**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Herencia e interfaces**

**ADEMAS Java desde consola**

**2020-1**

**Laboratorio 3/6**

**Conociendo:**

1.

a. Tiene dos package el de presentacion y el de aplicacion

b. Posee 4 clases =

-AutomataGUI

-FotoAutomata

-AutomataCelular

-Celula

Y una interfaz llamada Elemento

4 tienen fuentes:

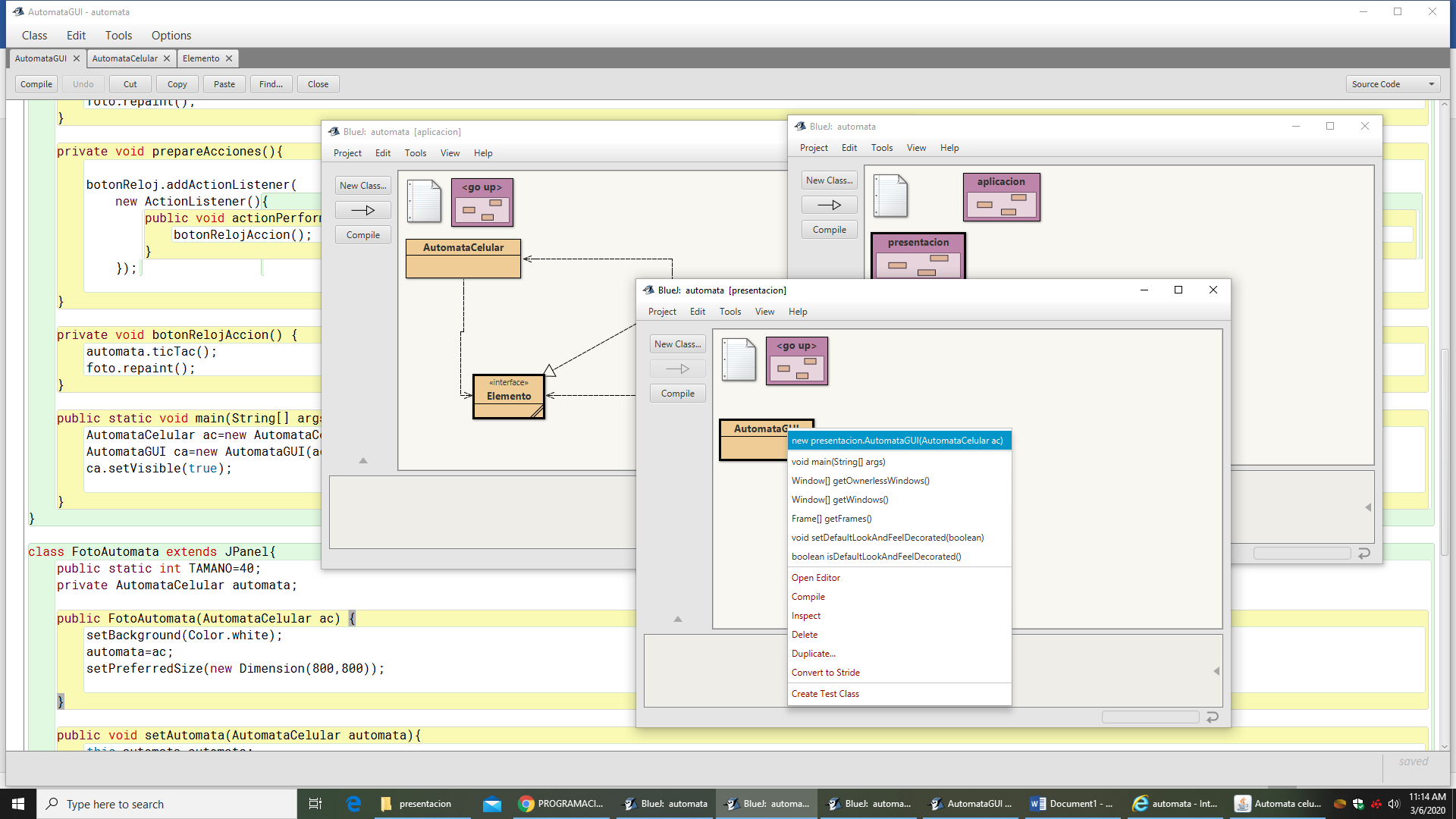
-AutomataGUI

-AutomataCelular

-Celula

-Elemento

c. AutomataGUI, porque esta clase es la que ejecuta las que se encuentran en el paquete Aplicacion.

2. Solo se dibuja una cuadricula vacia ya que los otros metodos no se estan ejecutando

**Arquitectura General**

**Package:** paquete, sirve para facilitar la modularidad del codigo; Las clases se agrupan en unidades de un nivel superior, los paquetes, que actuan como ambitos de contencion de tipos

**Import:** importar, con esta clausula, cualquier modulo de codigo puede hacer referencia a tipos definidos en otros paquetes**.**

El uso d los paquetes en este programa facilita la agrupacion de clases de acuerdo a su function.

1. El directorio de trabajo contiene subdirectorios que hacen referencia a los paquetes que posee el Proyecto y cada uno de estos contiene sus respectivos html;Las coincidencias que encontramos es que cada directorio contiene el html de cada paquete.

**Arquitectura detallada**

**2.** Deben estar en el paquete de aplicación, porque desde estepaquete se acceden a los elementos a los cuales se les realizan las pruebas. Esta asociado a la clase AutomataCelular, porque esta es la que maneja los diferentes elementos.

**Ciclo 1. Iniciando con las células normales**

**3.** AutomataCelular alberga los elementos en una matriz, si puede recibir células porque estas implementan Elemento.

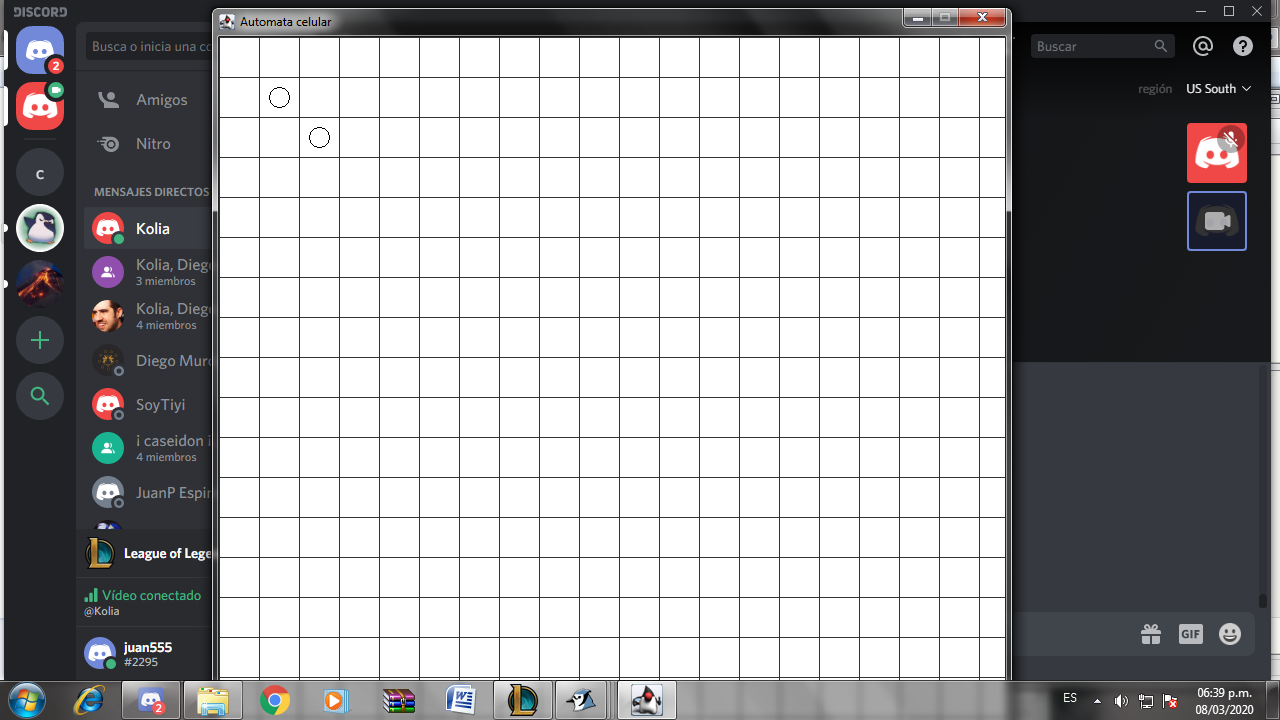
4. La definine la clase Elemento ya que esta la implementa, y posee un atributo de AutomataCelular.

**5.** Las células saben obtener su fila, columna y color, saben si están vivas o no, saben su edad, y saben decidir, saben hacer lo que hace un elemento ya que Celula esta implementando Elemento. Debe aprender a decidir y a cambiar, porque cada celula es autónoma.

**6.**Por comportarse como un elemento las células saben decidir, cambiar, obtener la forma, obtener color y saber si esta vivo.

**7.** La celula es de color negro, lo decide gracias a que esta definido en el constructor, y cambia la edad y su estado actual.

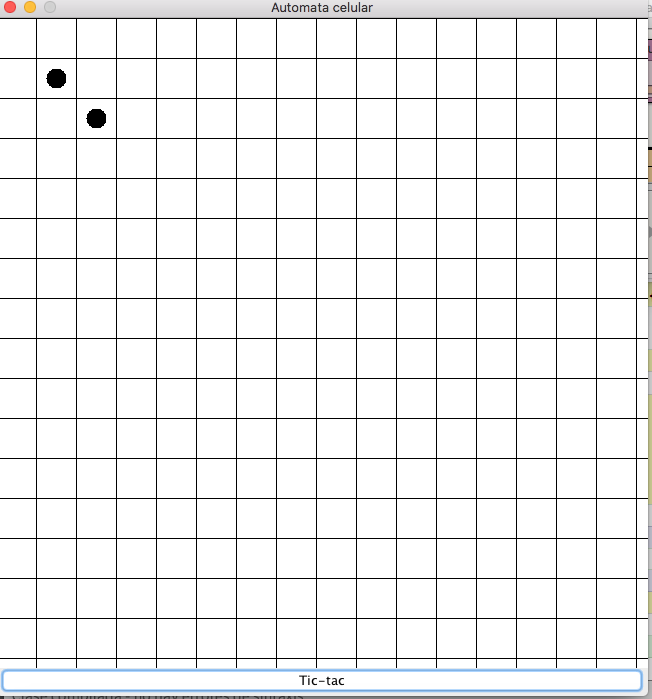
**8.**

****

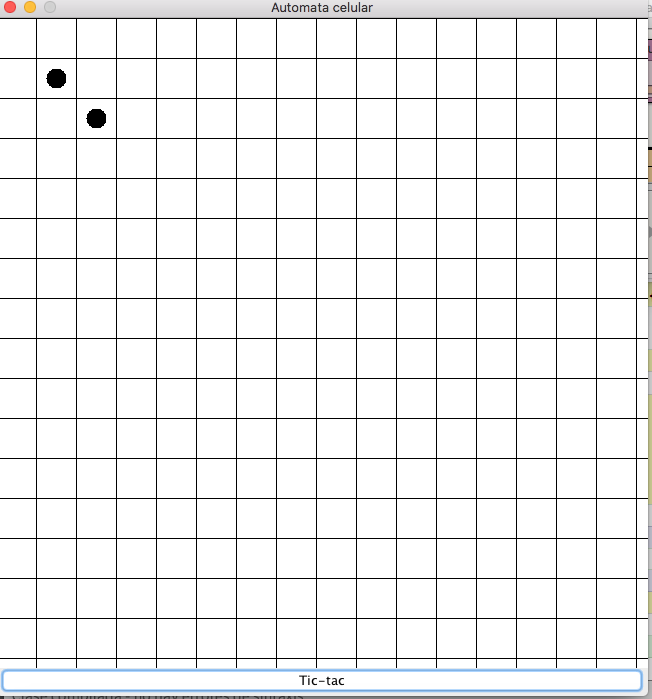
**9.** Depues del primer y segundo TicTac indiana y OO7 quedarian vivas, en el tercer TicTac se morirían, ya que su edad es 2.

**10.**

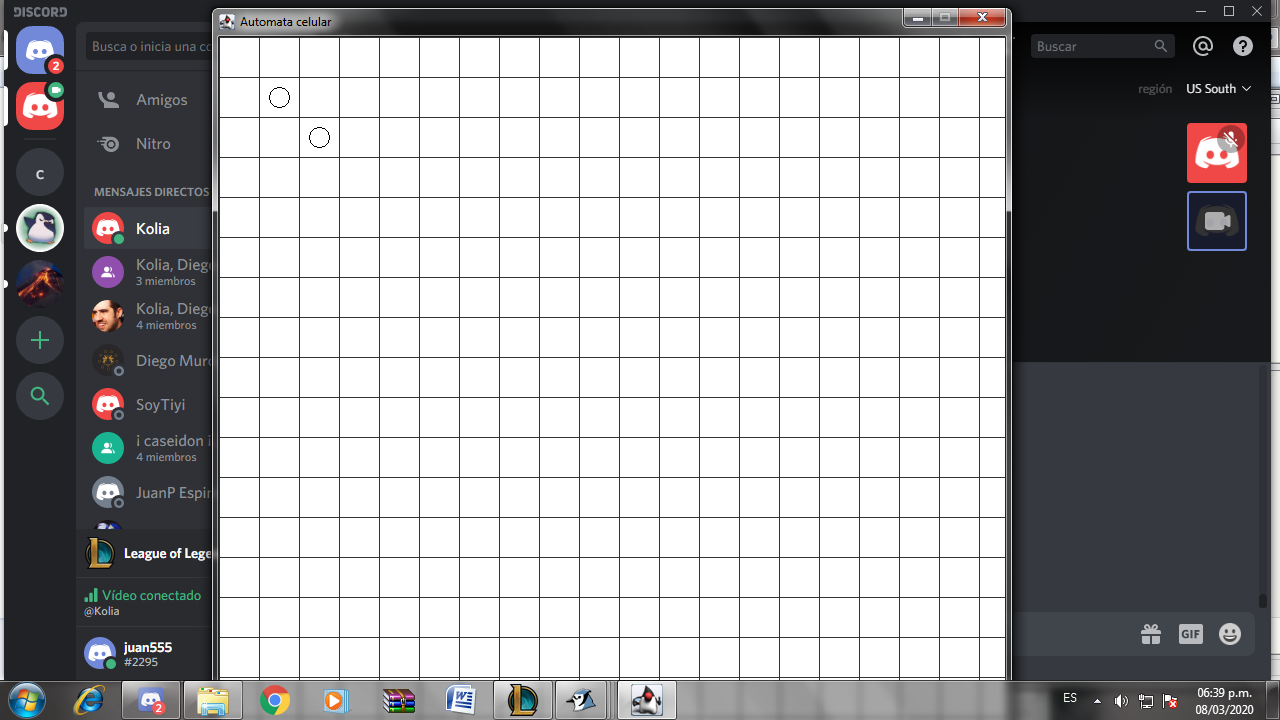
**11.** **Primer clic:**



Segundo clic:



Tercer clic:

****

**Ciclo 2. Incluyendo a las células izquierdosas**

**4.** Primer clic: se muere la célula que está a la izquierda y la célula que queda aumenta su edad en uno; Segundo clic: la célula que queda aumenta su edad en uno; Tercer clic: la célula que queda muere.

**5.** Las adiciones necesarias son: una herencia y un método. Se sobrescribió el método decida de la clase célula.

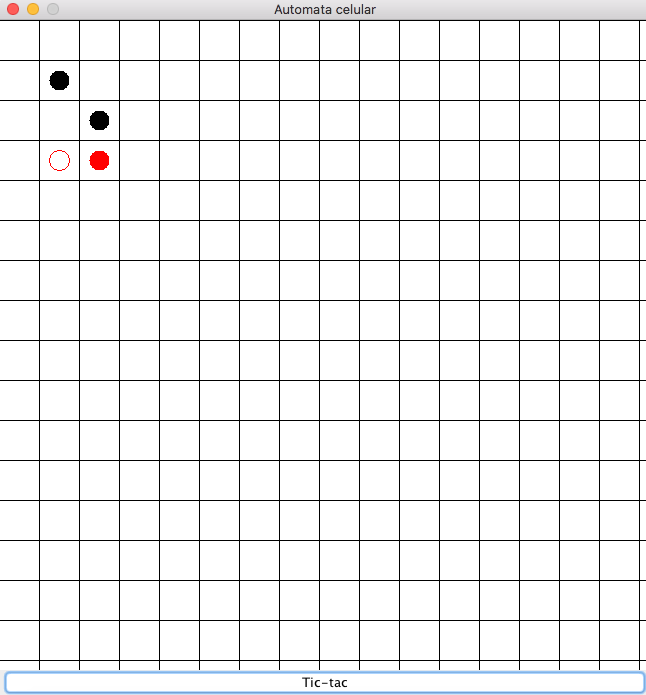
**6.** No, porque la clase izquierdosa hereda de célula y como ésta a su vez implementa elemento, quiere decir que izquierdosa es de tipo elemento y en autómata celular se aceptan los objetos que se comportan como elementos.

**7.** Despues de uno, la celula de la izquierda moriría y la de la derecha vive, después de dos la celula de la derecha vive, después de tres ambas terminan muertas.

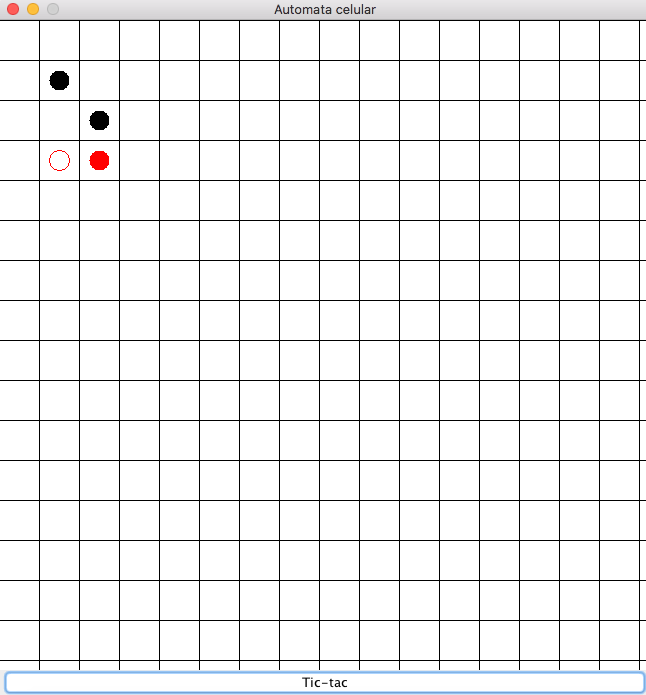
**8.**

**9.**

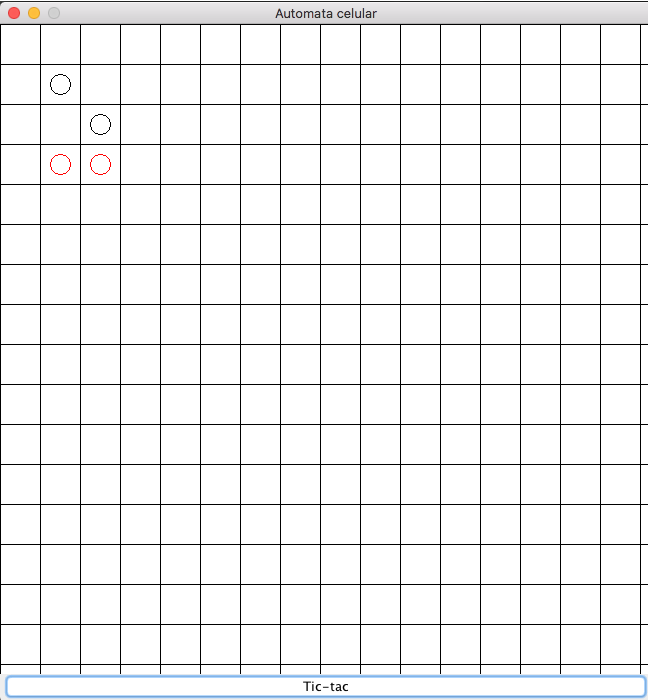
Primer clic:



Segundo clic:

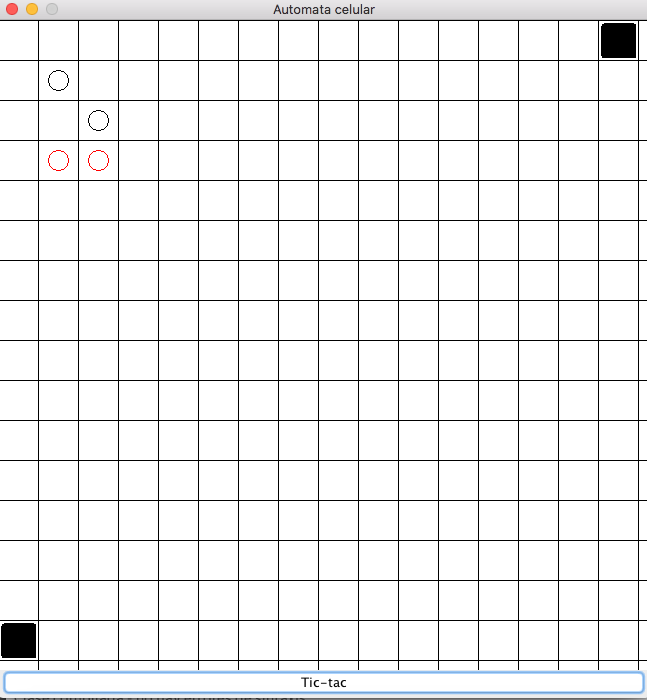


Tercer clic:



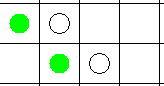
**Ciclo 3. Adicionando una barrera**

1. Creamos la clase barrera como una implementacion de Elemento, colocando los atributos escenciales para las barreras, y haciendo un Override del metodo de getForma() de Elemento.
2. No, porque la clase barrera implementa elemento, quiere decir que barrera es de tipo elemento y en autómata celular se aceptan los objetos que se comportan como elementos.
3. Despues de uno, dos y tres tic tac las barreras quedan iguales.
4. Despues de los tres clics lo unico que cambian son las celulas.



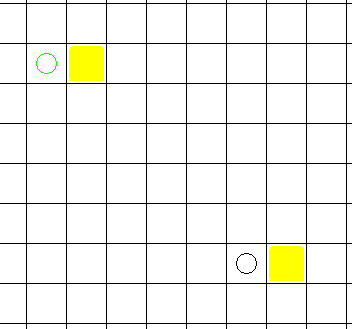
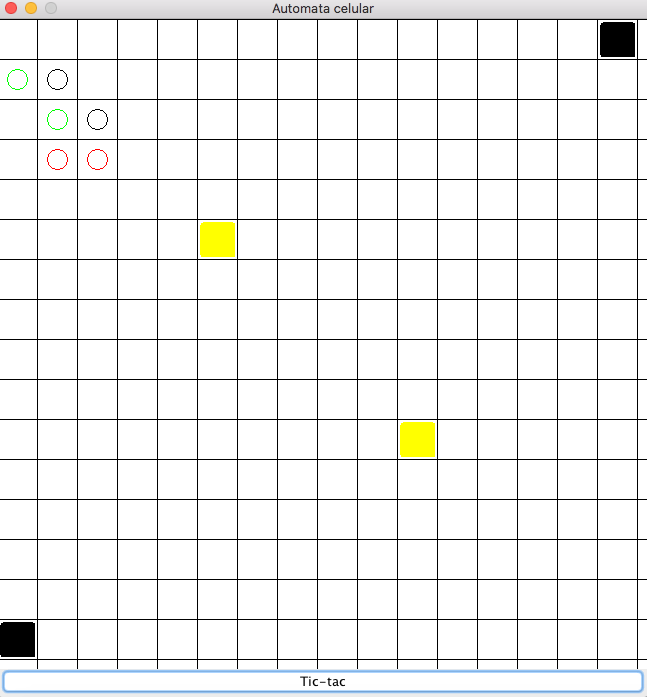
**Ciclo 4. Nueva Celula: Proponiendo y diseñando**

1. Células cancerígenas, estas células siempre estarán vivas, de color verde, y matan a las células que están ubicadas a la derecha de ella.
2. Se colocaron las dos células cancerígenas a la izquierda de dos células normales, lo que causa que estas ultimas estén muertas.

****

**Ciclo 5. Nuevo elemento: Proponiendo y diseñando**

1. Barrera aleatoria, de color amarillo, la cual genera células aleatoria a su izquierda.
2. Se muestran las dos barreras en el autómata, y al hacer clic se genera una celula aleatoria al lado izquierdo de la barrera.

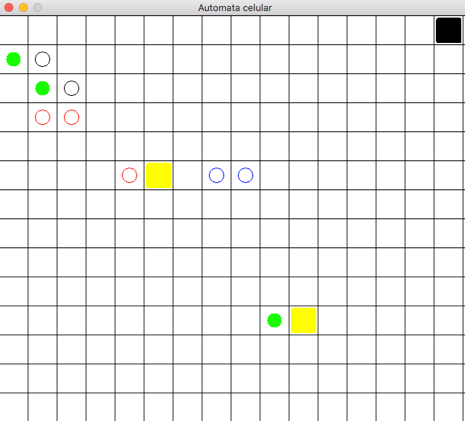


**Caso 6. El Juego de la vida**

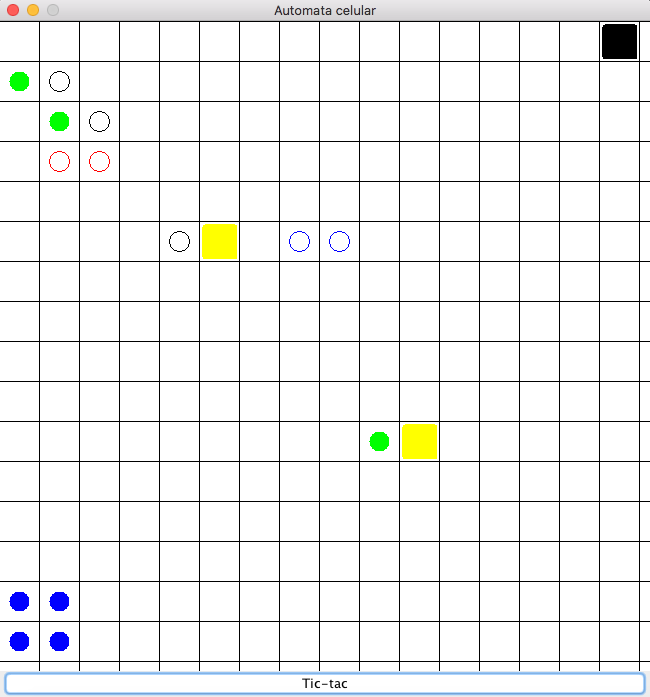
3. En el primer clic ambas celulas se mueren, ya que solo tienen un vecino, y en los otros dos clics permanecen muertas.

4. Se adiciona una nueva herencia de celula, a la cual le modificamos el metodo decida(), y le agregamos otros metodos.

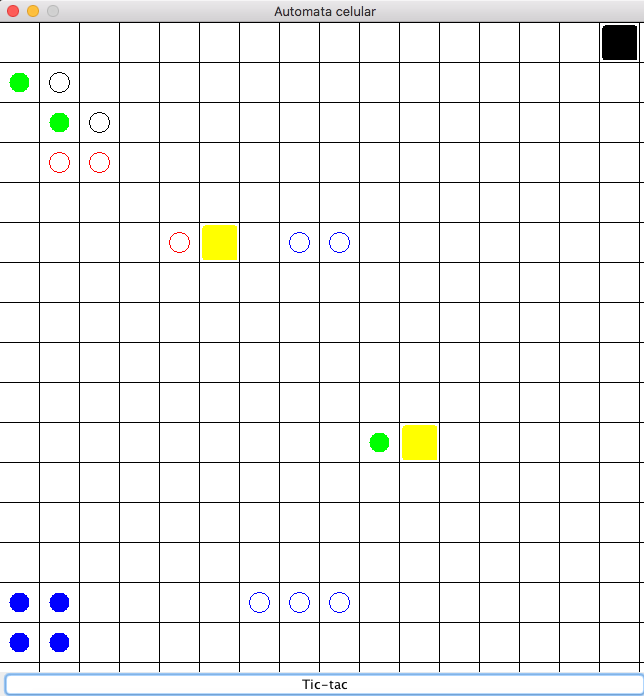
5. Ambas celulas terminan muertas desde el primer clic.



6. Las celulas permanecen vivas.



7.



**Empaquetando la versión final para el usuario.**

2. java -jar nombre archivo.jar

3. Seguridad, disminucion en tiempo de descarga, comprension, portabilidad

**DE BLUEJ A CONSOLA**

**Comandos básicos del sistema operativo**

1. Crear: mkdir

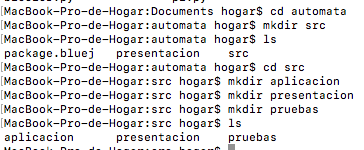
Borrar: rmdir

Listar su contenido: ls

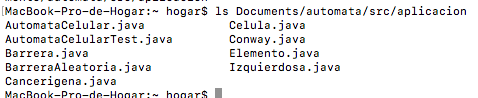
Copiar un archivo: cp

Eliminar un archivo: rm









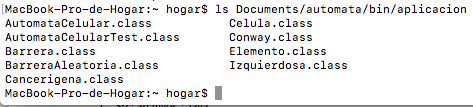
**Estructura de proyectos java**

1. Src: archivos \*.java

Bin: archivos \*.class

Docs: archivos \*.html

1. Todos los archivos \*.class que se encuentran en aplicación



**Comandos de java**

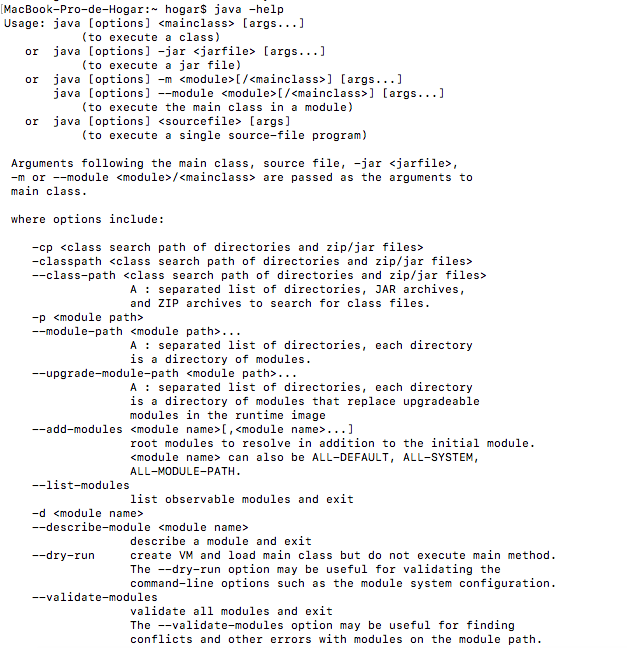
1. javac: es el compilador de java.

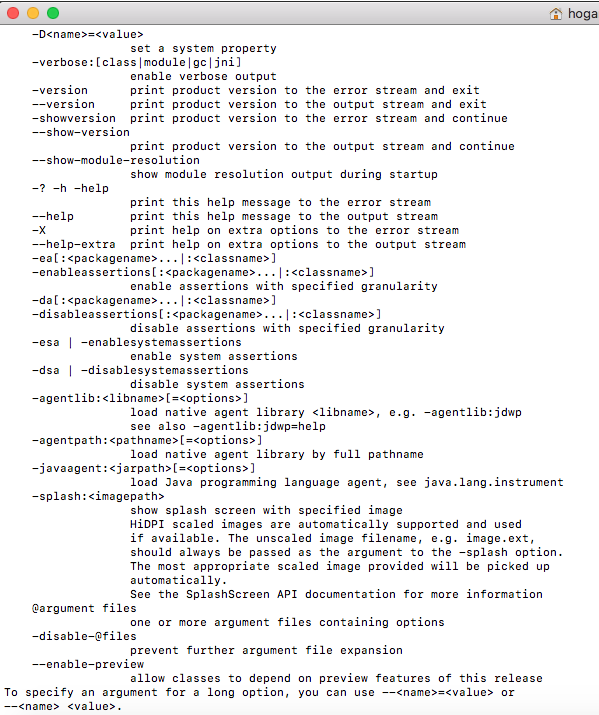
java: es un interfaz simple para acceder a la Maquina Virtual Java.

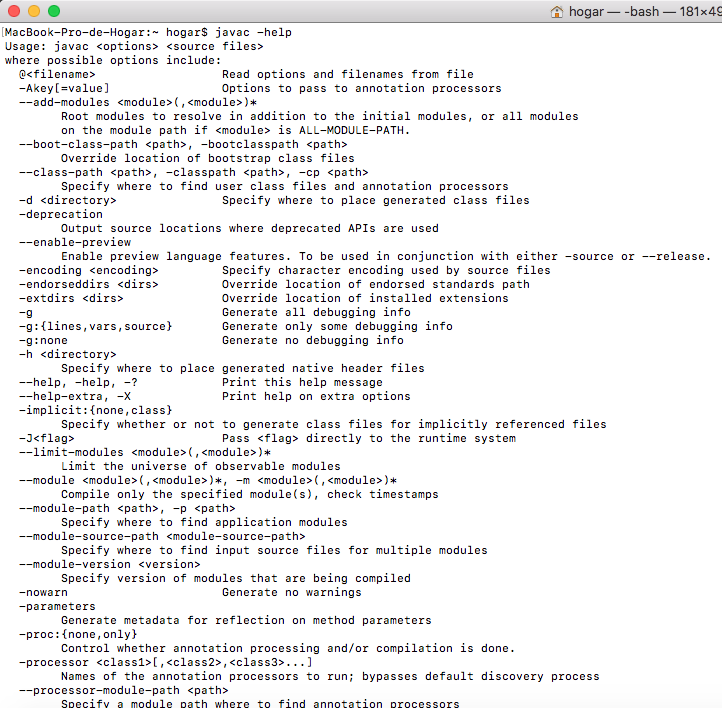
javadoc: genera la documentacion API en formato HTML.

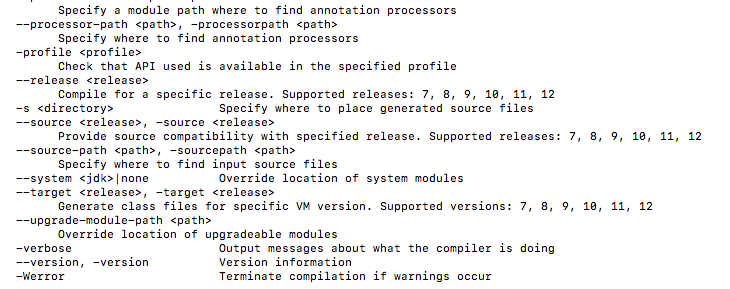
jar: sirve para crear ficheros .jar.

1. java -help



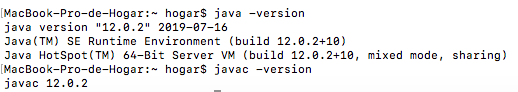


javac -help



1. java -version

javac -version

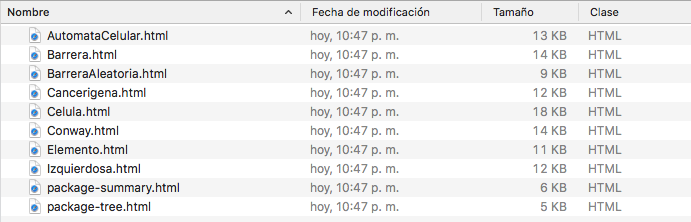


**Compilando**

1. javac src/aplicacion/\*.java -d bin
2. Aparecen los elemento \*.class en la carpeta aplicación la cual esta en bin.

**Documentando**

1. javadoc src/aplicacion/\*.java -d docs



**Ejecutando**

**Probando**



**Empaquetando**



**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)

22h

1. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)

Mayormente completo, ya que nos falto realizar desde ejecutando.

1. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importante?

Porque nos ayudarona verificar que con cada cambio que hicieramos, el programa seguira funcionando correctamente.

1. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?

Realizar la mayor parte del laboratorio, porque le hemos dedicado bastante tiempo esforzandonos para que quede bien.

Nuestro mayor problema lo encontramos cuando nos pedian que trabajaramos java desde consola. Para intentar resolverlo solucionabamos dudas con compañeros.

1. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

Implementar las nuevas tecnologias para trabajar a pares. Solucionar dudas con anticipacion.