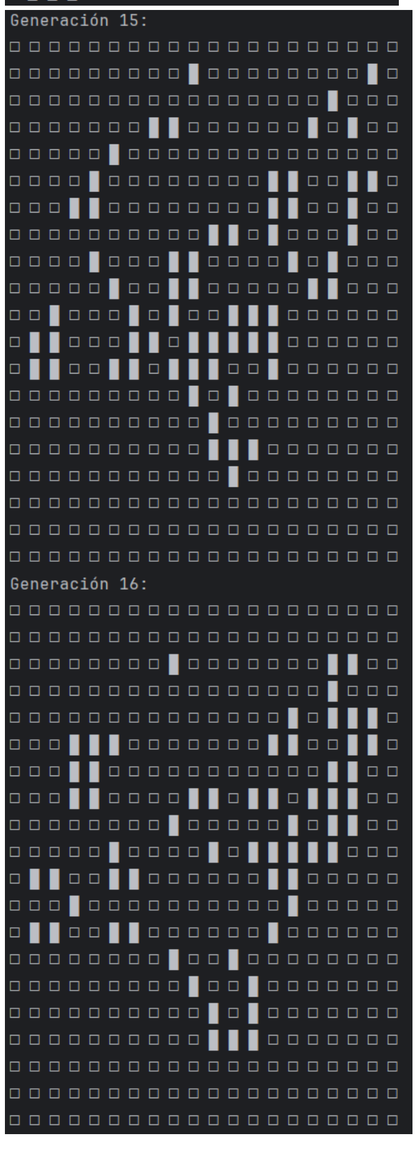


Generación 13:



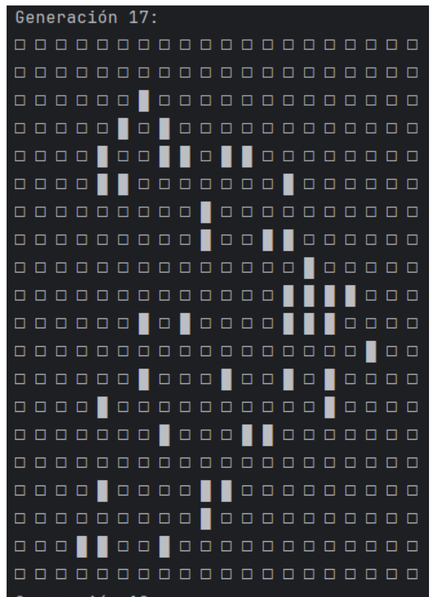
Generación 14:

Generación 15:



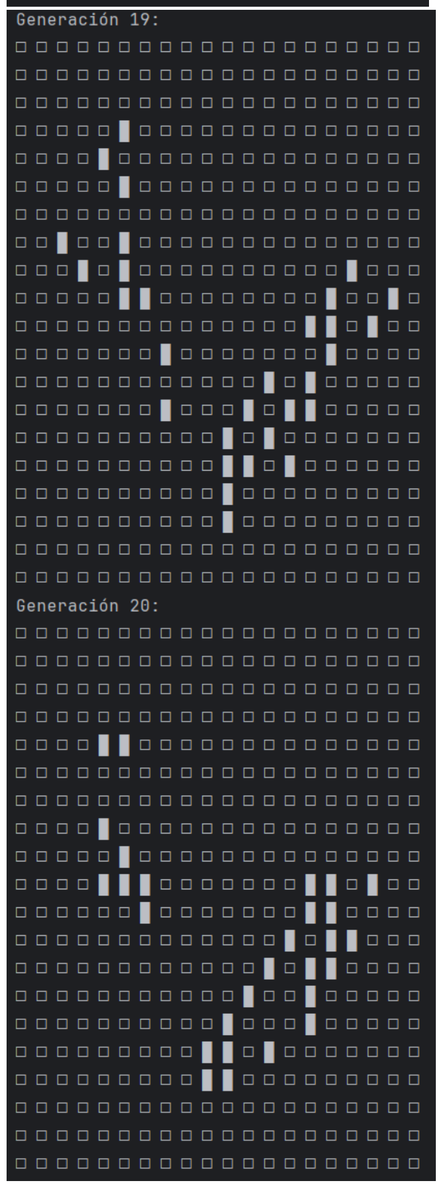
Generación 16:

Generación 17:



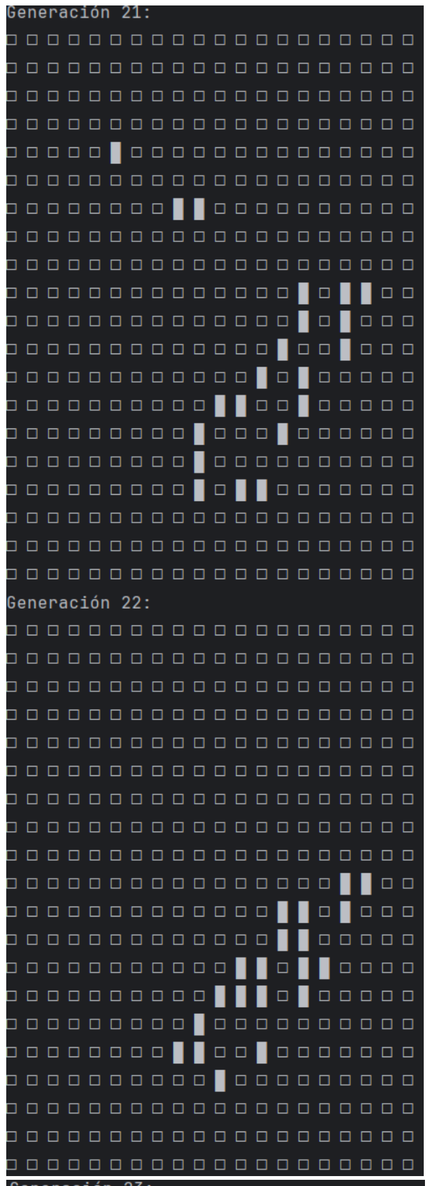
Generación 18:

Generación 19:



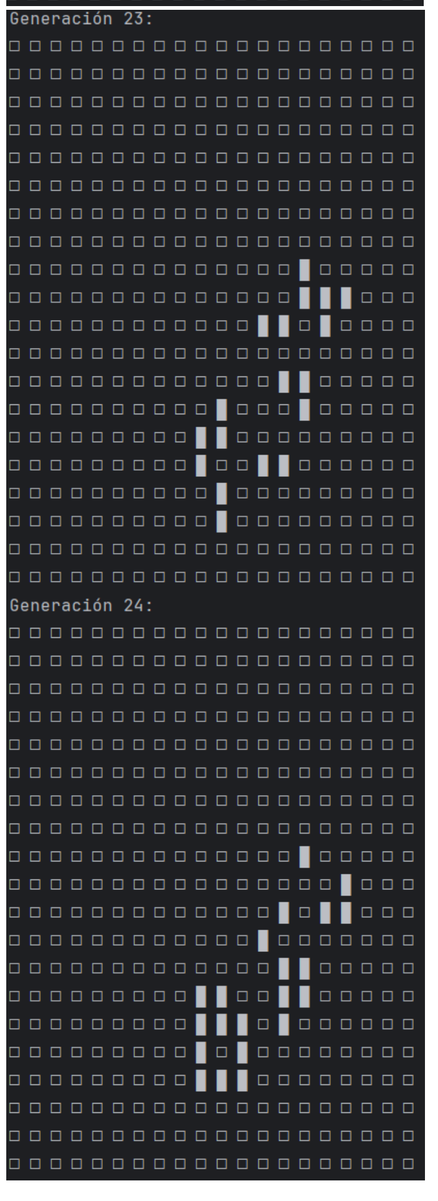
Generación 20:

Generación 21:



Generación 22:

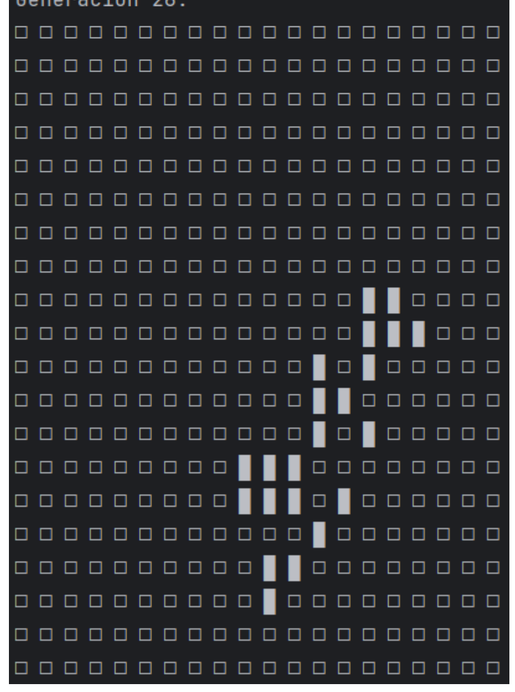
Generación 23:



Generación 24:

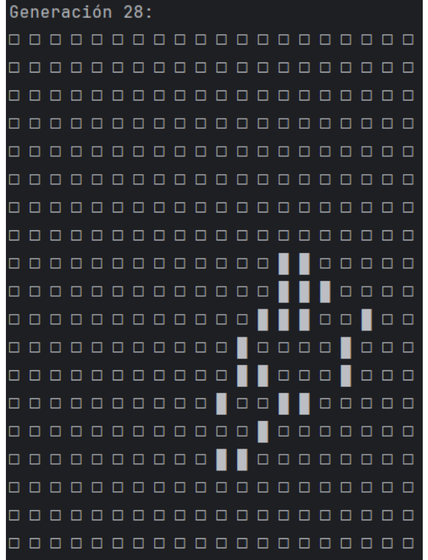
Generación 25:

Generación 26:



Generación 27:

Generación 28:



Generación 29:

Generación 30:

3. Capturas del código:
    a. Clase JuegoDeLaVida

```
package unam.fesaragon.estructurasdatos;

import java.util.ArrayList;

public class JuegoDeLaVida {
    //Alumno:Carlos Saul Paz Maldonado
    private ArrayList<Coordenada>
desplazamientosVecinos = new ArrayList<>();
    private ADTArray2D<Celula>
generacionActual;
    private ADTArray2D<Celula>
siguienteGeneracion;

    public JuegoDeLaVida(ADTArray2D<Celula>
generacionActual) {
        this.generacionActual =
generacionActual;
        this.siguienteGeneracion = new
ADTArray2D<>(generacionActual.getRow(),
generacionActual.getCol());

cargarCoordenadas(generacionActual.getRow()
, generacionActual.getCol());

iniciarlizarCelulasDeSiguienteGeneracion();
    }

    private void
iniciarlizarCelulasDeSiguienteGeneracion()
{
        // Inicializar las células en la
siguiente generación
        for (int fila = 0; fila <
siguienteGeneracion.getRow(); fila++) {
            for (int columna = 0; columna <
siguienteGeneracion.getCol(); columna++) {

siguienteGeneracion.set_item(fila, columna,
new Celula(fila, columna));
            }
        }
    }

    private void cargarCoordenadas(int
filas, int columnas) {
        // Reducir en 1, empieza en 0 el
index de la cuadrícula
        filas--;
        columnas--;
        // Coordenadas de los vecinos
        desplazamientosVecinos.add(new
Coordenada(-1, -1));
        desplazamientosVecinos.add(new
Coordenada(-1, 0));
        desplazamientosVecinos.add(new
Coordenada(-1, 1));
        desplazamientosVecinos.add(new
Coordenada(0, -1));
        desplazamientosVecinos.add(new
Coordenada(0, 1));
        desplazamientosVecinos.add(new
Coordenada(1, -1));
        desplazamientosVecinos.add(new
Coordenada(1, 0));
        desplazamientosVecinos.add(new
Coordenada(1, 1));

    }

    public void actualizarGeneracion() {
        actualizarCelulas(generacionActual,
siguienteGeneracion);
        ADTArray2D<Celula> temp =
generacionActual;
        generacionActual =
siguienteGeneracion;
        siguienteGeneracion = temp;
    }

    private void
actualizarCelulas(ADTArray2D<Celula>
generacionAActualizar, ADTArray2D<Celula>
celdaAGuardar) {
        for (int fila = 0; fila <
generacionAActualizar.getRow(); fila++) {
            for (int columna = 0; columna <
generacionAActualizar.getCol(); columna++)
{

actualizarSiguienteGeneracion(generacionAAc
tualizar.get_item(fila, columna),

celdaAGuardar.get_item(fila, columna));
            }
        }
    }

    private void
actualizarSiguienteGeneracion(Celula
celula, Celula celulaAGuardar) {
        boolean estaViva =
celula.isEstaVivo();
        int vecinosVivos =
cuantosVecinosVivosTiene(celula);
        celulaAGuardar.setEstaVivo(false);
        if (estaViva && (vecinosVivos == 2
|| vecinosVivos == 3)) {

celulaAGuardar.setEstaVivo(true);
        } else if (!estaViva &&
vecinosVivos == 3) {

celulaAGuardar.setEstaVivo(true);
        }
    }

    private int
cuantosVecinosVivosTiene(Celula celula) {
        int vecinosvivos = 0;
        vecinosvivos =
contarVecinosVivos(celula);
        return vecinosvivos;
    }

    private int contarVecinosVivos(Celula
celula) {
        int vecinosVivos = 0;
        for (Coordenada desplazamiento :
desplazamientosVecinos) {
            int nuevaFila =
celula.getFila() +
desplazamiento.getFila();
            int nuevaColumna =
celula.getColumna() +
desplazamiento.getColumna();
            if (esCeldaValida(nuevaFila,
nuevaColumna) &&
generacionActual.get_item(nuevaFila,
nuevaColumna).isEstaVivo()) {
                vecinosVivos++;
            }
        }
        return vecinosVivos;
    }

    private boolean esCeldaValida(int fila,
int columna) {
        boolean res = false;
        if (fila >= 0 && fila <
generacionActual.getRow() && columna >= 0
&& columna < generacionActual.getCol()) {
            res = true;
        }
        return res;
    }

    public ADTArray2D<Celula>
getGeneracionActual() {
        return generacionActual;
    }
}
```

    b. Clase Consola

```
package unam.fesaragon.estructurasdatos;

import java.util.Random;

public class Consola {
    // Metodo para inicializar la
cuadrícula con estados aleatorios
    private static ADTArray2D<Celula>
inicializarCuadricula(int filas, int
columnas) {
        ADTArray2D<Celula> cuadricula = new
ADTArray2D<>(filas, columnas);
        Random random = new Random();
        for (int fila = 0; fila < filas;
fila++) {
            for (int col = 0; col <
columnas; col++) {
                boolean estadoInicial =
random.nextBoolean();
                cuadricula.set_item(fila,
col, new Celula(estadoInicial, col, fila));
            }
        }
        return cuadricula;
    }

    private static void
imprimirCuadricula(ADTArray2D<Celula>
cuadricula) {
        for (int fila = 0; fila <
cuadricula.getRow(); fila++) {
            for (int col = 0; col <
cuadricula.getCol(); col++) {
                Celula celula =
cuadricula.get_item(fila, col);

System.out.print(cuadricula.get_item(fila,
col) + " ");
            }
            System.out.println();
        }
    }

    private static void esperar(int
milisegundos) {
        try {
            Thread.sleep(milisegundos);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void iniciar(int filas,
int columnas, int generaciones) {
        ADTArray2D<Celula> cuadricula =
inicializarCuadricula(filas, columnas);
        JuegoDeLaVida juego = new
JuegoDeLaVida(cuadricula);
```

```
        for (int i = 0; i < generaciones;
i++) {
            System.out.println("Generación
" + (i + 1) + ":");

imprimirCuadricula(juego.getGeneracionActua
l());
            juego.actualizarGeneracion();
            esperar(500);
        }
    }
}
```

c. Clase ADTArray2D

```
package unam.fesaragon.estructurasdatos;

public class ADTArray2D<T> {
    private int re;
    private int col;
    private T array[][];

//  Constructor de la clase
    public ADTArray2D(int re, int col) {
        this.re = re;
        this.col = col;
        this.array = (T[][]) new
Object[re][col];
    }
//  Metodo limpiar
    public void clear(T dato) {
        for (int i = 0; i < re; i++) {
            for (int j = 0; j < col; j++) {
                this.array[i][j] = dato;
            }
        }
    }

//  Colocar item en la coordenada
específica
    public void set_item(int renglon, int
columna, T dato){
        this.array[renglon][columna] =
dato;
    }

//   Obtener item en la coordenada
específica
    public T get_item(int renglon, int
columna){
        return
this.array[renglon][columna];
    }

    public int getRow() {
        return re;
    }

    public int getCol() {
        return col;
    }

    @Override
    public String toString() {
        String matriz= "";
        for (T[] elemento : this.array) {
            for (T el : elemento) {
                matriz= matriz+el+" ";
            }
            matriz+="\n";
        }
        return matriz;
    }
}
```

d. Clase Celula

```
package unam.fesaragon.estructurasdatos;

public class Celula {
    private boolean estaVivo;
    private int columna;
    private int fila;

    public Celula(boolean estaVivo, int
columna, int fila) {
        this.estaVivo = estaVivo;
        this.columna = columna;
        this.fila = fila;
    }

    public Celula(int columna, int fila) {
        this.columna = columna;
        this.fila = fila;
        this.estaVivo = false;
    }

    public Celula() {
        this.estaVivo=false;
    }

    public boolean isEstaVivo() {
        return estaVivo;
    }

    public void setEstaVivo(boolean
estaVivo) {
        this.estaVivo = estaVivo;
    }

    public int getColumna() {
        return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    public int getFila() {
        return fila;
    }

    public void setFila(int fila) {
        this.fila = fila;
    }

    @Override
    public String toString() {
        return estaVivo ? "■" : "□";

    }
}
```

e. Clase Coordenada

```
4. package
   unam.fesaragon.estructurasdatos;

   import java.util.ArrayList;

   public class Coordenada {
       private int fila;
       private int columna;
       private static final int[][]
   DESPLAZAMIENTOS_VECINOS = {
               {-1, -1}, {-1, 0}, {-1,
   1},  // Vecinos superiores
               {0, -1},           {0, 1},
   // Vecinos laterales
               {1, -1},  {1, 0},  {1, 1}
   // Vecinos inferiores
       };

       public Coordenada(int fila, int
   columna) {
           this.fila = fila;
           this.columna = columna;
       }

       public Coordenada() {
       }

       public ArrayList<Coordenada>
   obtenerVecinos() {
           ArrayList<Coordenada> vecinos
   = new ArrayList<>();

           for (int[] desplazamiento :
   DESPLAZAMIENTOS_VECINOS) {
               int nuevoX = this.fila +
   desplazamiento[0];
               int nuevoY = this.columna
   + desplazamiento[1];
               vecinos.add(new
   Coordenada(nuevoX, nuevoY));
           }

           return vecinos;
       }

       public int getFila() {
           return fila;
       }

       public int getColumna() {
           return columna;
       }

       public void setFila(int fila) {
           this.fila = fila;
       }

       public void setColumna(int
     columna) {
           this.columna = columna;
       }
   }
```

a. Clase Main

```
package unam.fesaragon.estructurasdatos;

public class Main {
    public static void main(String[] args)
{
        int filas = 20;
        int columnas = filas;
        int generaciones = 30;

        Consola.iniciar(filas, columnas,
generaciones);
    }
}
```