**PROGRAMACIÓN ORIENTADA A OBJETOS**

**DESARROLLO**

**Conociendo**

1. ¿Cuántos paquetes tiene?

El archive tiene dos paquetes, los cuales son aplicación y presentación.

1. ¿Cuántas clases tiene en total? ¿Cuántas tienen fuente?

El paquete aplicación las clases AutomataCelular y Celula, el paquete presentación tiene una clase la cual es AutomataGUI y esta tiene dos clases de más adentro de ella, es decir en total hay 4 clases.

Todas tienen fuente, solo que en la clase autómata celular hacen falta implementar los métodos.

1. ¿Cuál es la clase ejecutiva?

AutomataGUI, ya que es la que tiene todas las funcionalidades.

1. ¿Que funcionalidades ofrece?

La clase célula nos permite crear una célula y así mismo mirar su edad y estado, como también cambiar este mismo.

La clase AutomataGUI nos muestra la interfaz gráfica.

La clase AutomataCelular nos genera una matriz de células

**Arquitectura general**

1. Consulte el significado de las palabras package e import:

Las cláusulas package e import están fuertemente relacionadas entre si y son usualmente las primeras lineas presentes en los archivos de Código fuente Java.

**Package:** Los paquetes son el mecanismo que usa Java para facilitar la modularidad del Código.

**Import:** Con la cláusula import cualquier módulo de código puede hacer referencia a tipos definidos en otros paquetes.

En el Proyecto es utilizado la cláusula import para poder importar paquetes los cuales tienen código que Podemos utilizar sin necesidad de sobrescribirlo.

Y utilizamos la palabra package para referenciar que esa clase pertenece a ese paquete

1. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido.

En el directorio autómata tenemos unos subdirectorios los cuales son aplicación, doc y presentación. Los paquetes son directorios los cuales almacenan archivos.

1. Archivo autómata.astah

**Arquitectura detallada**

1. Archivo automata.astah
2. ¿Es qué paquete debe estar?

Para nosotros las pruebas deberían estar en el paquete de aplicación asociado a la clase Célula ya que es la clase que crea las células, maneja su estado y nos permite saber si esta viva.

Archivo automata.bluej

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

1. Para almacenar los objetos de tipo Elemento utiliza una matriz llamada autómata. Si porque célula implementa la interfaz Elemento, por lo tanto, es un tipo compatible.
2. La define la clase AutomataCelular, Color y utiliza métodos de la interfaz Elemento
3. ¿Qué saben hacer?

Saben retornar en que fila-columna están ubicadas, pueden retornar su color.

Las células pueden retornar su estado actual, es decir si están vivas o por el contrario están muertas, también nos pueden retornar su edad y cambiar su estado.

¿Qué no puede hacer distinto?

Que, si la edad de la célula es mayor a 2, no pueden cambiar su estado a VIVA, siempre estarán en un estado MUERTA

¿Qué debe aprender hacer?

Saber cuál será su estado siguiente dependiendo de los vecinos que tengan.

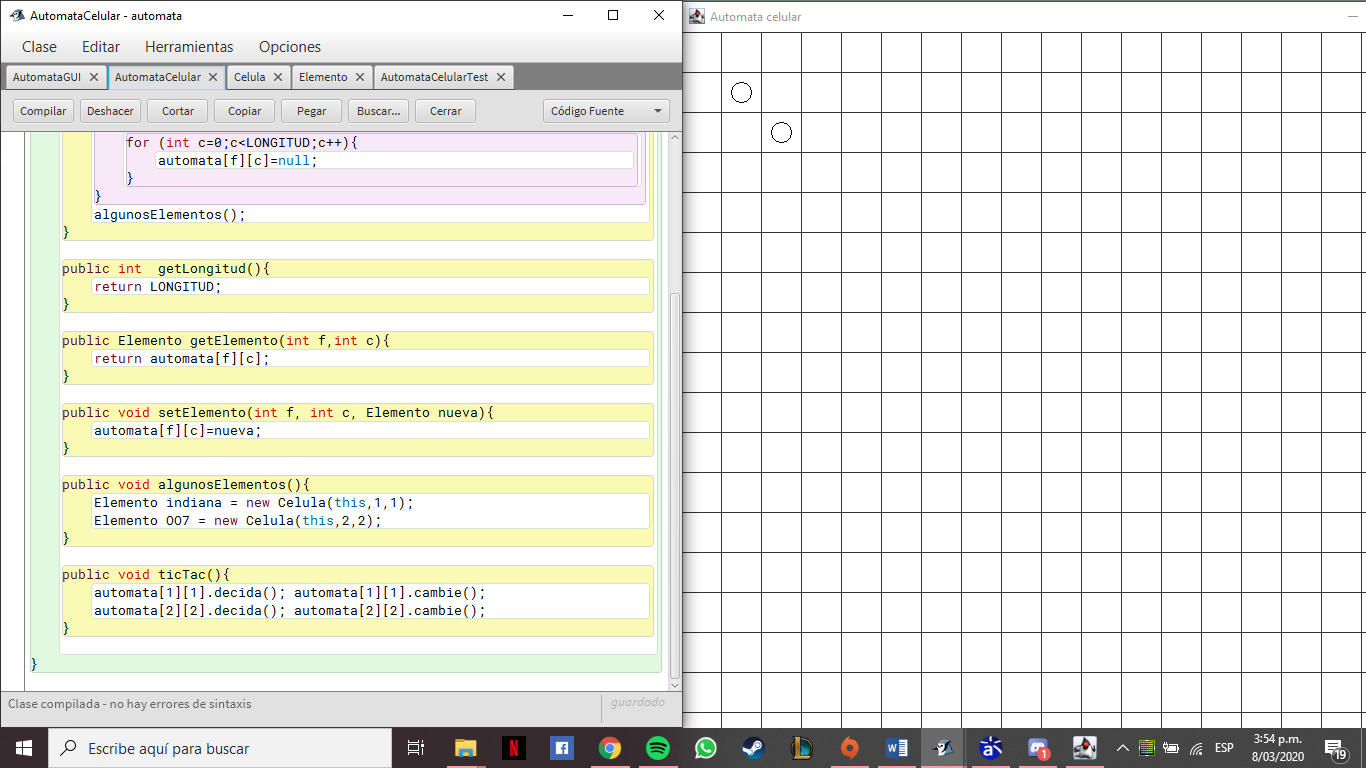
1. Por comportarse como un elemento, ¿qué sabe hacer? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer?

* Retornar su forma y si está vivo
* En el caso que no se implementó el método isVivo, no podrá cambiar el retorno que es FALSE
* Debe aprender a cambiar de estado, a decidir cuál será su estado siguiente y a retornar cuál es su color

1. Considerando lo anterior, una Célula, ¿de qué color es? ¿Cómo decide? ¿Cómo cambia? Justifique sus respuestas.

* La célula según la clase Color, es de color negro
* Si la edad es igual-mayor a 2, el estado siguiente será MUERTA.
* Primero a la edad le sumamos uno, y el estado actual cambia al estado siguiente.

1. Ahora vamos a crear dos células en diferentes posiciones (1,1) (2,2) llámelos indiana y 007 usando el método algunosElementos(). Ejecuten el programa, ¿Cómo quedan todas las células? Capturen una pantalla significativa.



1. En este punto vamos a construir (diseño y código) el método que atiende el botón tic-tac: el método llamado ticTac() de la clase AutomataCelular. ¿Cómo quedarían indiana y 007 después de uno, dos y tres tic-tac? Escriba la prueba correspondiente.

* Después de 1: deberían aparecer vivas las dos

Su estado siguiente seria viva

* Después de 2: debería aparecer vivas las dos

Su estado siguiente seria Muerta

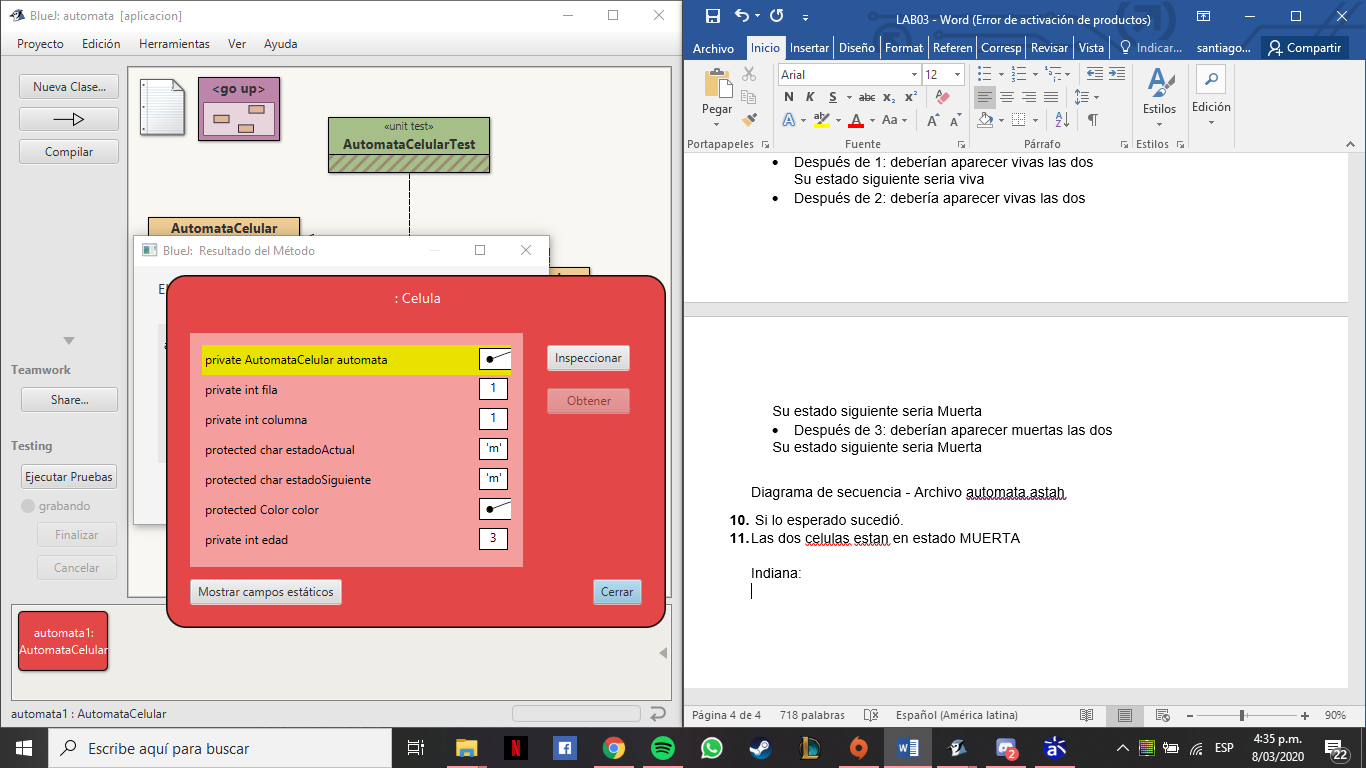
* Después de 3: deberían aparecer muertas las dos

Su estado siguiente seria Muerta

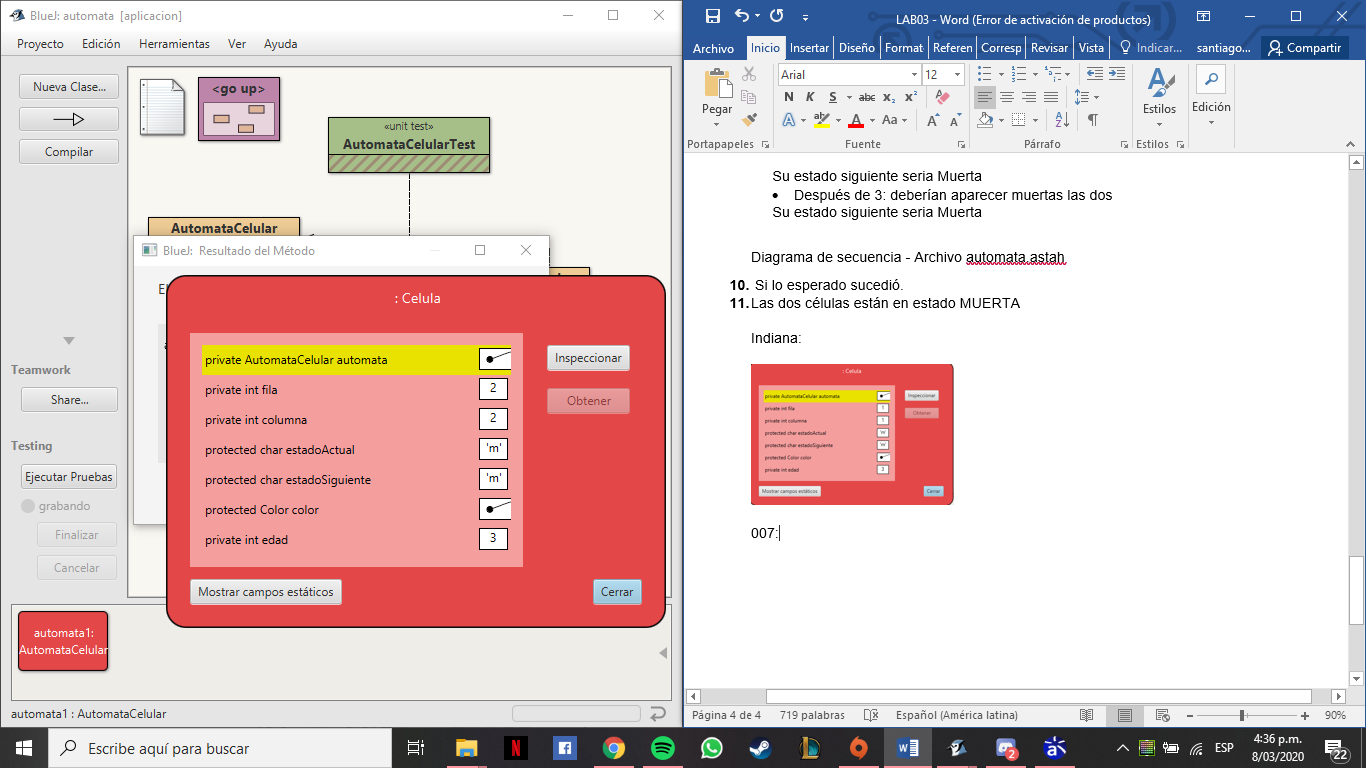
Diagrama de secuencia - Archivo automata.astah

1. Si lo esperado sucedió.
2. Las dos células están en estado MUERTA

Indiana:



007:



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

1. Si tenemos seguidas dos células izquierdosas vivas en la misma fila, ¿qué debería pasar en el primer, segundo y tercer clic? ¿por qué? Escriba la prueba correspondiente

En el primer clic la que está a la parte izquierda tendrá un estado MUERTA ya que tiene una célula viva a su derecha.

En el segundo clic la que está a la derecha permanecerá viva, pero su estado siguiente será MUERTA

EN el tercer clic la que está a la derecha tendrá un estado de MUERTA y sus estados siguientes serán MUERTA.

Archivo automata.bluej

1. ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿Cuáles métodos se sobre-escriben? Ahora escriba el código correspondiente a la célula Izquierdosa ¿Las pruebas son correctas?

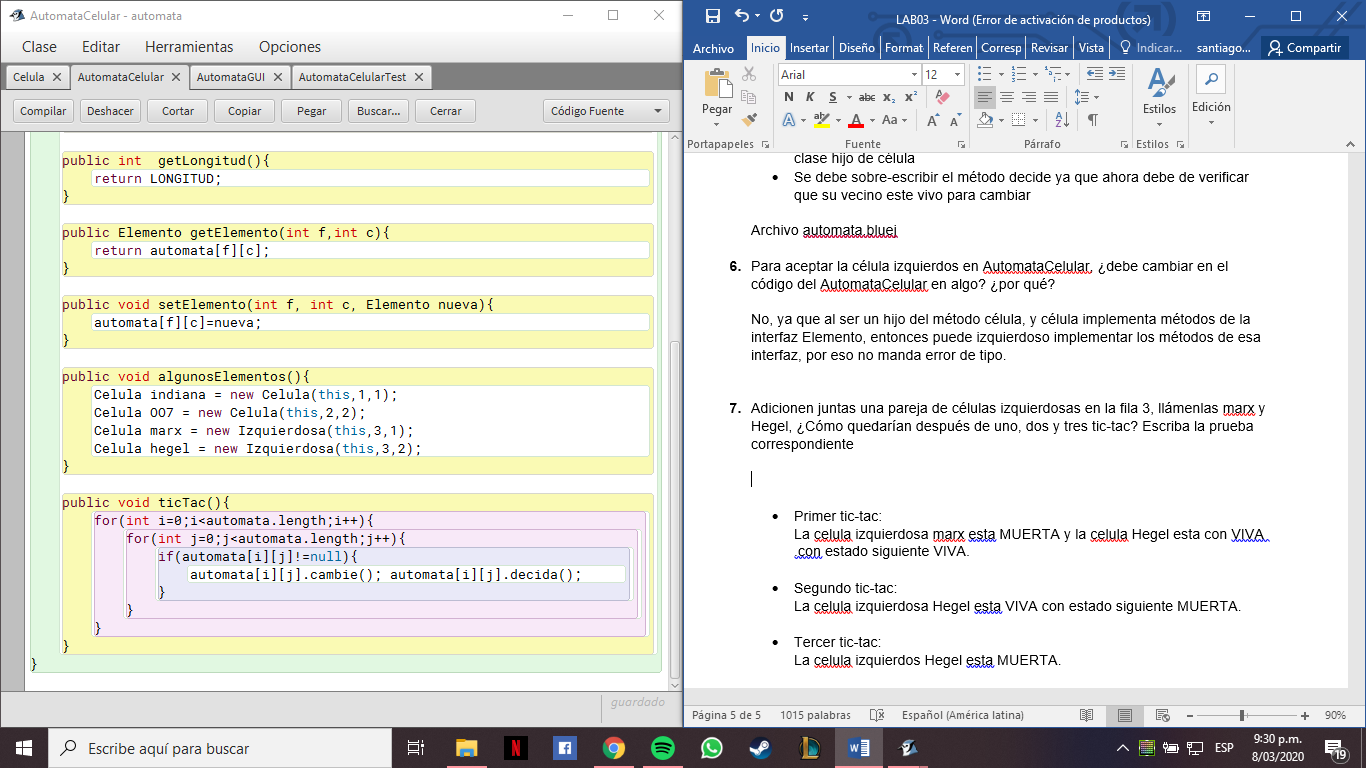
* Se debe añadir una nueva clase llamada izquierdosa
* Debemos de crear una herencia, en la cual izquierdosa será una nueva clase hijo de célula
* Se debe sobre-escribir el método decide ya que ahora debe de verificar que su vecino este vivo para cambiar

Archivo automata.bluej

1. Para aceptar la célula izquierdos en AutomataCelular, ¿debe cambiar en el código del AutomataCelular en algo? ¿por qué?

No, ya que al ser un hijo del método célula, y célula implementa métodos de la interfaz Elemento, entonces puede izquierdoso implementar los métodos de esa interfaz, por eso no manda error de tipo.

1. Adicionen juntas una pareja de células izquierdosas en la fila 3, llámenlas marx y Hegel, ¿Cómo quedarían después de uno, dos y tres tic-tac? Escriba la prueba correspondiente



* Primer tic-tac:

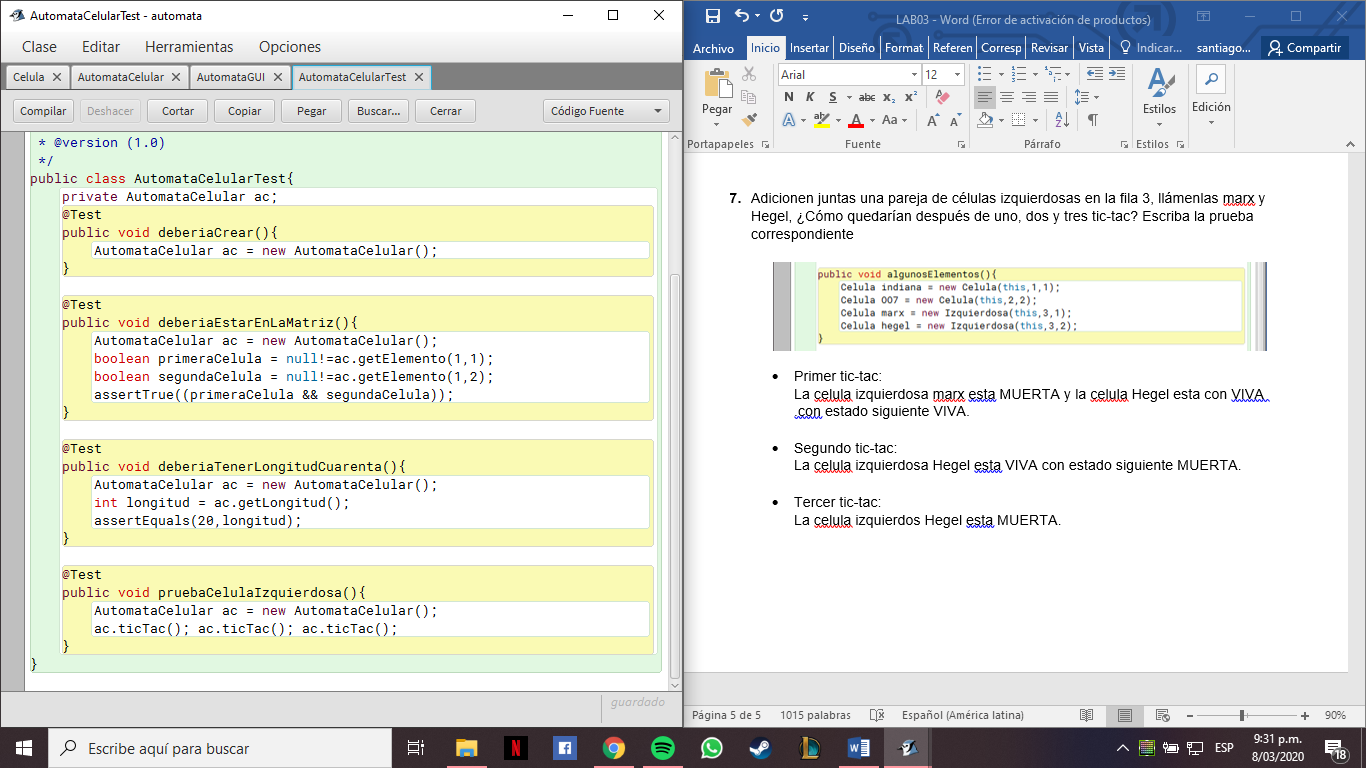
La celula izquierdosa marx esta MUERTA y la celula Hegel esta con VIVA .con estado siguiente VIVA.

* Segundo tic-tac:

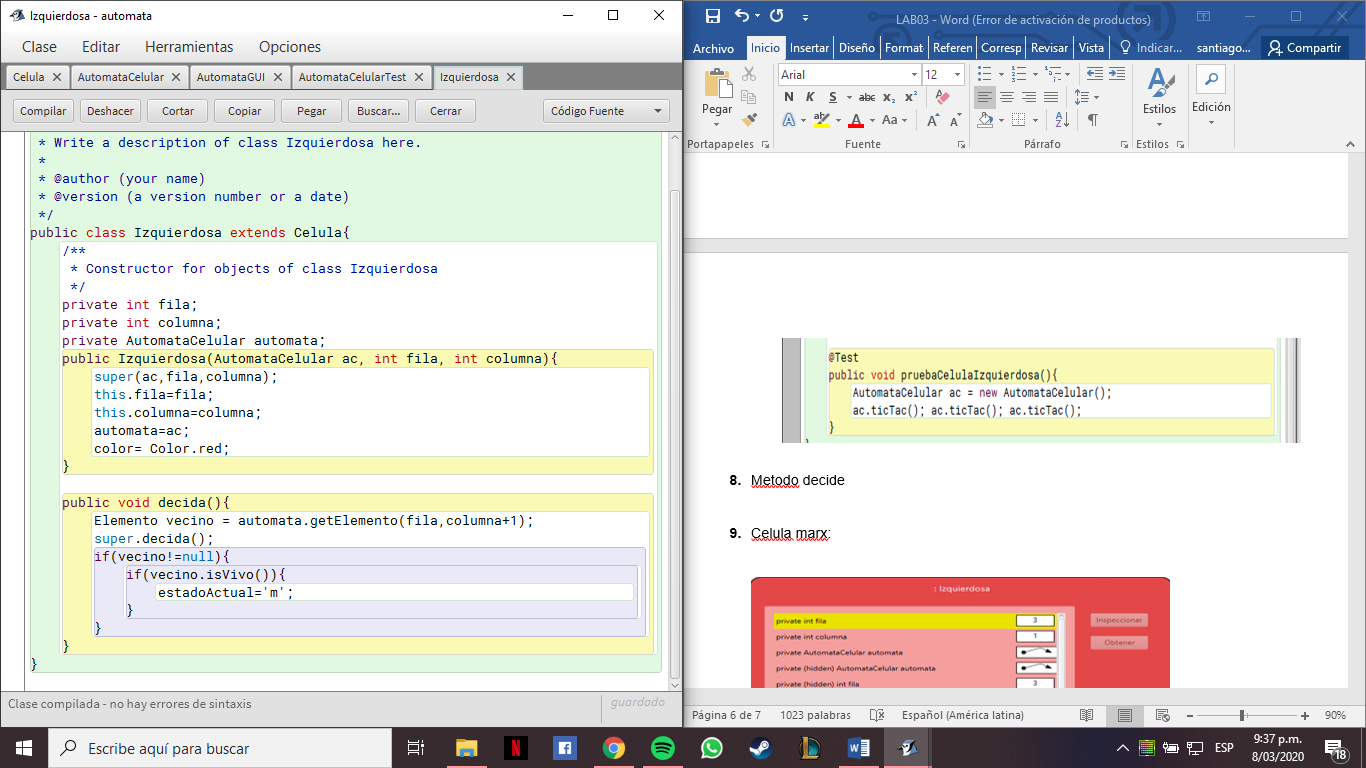
La celula izquierdosa Hegel esta VIVA con estado siguiente MUERTA.

* Tercer tic-tac:

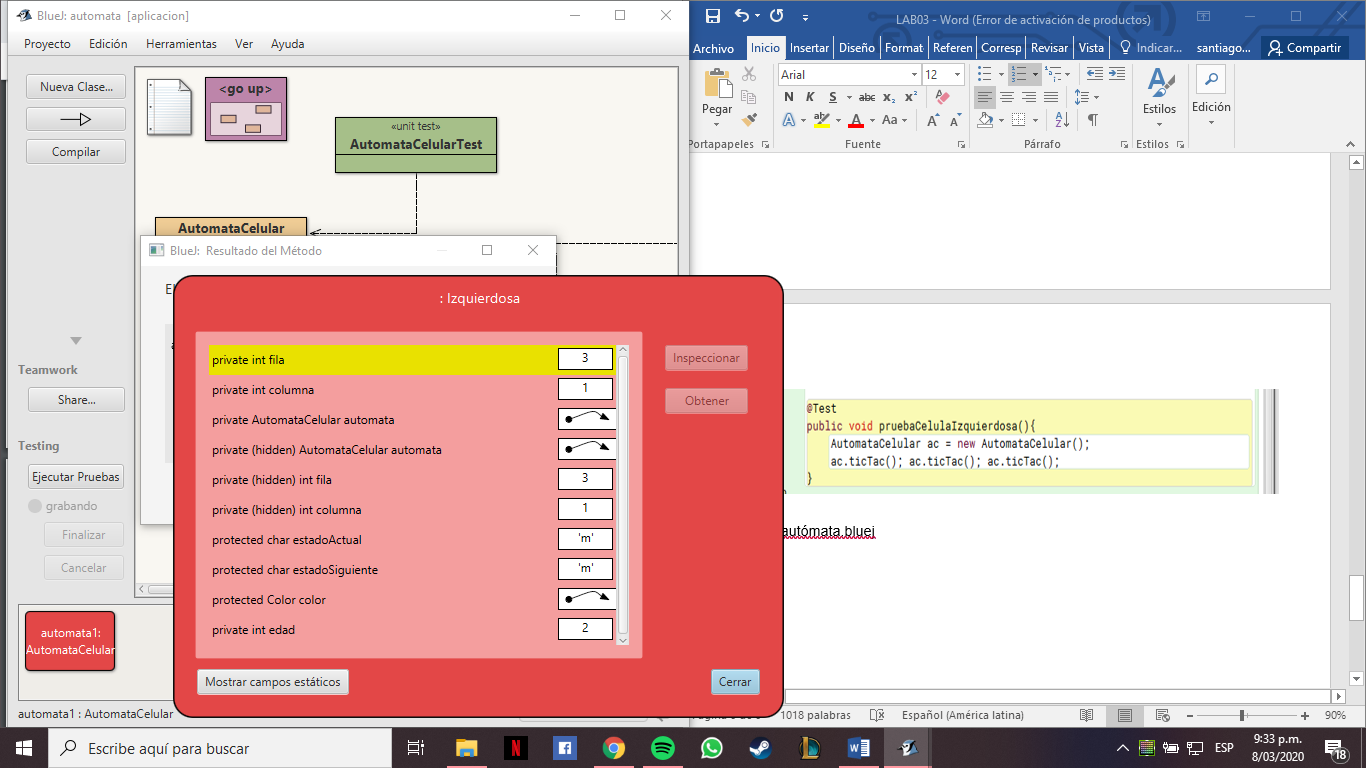
La celula izquierdos Hegel esta MUERTA.



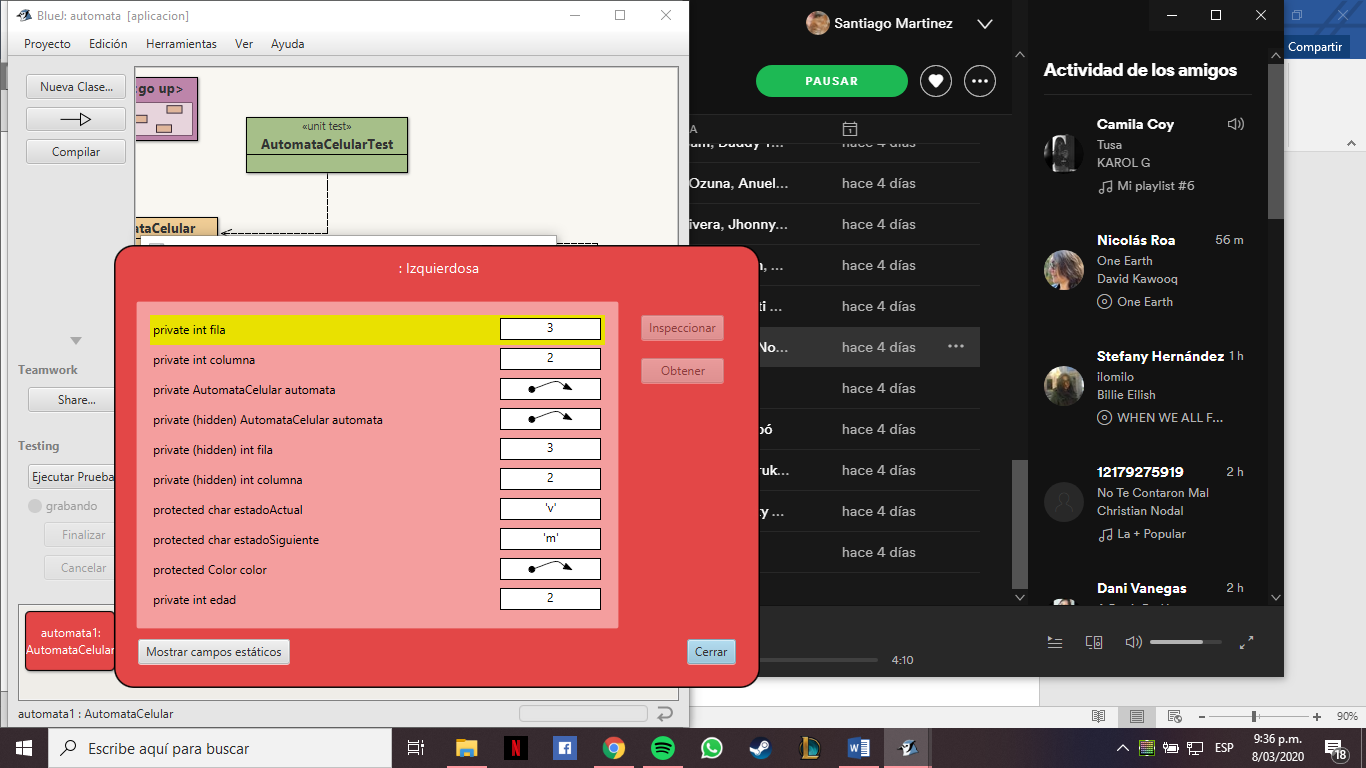
1. Metodo decide



1. Celula marx:



Celula Hegel:



**Ciclo 3. Adicionando una barrera**

1. Construya la clase Barrera para poder adicionarla en el AutomataCelular ¿qué hicieron?

Se creó la clase Barrera, la cual es un hijo de la Clase Célula, para esto se modificó el método getForma() para que retornara 2, esta es la forma cuadrada.

Para añadirlo en la clase AutomataCelular simplemente se creó una variable de tipo Barrera, como barrera es un hijo de Célula es posible que pueda ser añadida a la matriz de Elementos

1. Para aceptar este elemento, ¿debe cambiar en el codigo del AutomataCelular en algo? ¿por qué?

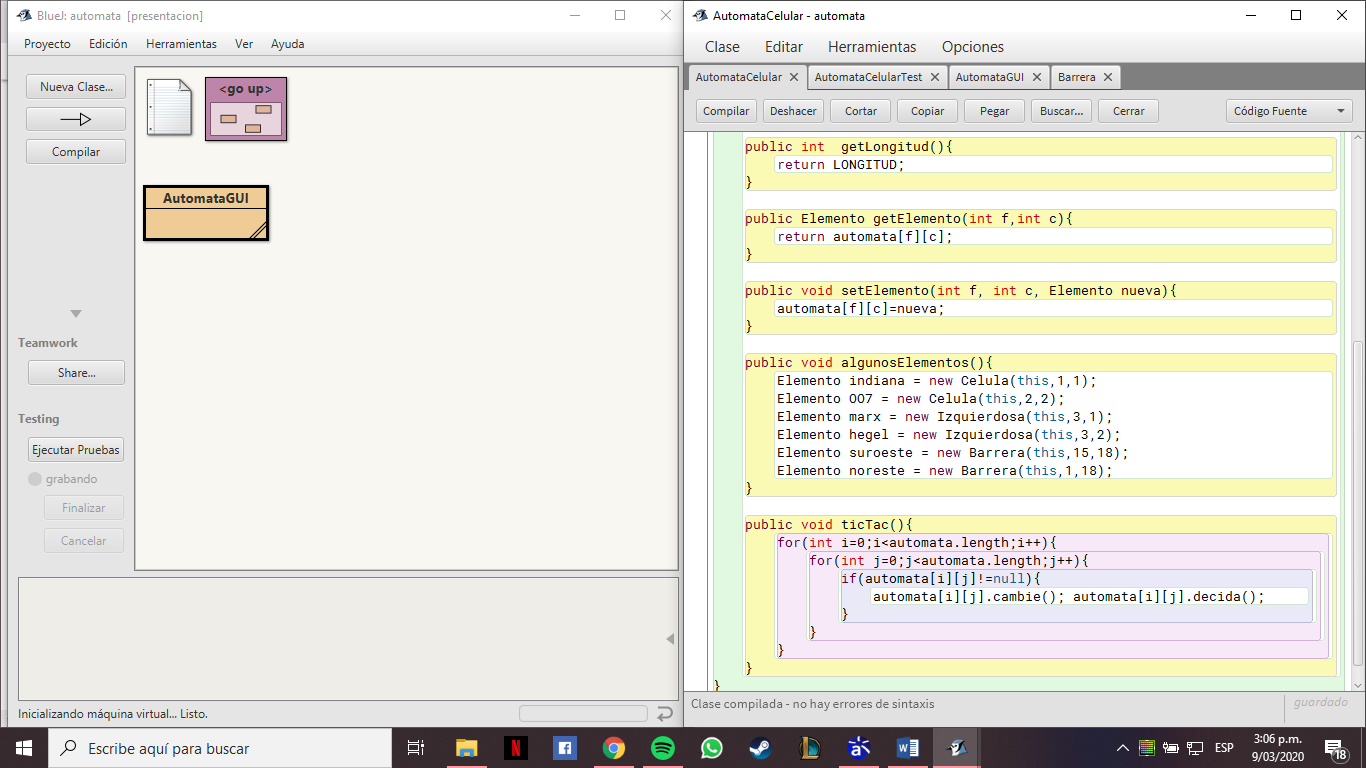
Simplemente se añade una línea de la creación de la barrera.



1. Adicionen dos Barreras cerca en las esquinas del AutomataCelular, llámenlas suroeste y noreste, ¿Cómo quedarían después de uno, dos y tres Tic-tac? Escriba las pruebas correspondientes.

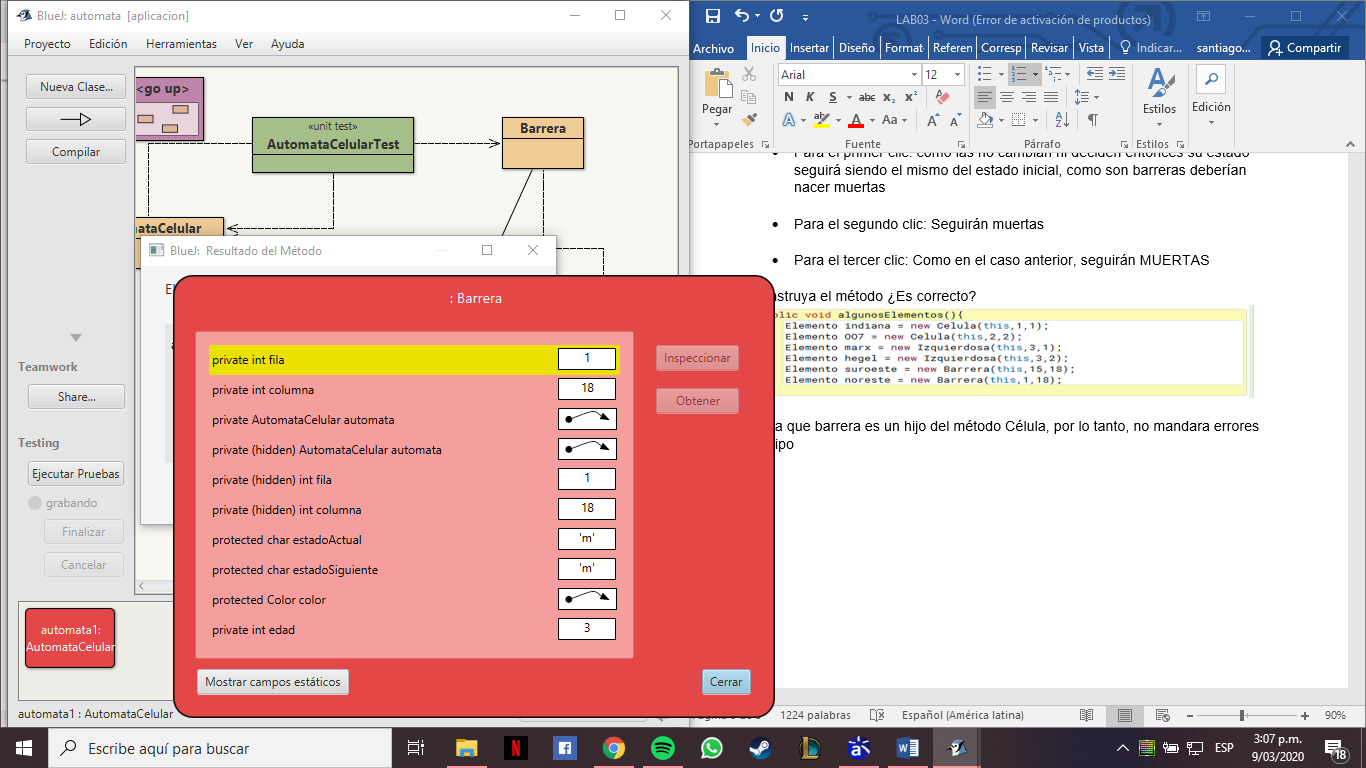
* Para el primer clic: como las no cambian ni deciden entonces su estado seguirá siendo el mismo del estado inicial, como son barreras deberían nacer muertas
* Para el segundo clic: Seguirán muertas
* Para el tercer clic: Como en el caso anterior, seguirán MUERTAS

1. Construya el método ¿Es correcto?

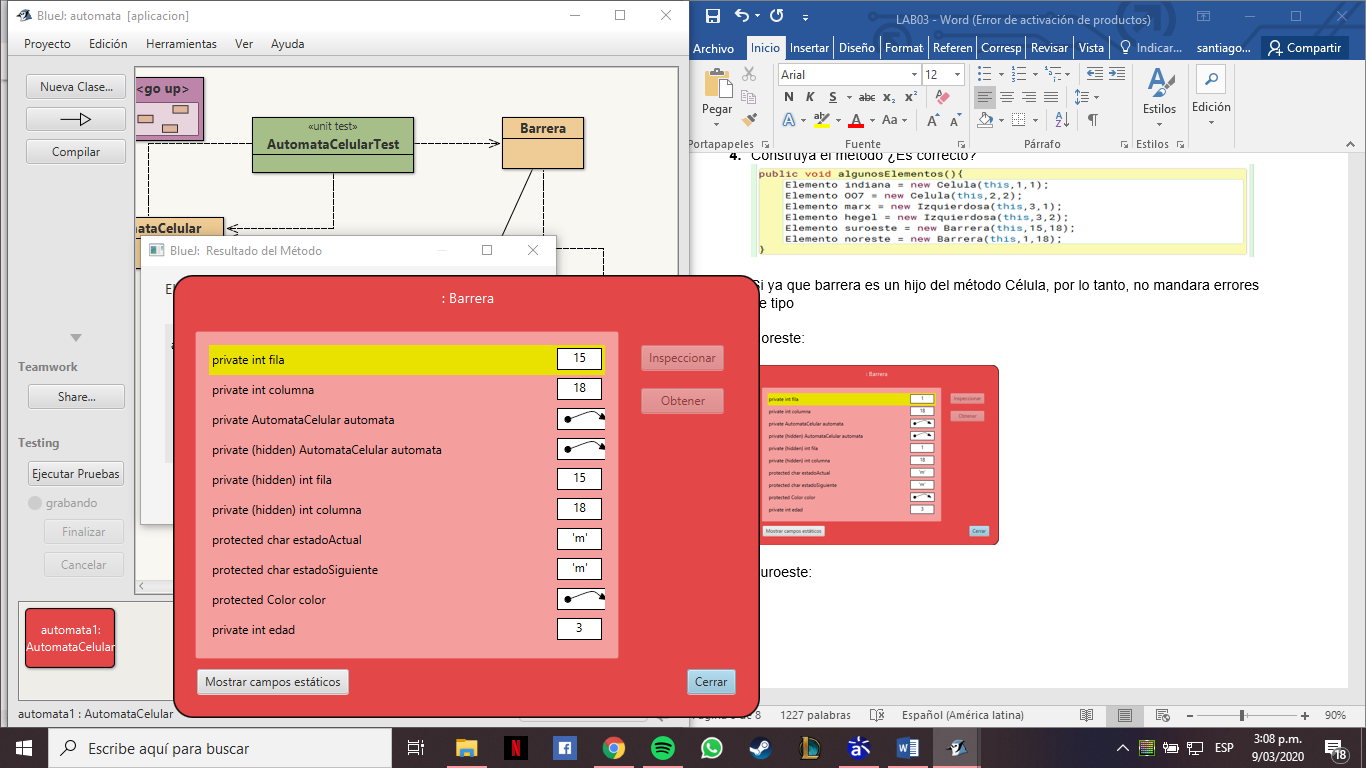


Si ya que barrera es un hijo del método Célula, por lo tanto, no mandara errores de tipo

1. Noreste:



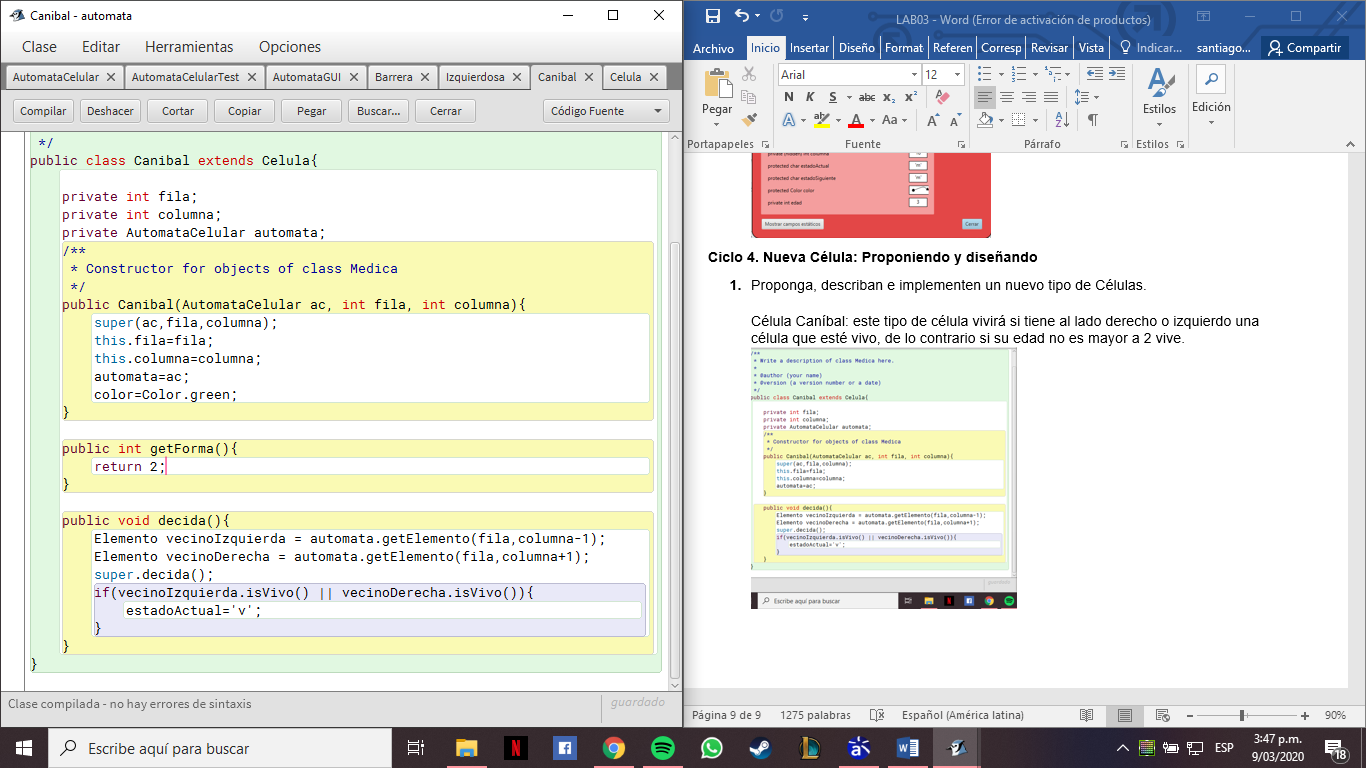
Suroeste:



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

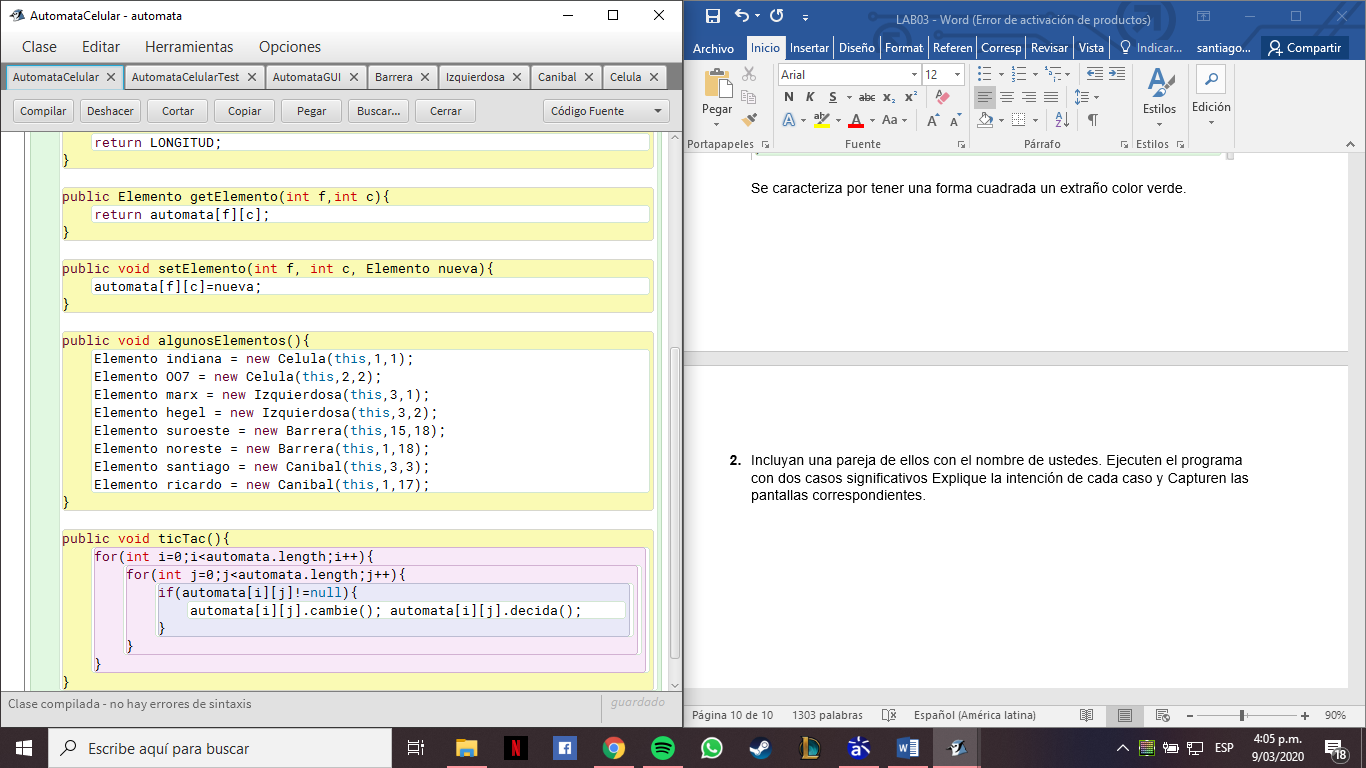
1. Proponga, describan e implementen un nuevo tipo de Células.

Célula Caníbal: este tipo de célula vivirá si tiene al lado derecho o izquierdo una célula que esté vivo.

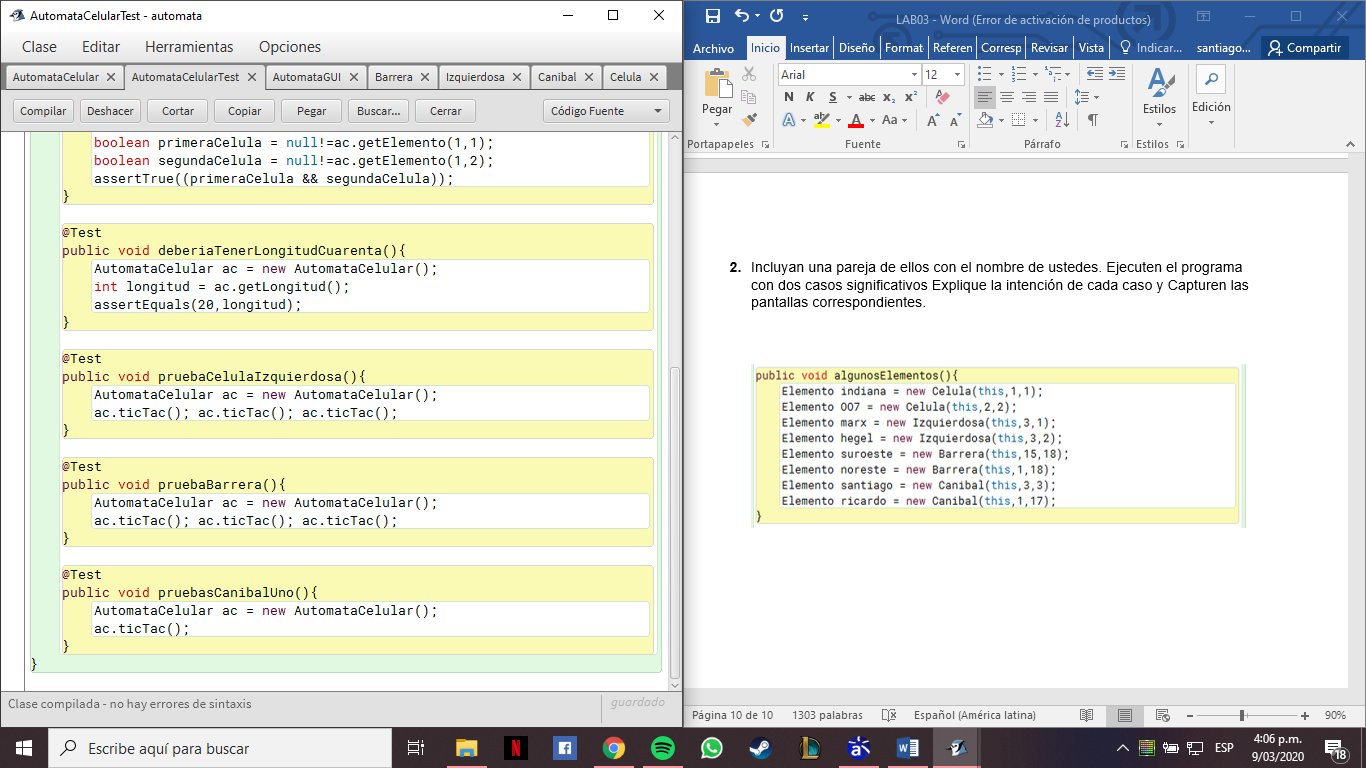


Se caracteriza por tener una forma cuadrada un extraño color verde.

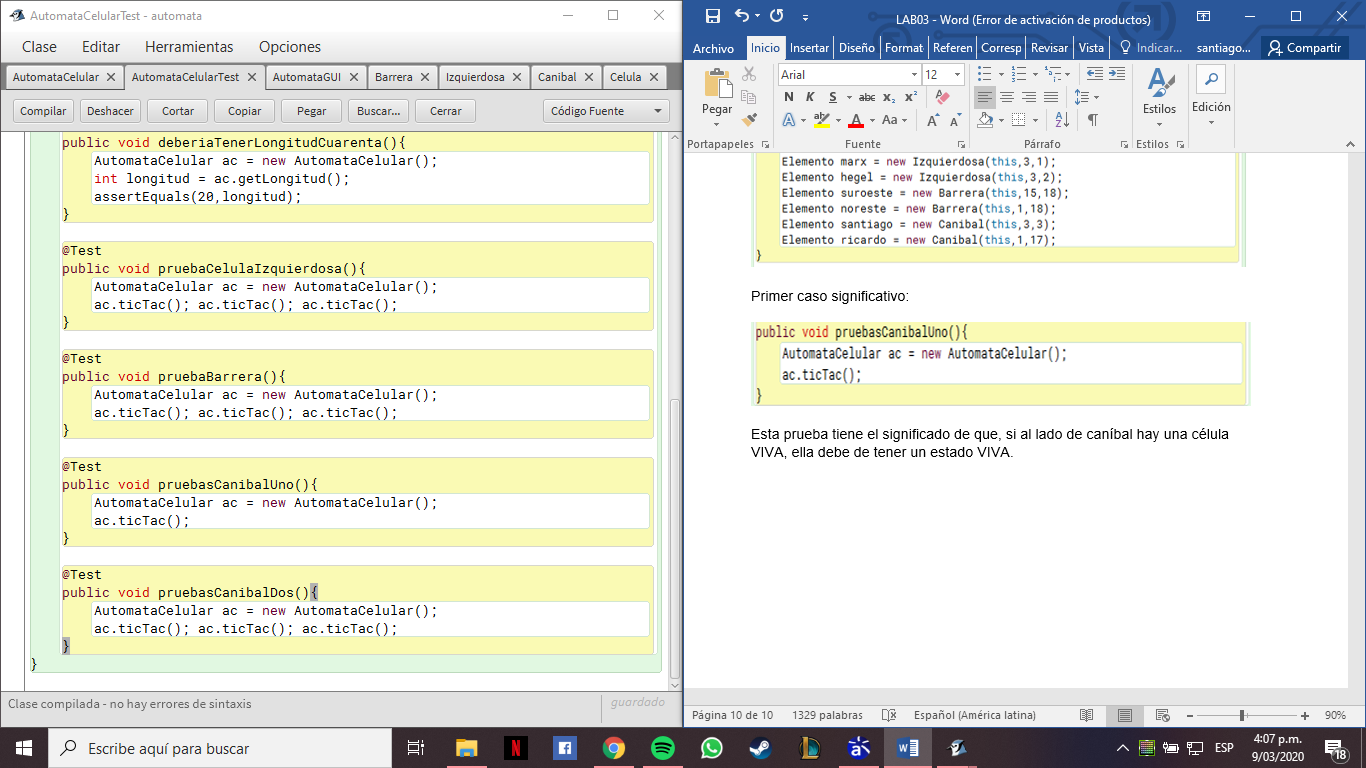
1. Incluyan una pareja de ellos con el nombre de ustedes. Ejecuten el programa con dos casos significativos Explique la intención de cada caso y Capturen las pantallas correspondientes.



Primer caso significativo:



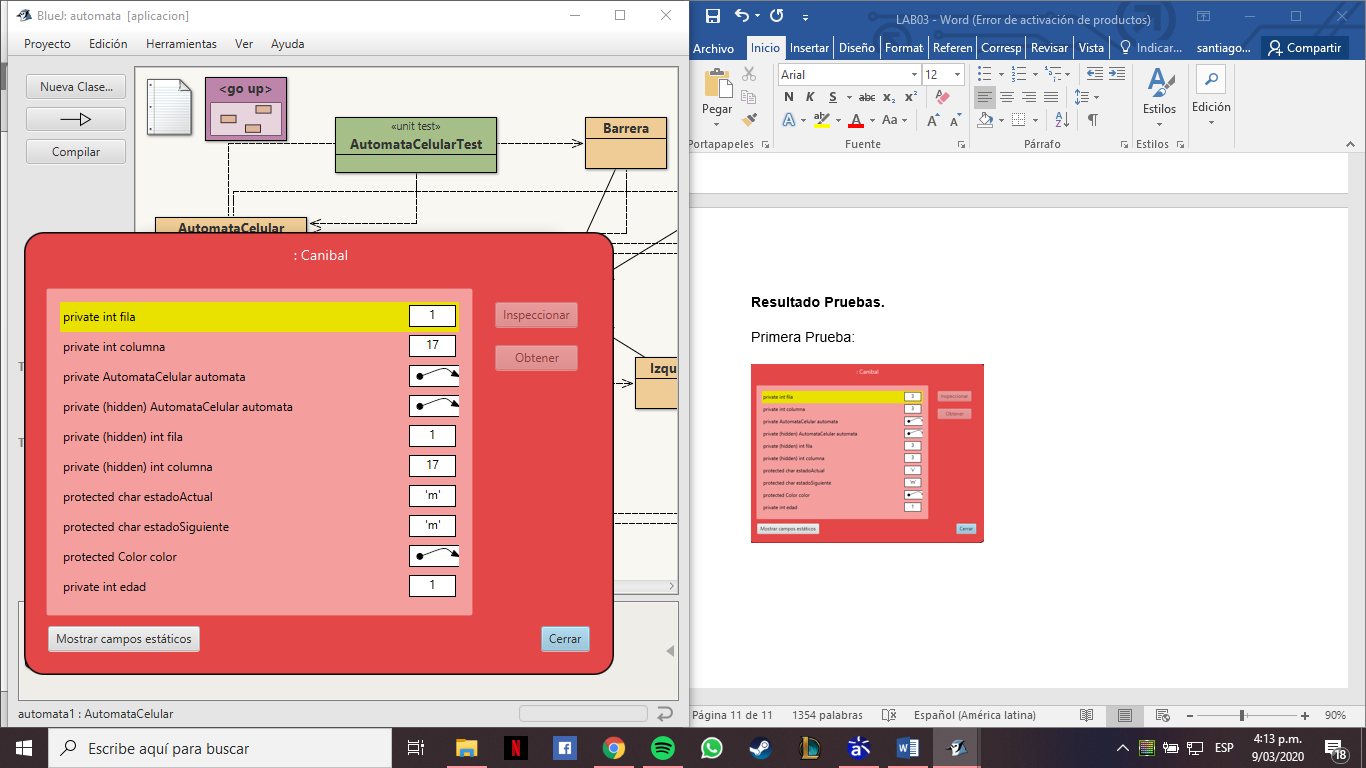
Esta prueba tiene el significado de que, si al lado de caníbal hay una célula VIVA, ella debe de tener un estado VIVA.

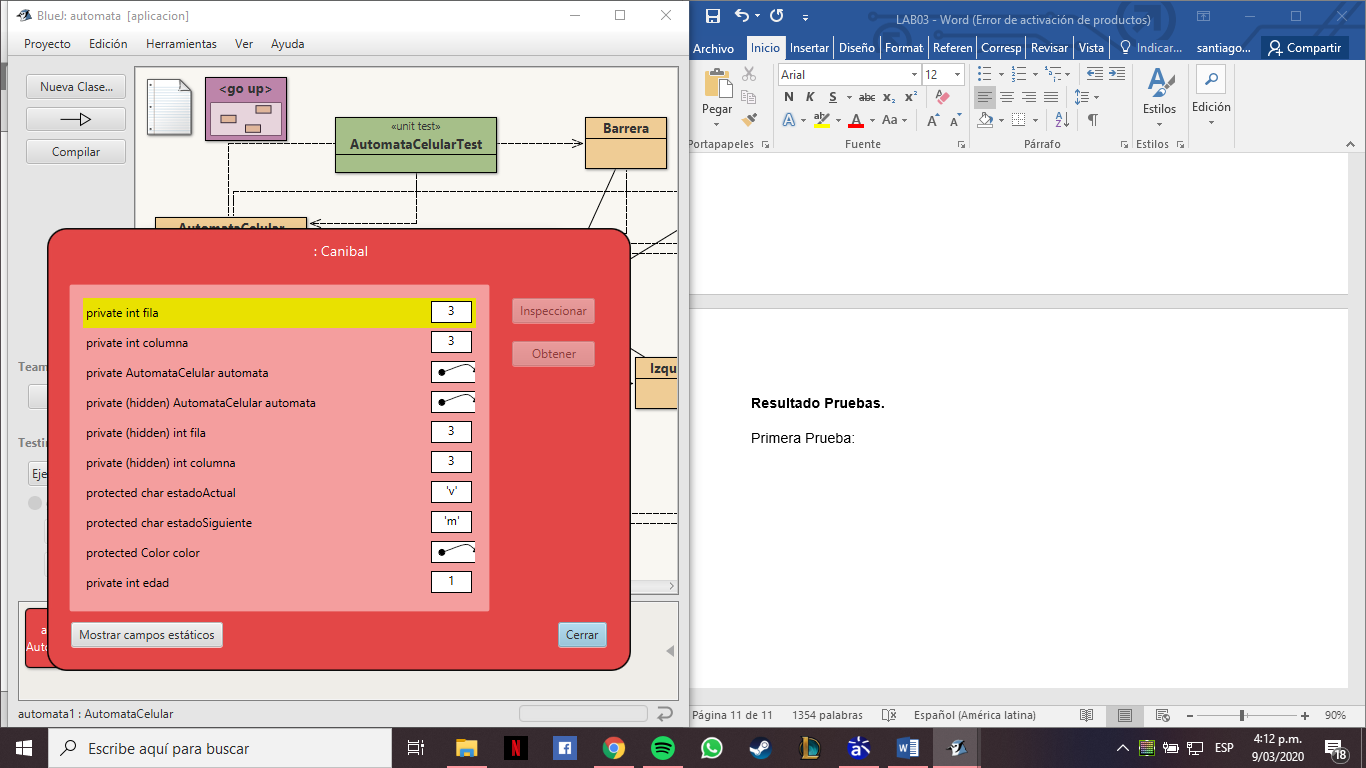


El objetivo de esta prueba es que al morir las células que estaban estado VIVA, las células caníbales deben de morir.

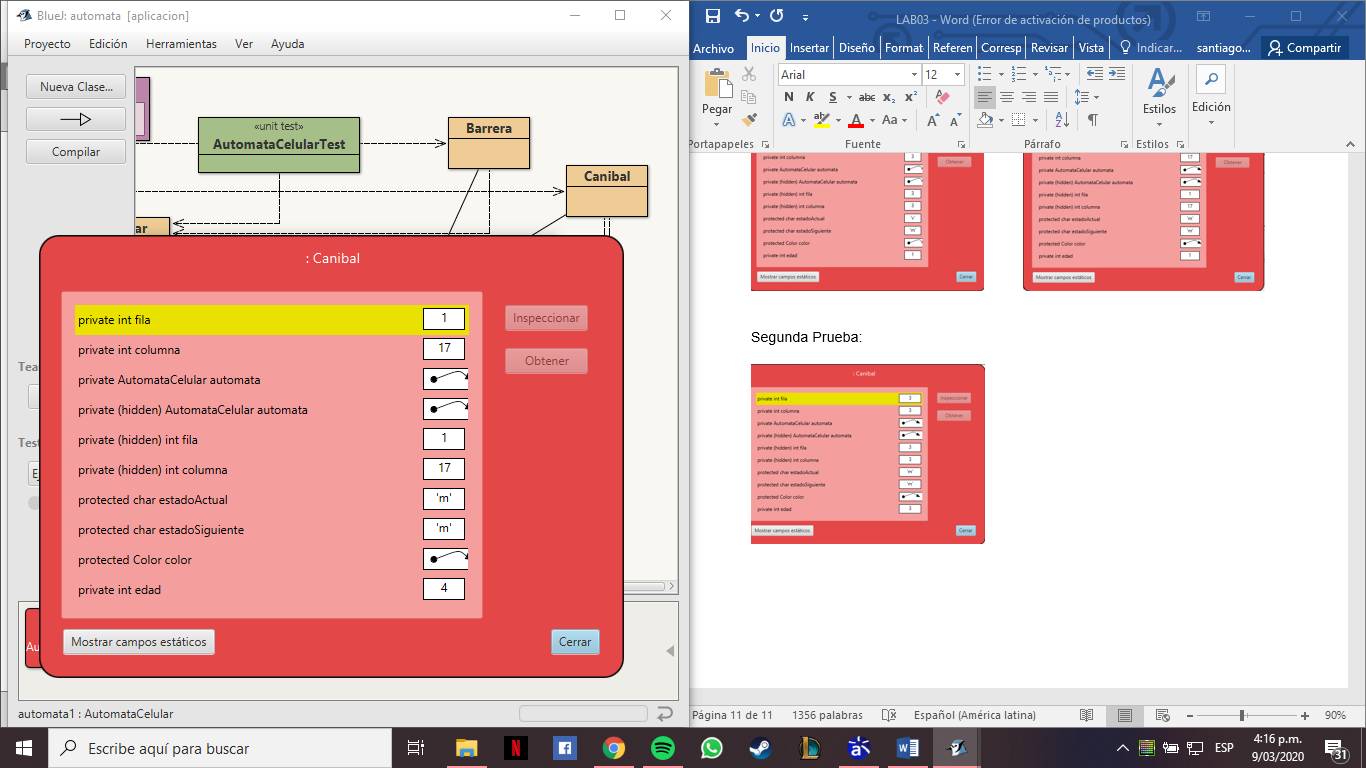
**Resultado Pruebas.**

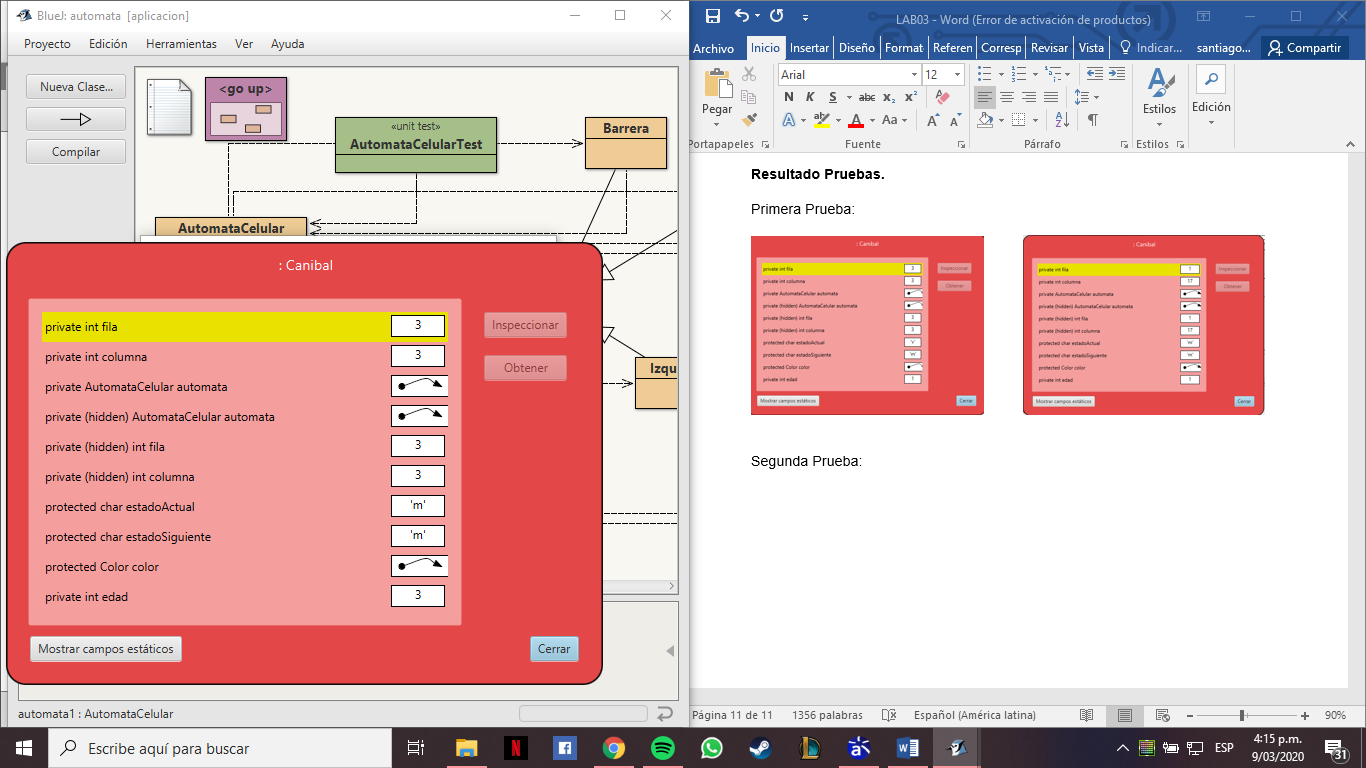
Primera Prueba:





Segunda Prueba:



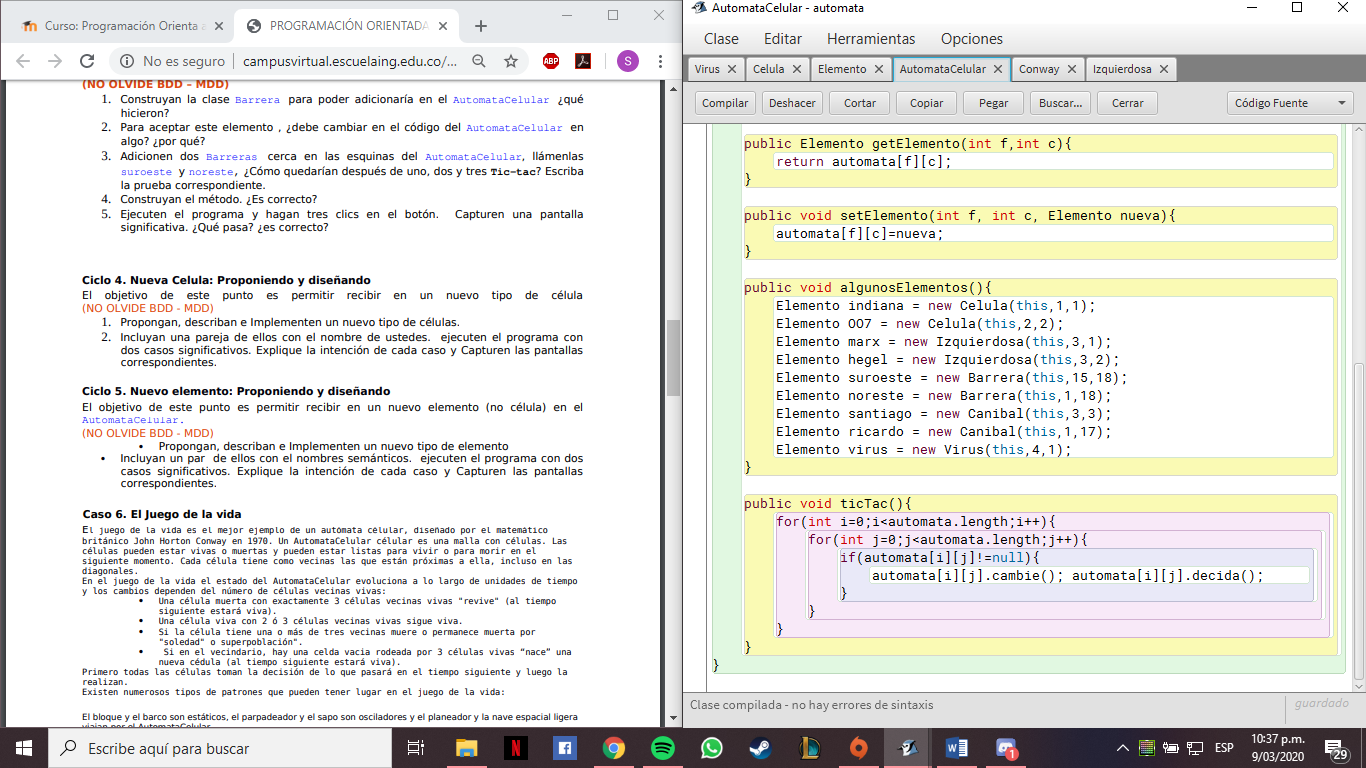


**Ciclo 5. Nuevo elemento: Proponiendo y diseñando**

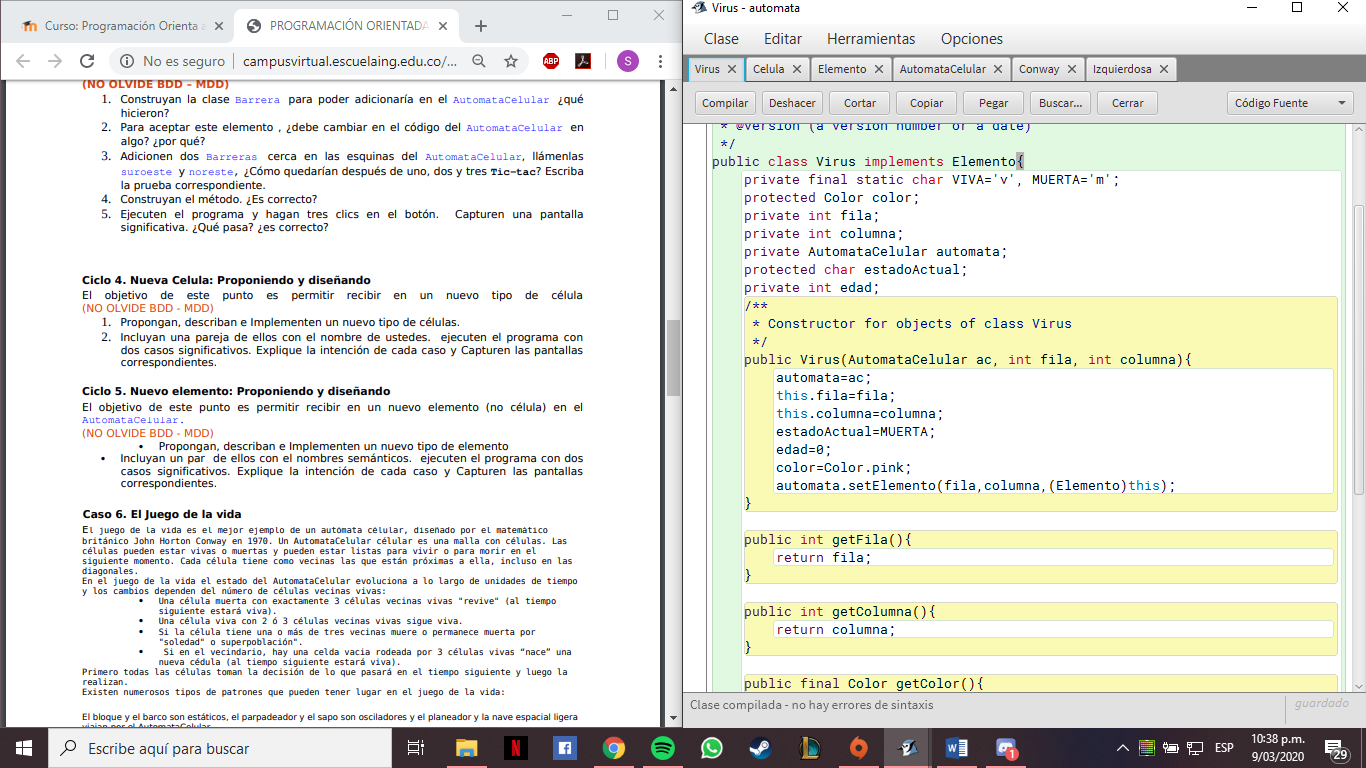
* Propongan, describan e implementen un nuevo tipo de elemento

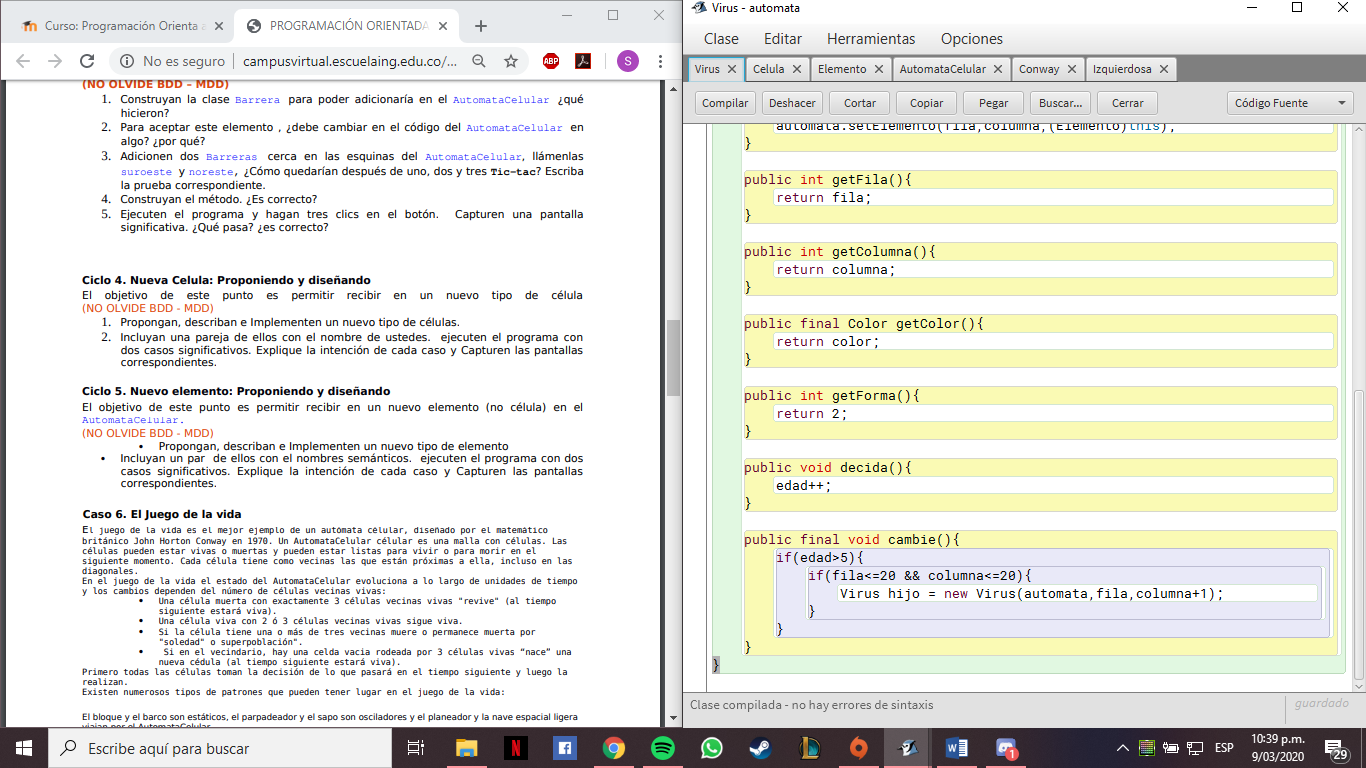
Proponemos un nuevo elemento llamado VIRUS, este elemento al tener una edad de 5 se reproduce y crea un nuevo VIRUS. Como es un virus siempre está muerto.

* Incluyan un par de ellos con el nombre semánticos. Ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes

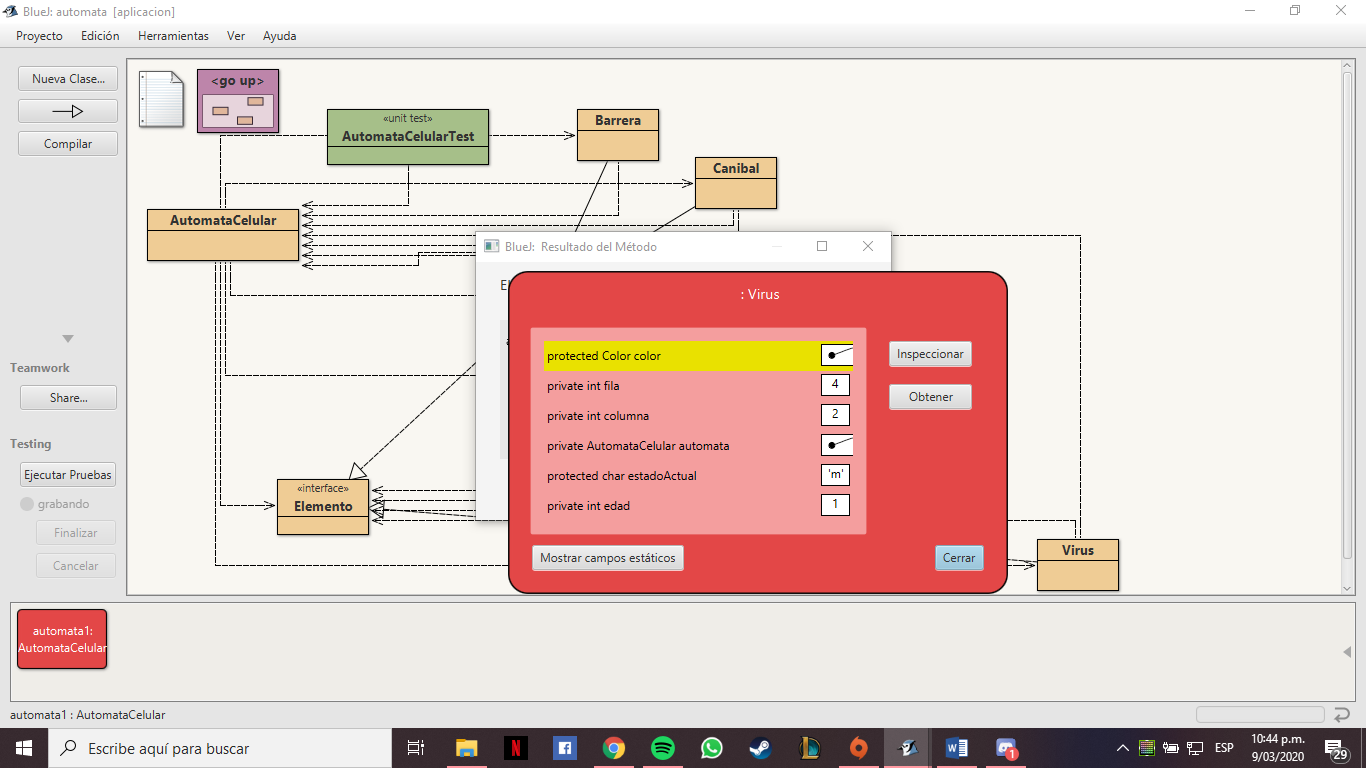
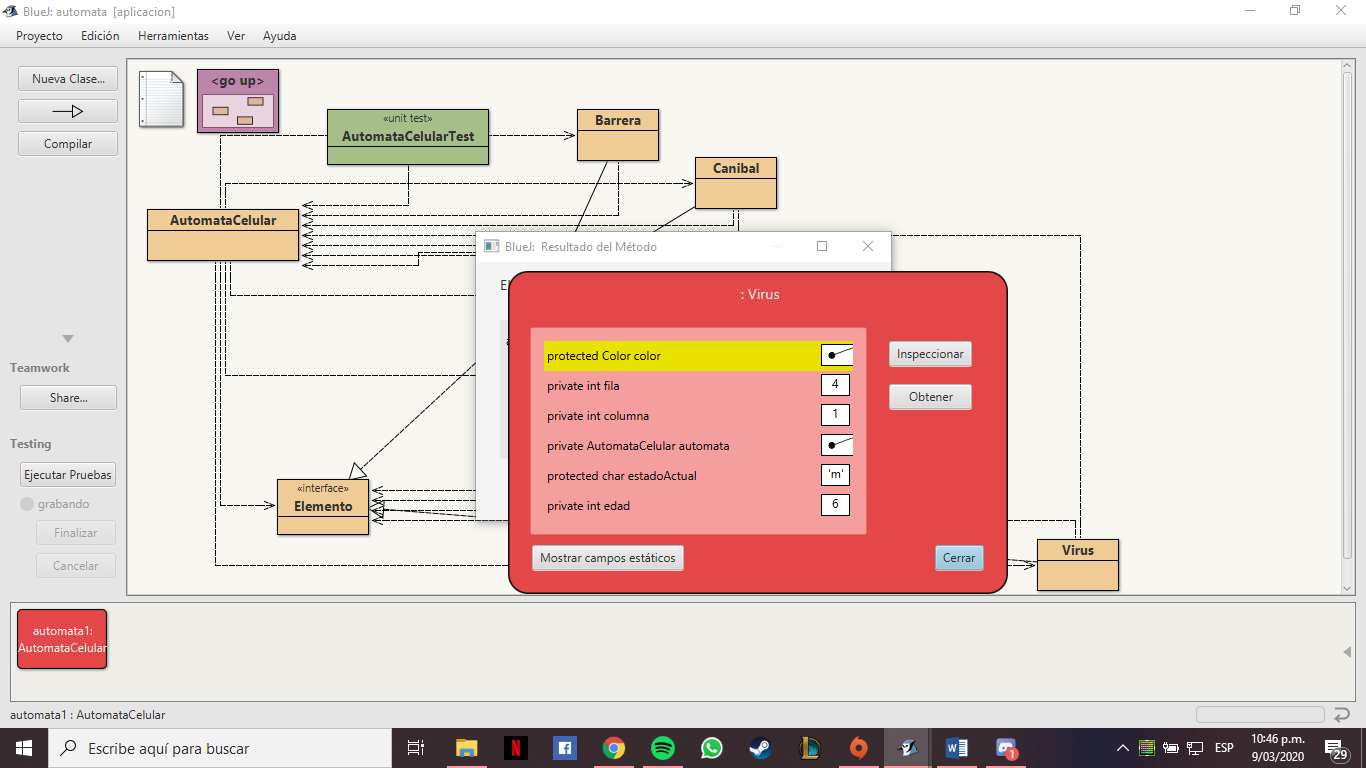
Creacion de virus:

Implementación de virus:

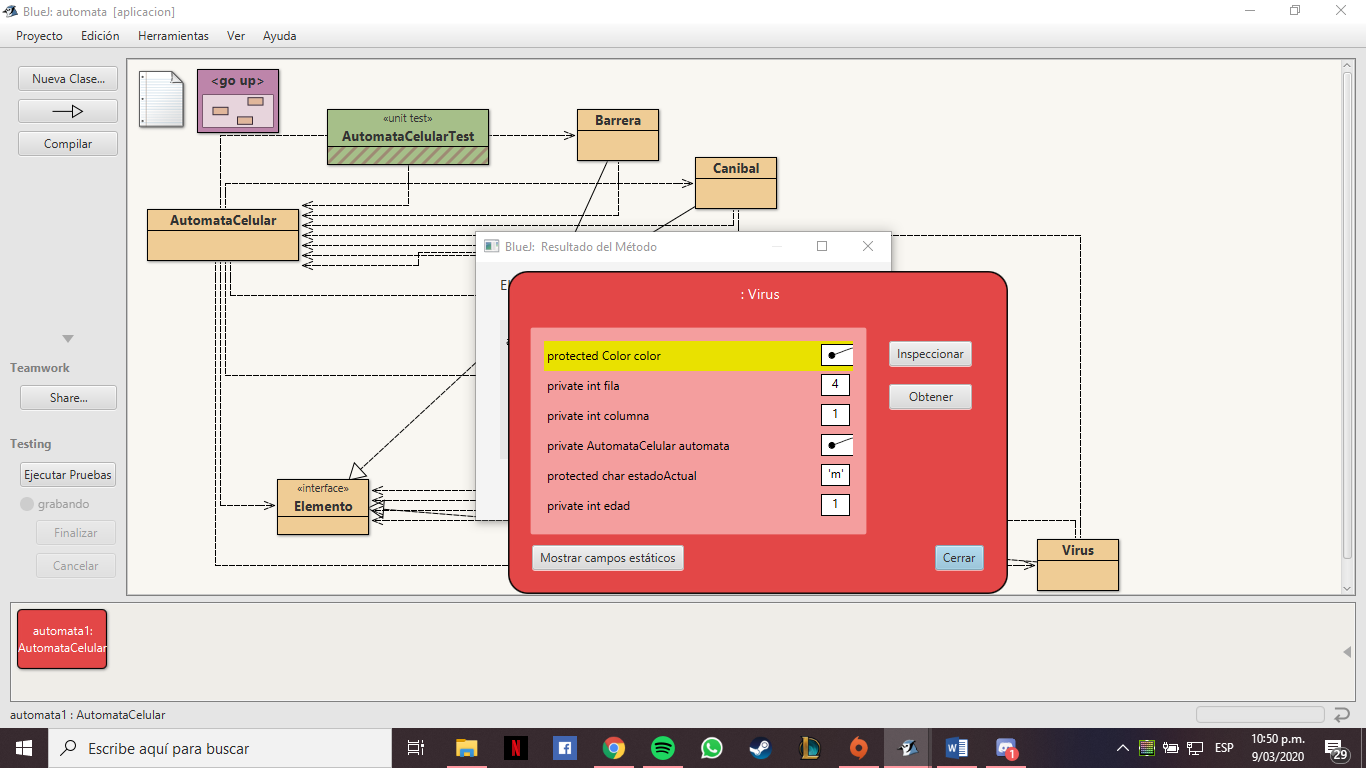




Resultado Prueba deberiaReproducirse():



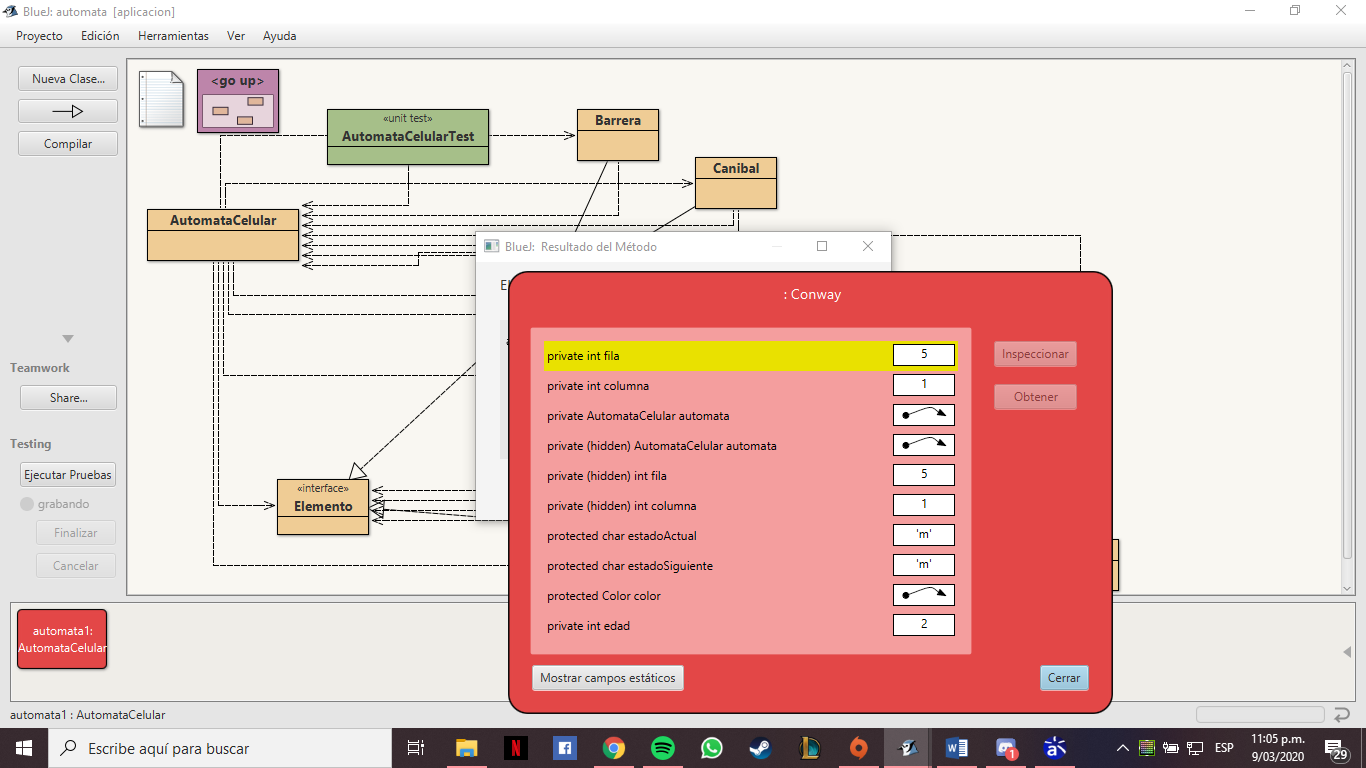
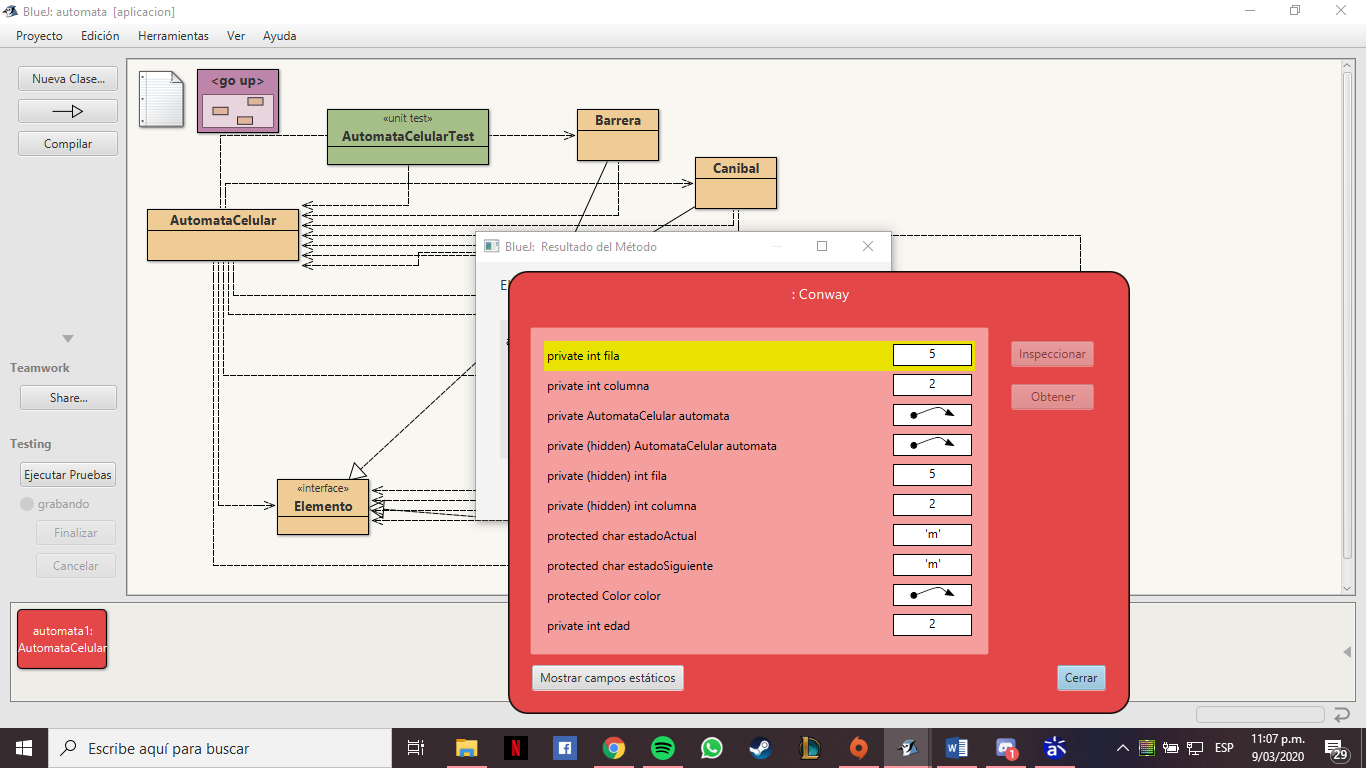
Resultado Prueba debeCrearVirus():



**Ciclo 6. El juego de la vida**

1. Muere por soledad ya que solo tiene una vecina viva, por lo tanto las dos celulas al estado siguiente mueren.
2. Se debe añadir un nuevo hijo de la clase Celula, el cual se llamara conway, por lo tanto se debe de añadir los metodos necesarios para que esta clase nueva sirva.

Las pruebas son correctas:



1. Es correcto, las dos células Células están MUERTAS

