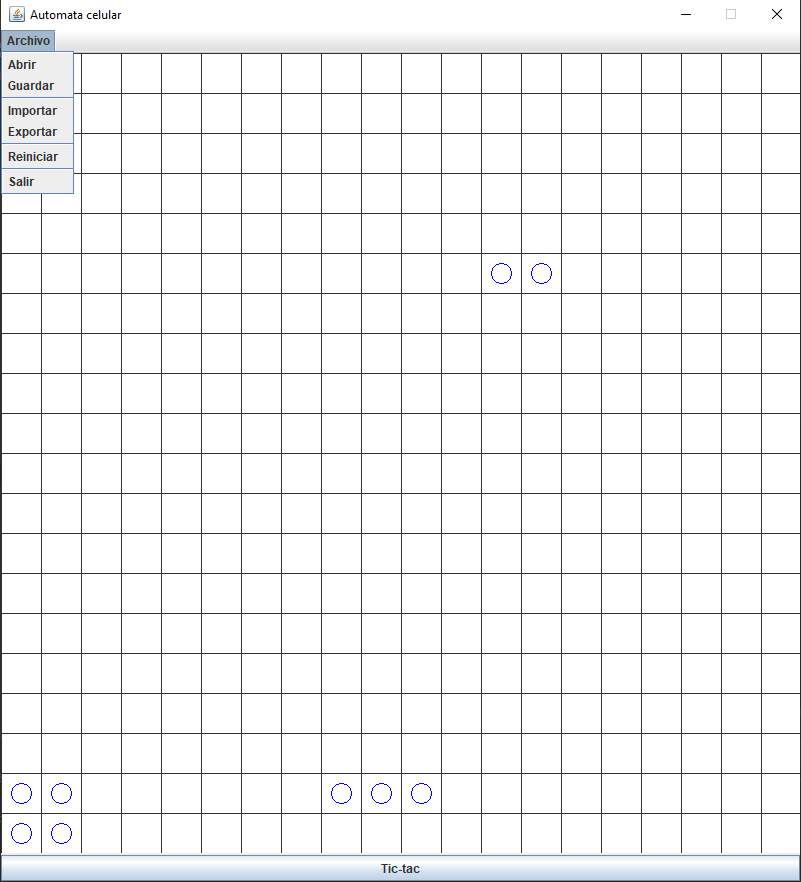
# PROGRAMACIÓN ORIENTADA A OBJETOS

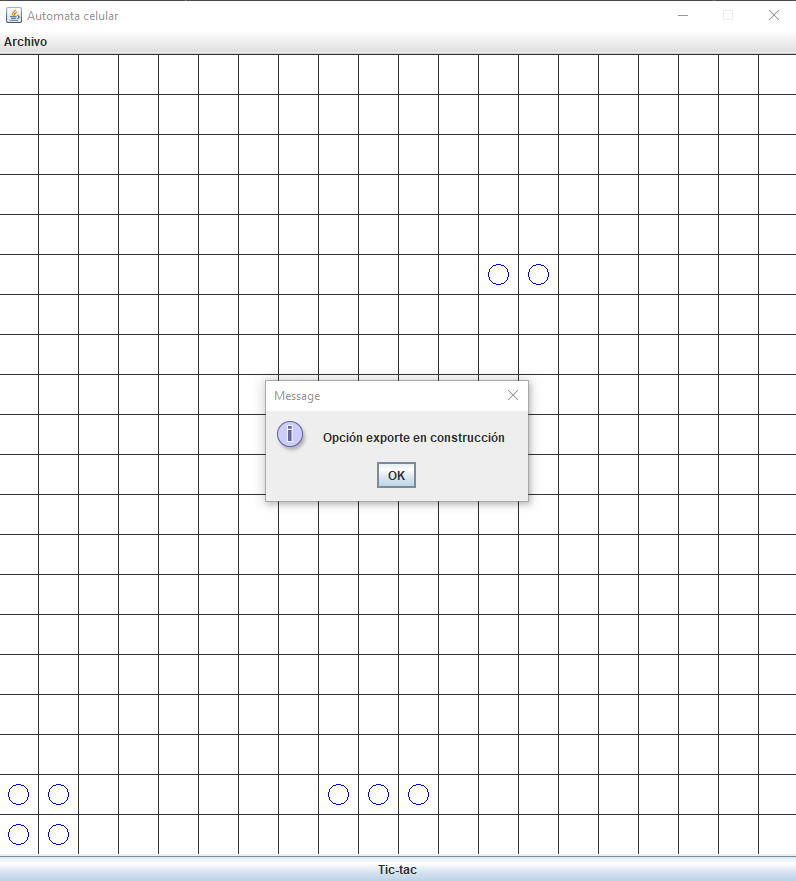
**I/O 2020-01**

**Laboratorio 6/6 [ :) ]**

### Creando la maqueta

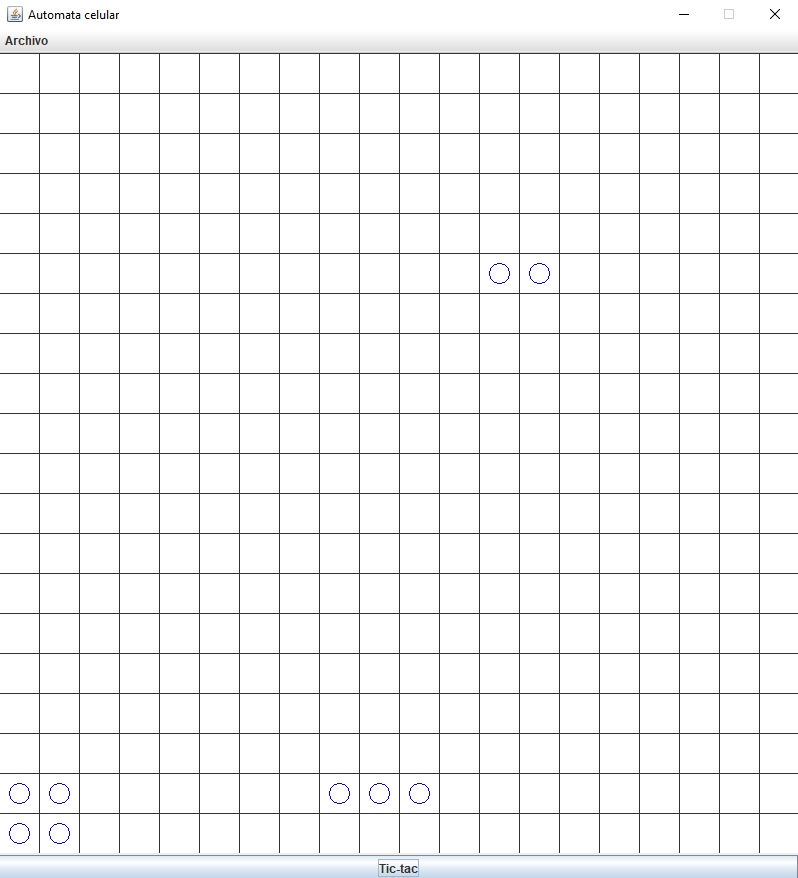
*En este punto vamos a construir la maqueta correspondiente a esta extensión siguiendo el patrón MVC.*

1. ***MODELO:*** --código --
2. ***VISTA:*** *menú de barra. /* --código --
3. ***CONTROLADOR:*** *controladores correspondientes a estas acciones/*--código --



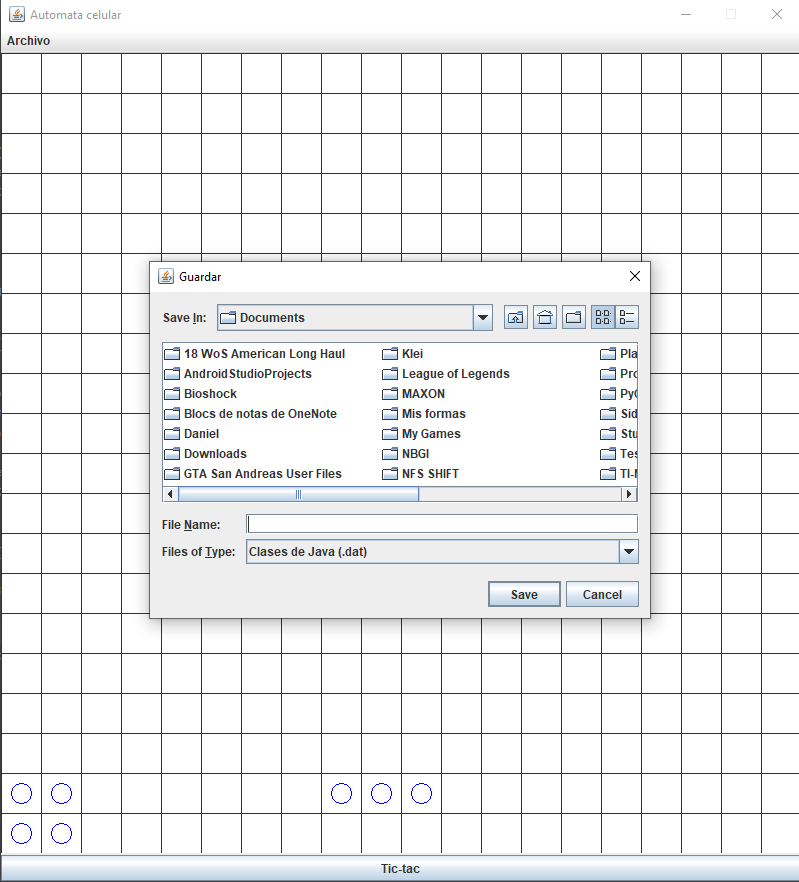
### Implementando salir e iniciar

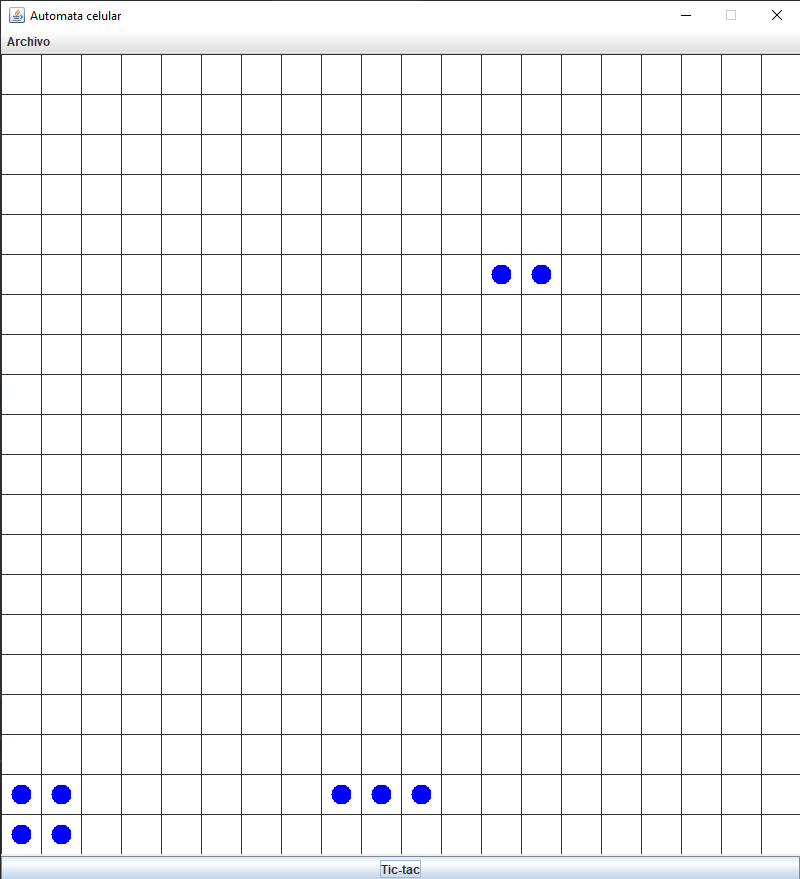
*Las opciones salir e iniciar van a ofrecer los dos servicios estándar de las aplicaciones. El primero no requiere ir a capa de aplicación y el segundo sí.*

1. --código --
2. *método opcionIniciar que crea un nuevo automata.*

### Implementando salvar y abrir

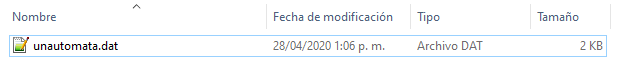
*Las opciones salvar y abrir van a ofrecer servicios de persistencia de un automata como objeto. Los nombres de los archivos deben tener como apellido .dat.*

1.  *método opcionSalvar que une de forma adecuada la capa de presentación con la capa de aplicación. / --código--*



1. *–código--*
2. *Validen este método guardando el automata inicial después de dos clics como*

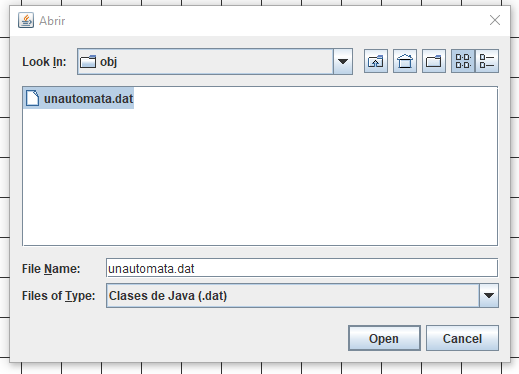
*unautomata.dat.*

*¿El archivo se creó en el disco?*

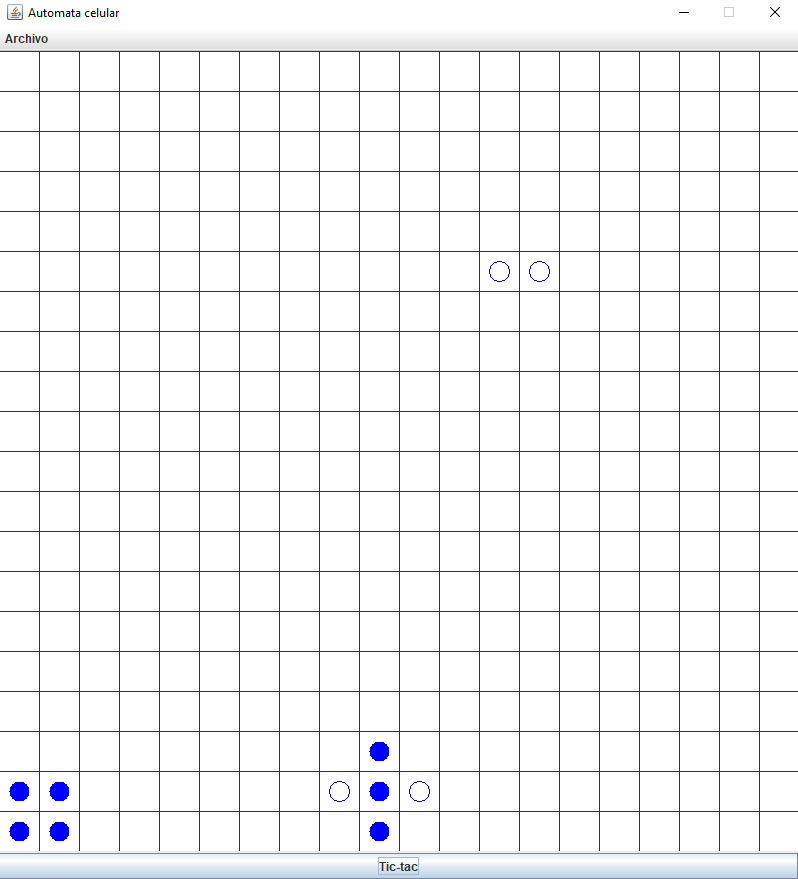
*¿Cuánto espacio ocupa?*

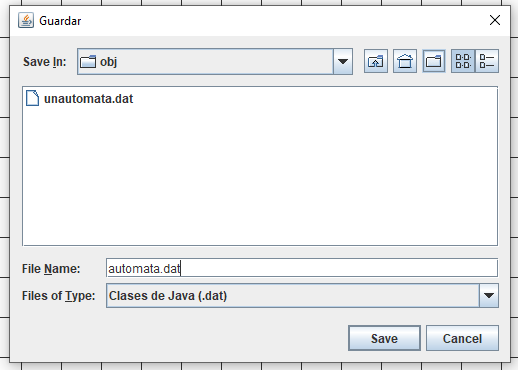
El archivo ocupa 2KB.

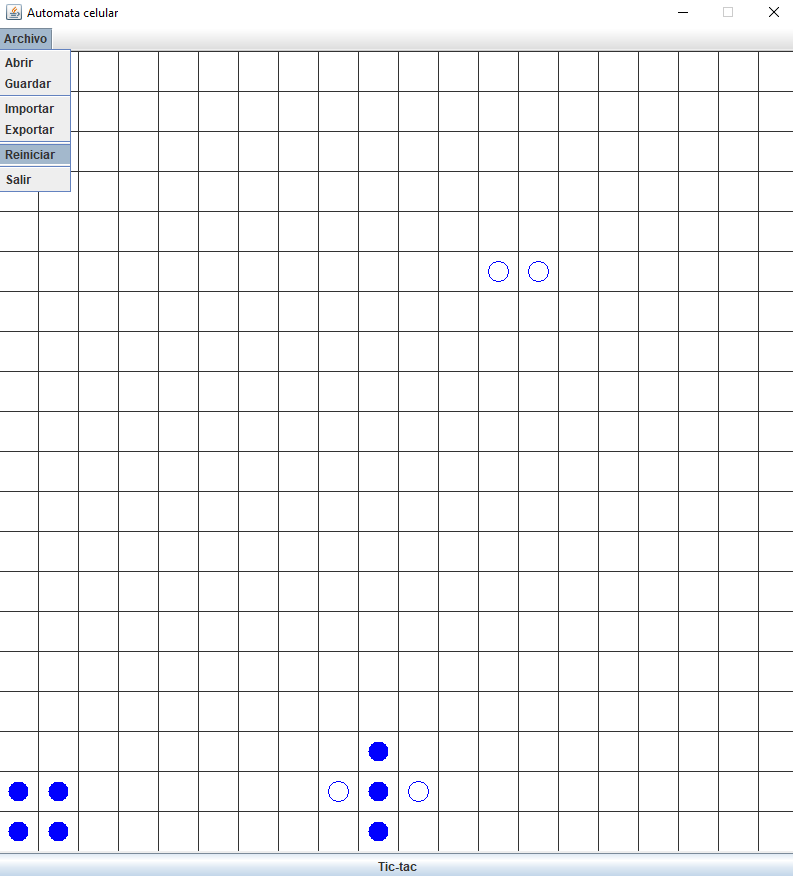
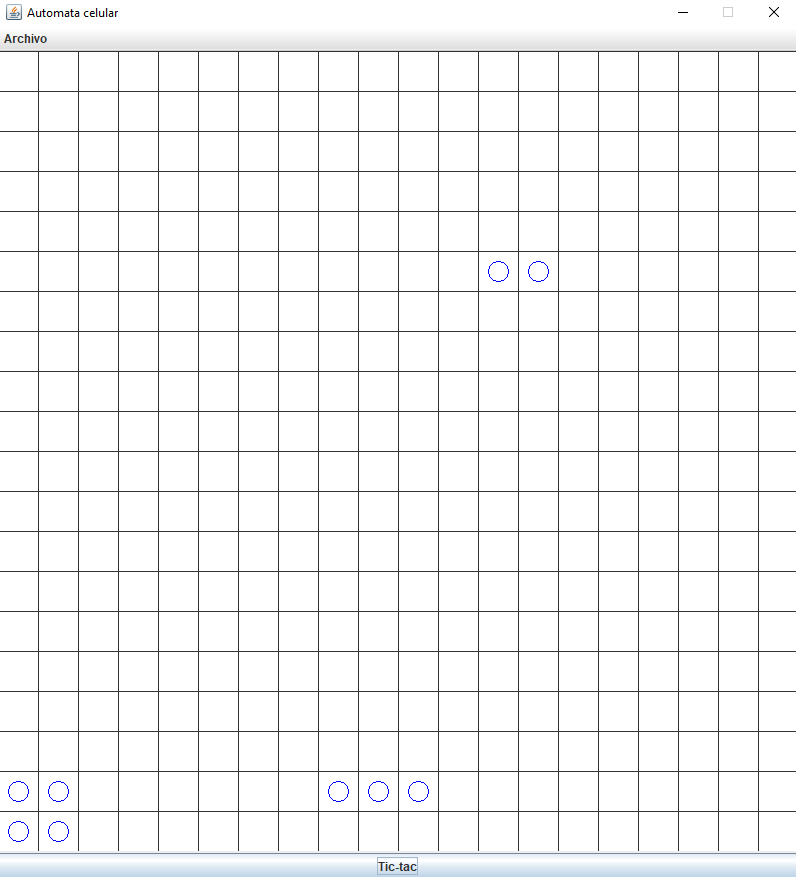
1. *método opcionAbrir que une de forma adecuada la capa de presentación con la capa de aplicación.*

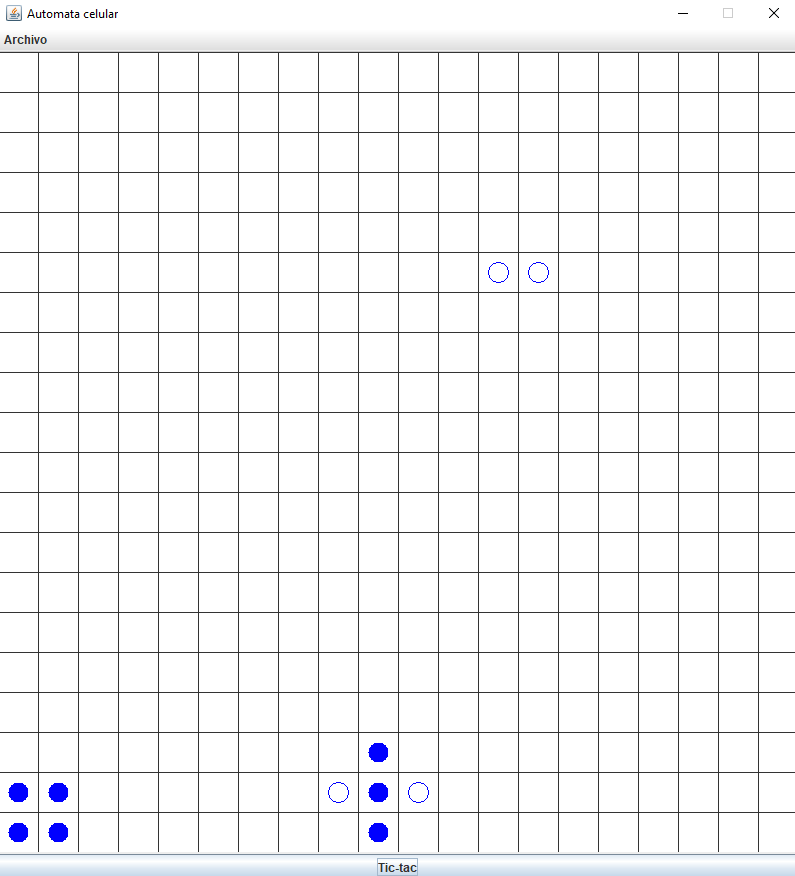
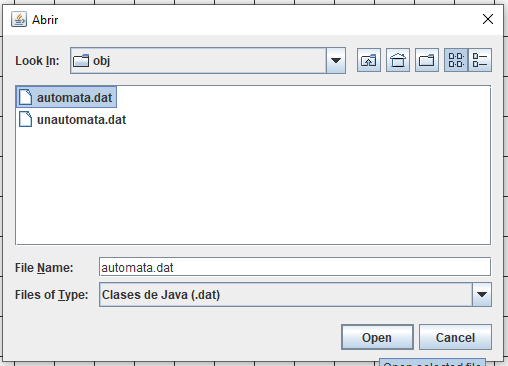


1. *–código—*
2. *prueba de aceptación para este método iniciando la aplicación/–código--*

El estado de la aplicación que se guardará es el que se presenta luego de presionar dos veces el botón tic-tac:

Guardaremos este estado por medio de la opción de “Guardar” de nuestra aplicación, el objeto será guardad en un archivo llamado”autómata.dat”.

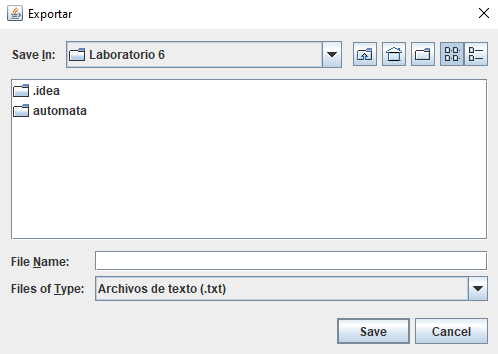
Luego de esto, usaremos el comando “Reiniciar” para limpiar nuestro autómata y regresarlo al estado inicial

Usaremos la opción “abrir” de nuestro programa para cargar el objeto guardado, y este debería retornar nuestro autómata al estado que se guardó

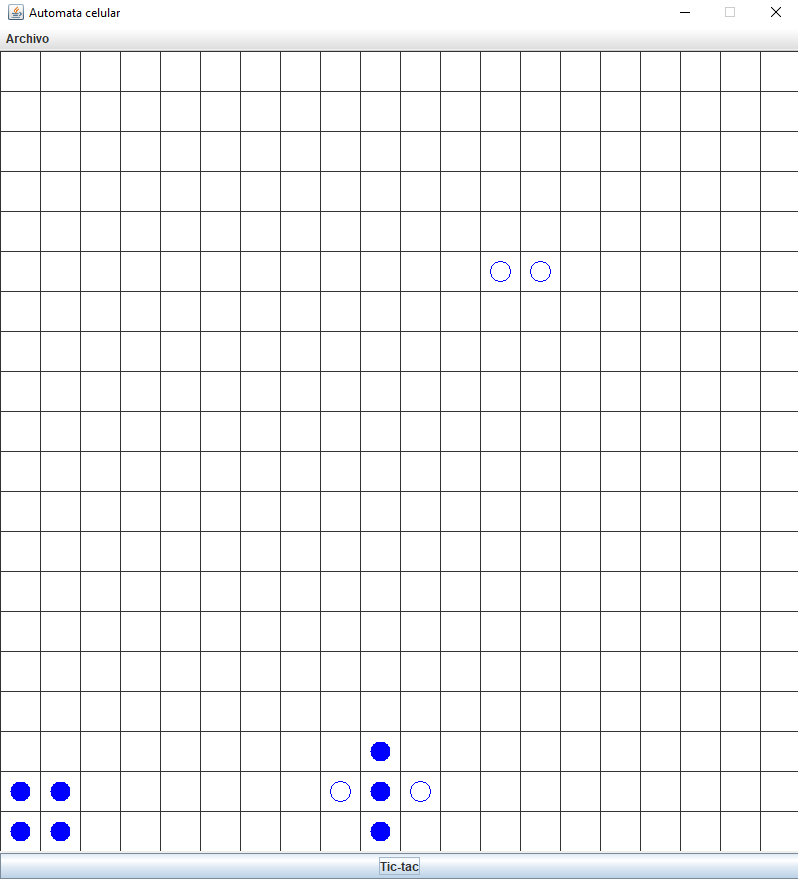
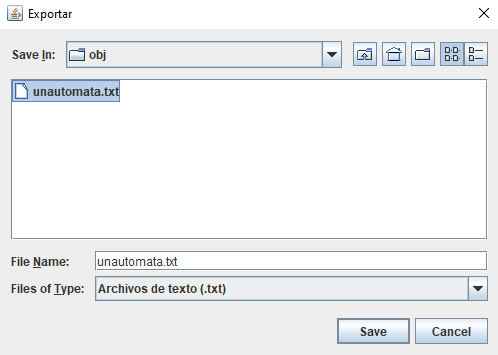
Efectivamente, nuestro programa retornó al estado guardado.

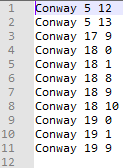
### Implementando importar y exportar

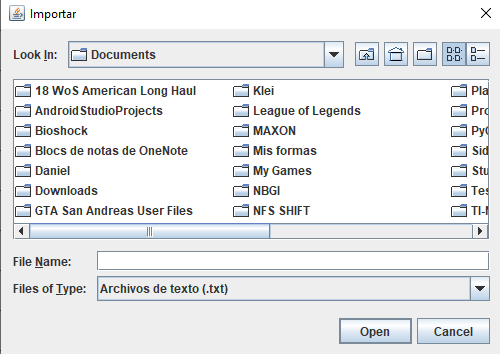
*Estas operaciones nos van a permitir importar información de un automata desde un archivo de texto y exportarlo. Los nombres de los archivos de texto deben tener como apellido .txt*

1. * método opcionExportar que une de forma adecuada la capa de presentación con la capa de aplicación. /--código--*
2. *método exporte que ofrece el servicio de exportar a un archivo texto/--código--*
3. *prueba de aceptación de este método: iniciando la aplicación y exportando como unautomata.txt. Editen el archivo y analicen los resultados. ¿Qué pasó? /--código--*

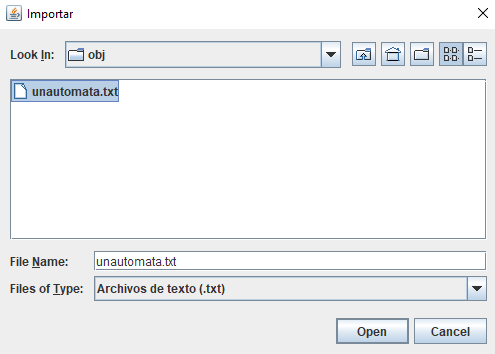
Iniciamos dando dos clicks al botón tic-tac y exportando la configuración del tablero

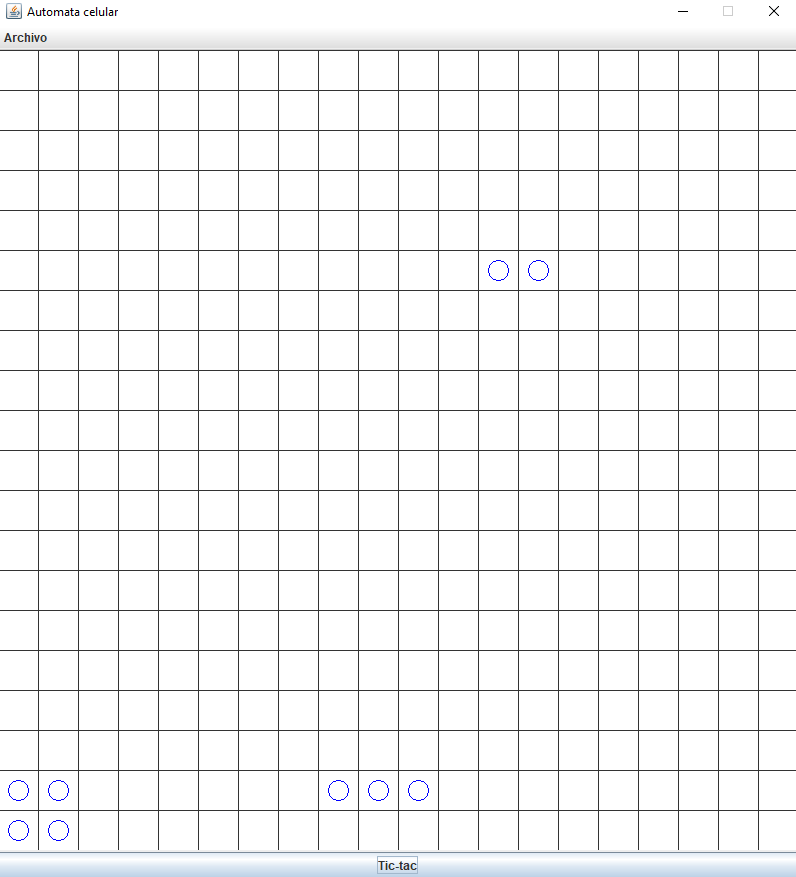


Podemos observar que todas las piezas se encuentran en el archivo

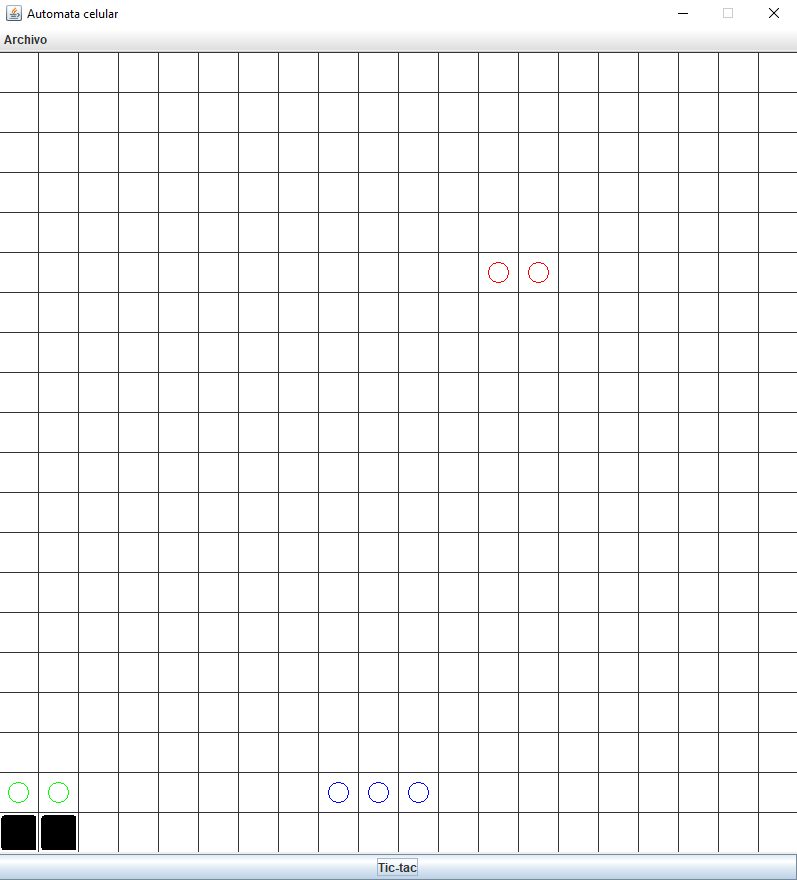
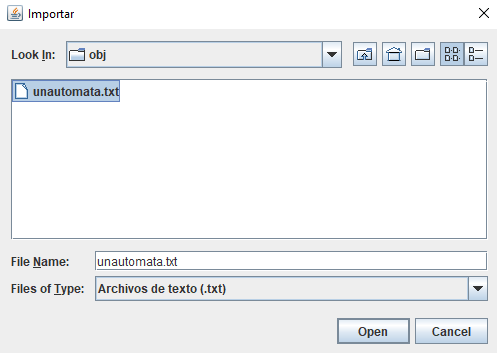
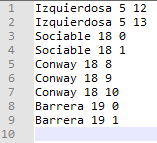
1. *método opcionImportar que une de forma adecuada la capa de presentación con la capa de aplicación. /--código--*
2. *método importe que ofrece el servicio de importar de un archivo texto/--código—*
3. *prueba de aceptación de este par de métodos: iniciando la aplicación exportando a unautomata.txt. saliendo, entrando, creando un nuevo automata e importando el archivo unautomata.txt. /--código—*

*¿Qué resultado obtuvieron? Capturen la pantalla final.*

Ya tenemos el archivo el punto 3, llamado “unautomata.txt”, procederemos a importarlo y ver el resultado:

Como podemos observar, las células se colocaron en el sitio correcto, sin embargo, las células retornaron a su estado inicial, ya que el estado no fue almacenado en ningún momento, tan solo se almacenó el tipo y la posición.

1. *prueba de aceptación de este método escribiendo un archivo de texto correcto en unautomata.txt, e importe este archivo. /--código—*

Se creó el siguiente archivo de texto:

*¿Qué resultado obtuvieron?*

Podemos concluir que la importación está funcionando de manera correcta con diferentes tipos de Elemento.

### Analizando comportamiento

1. *Ejecuten la aplicación, den tres clics, salven a un archivo cualquiera y ábranlo. Describan el comportamiento.*

Al guardar el AutomataCelular y luego volverlo abrir, este se guarda tal cual y continua con su funcionamiento normal.

1. *Ejecuten la aplicación, tres clics, exporten a un archivo cualquiera e importen. Describan el comportamiento.*

Al exportar el AutomataCelular y luego volverlo a abrir, los estados de las células se reinician, solo se conserva el tipo y su posición.

1. *¿Qué diferencias ven entre el comportamiento 1 y 2? Expliquen los resultados.*

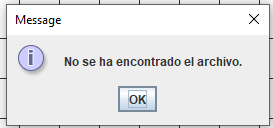
Al salvar, se guarda todo el objeto como tal, esto incluye sus atributos que a su vez pueden ser otros objetos, mientras que al exportar solo se está guardando el tipo de células y sus posiciones, por tanto, en el primero se conserva el estado de los objetos, mientras que en el segundo solo se conservan sus propiedades más básicas.

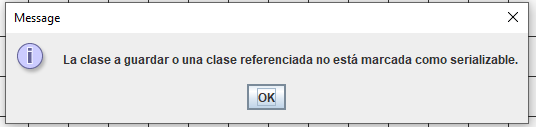
### Perfeccionando salvar y abrir

1. *–código--*
2. *–código--*
3. *prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.*

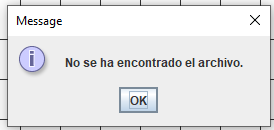
Se intentarán recrear la mayor cantidad de errores, sin embargo, no todos los errores conocemos como producirlos.

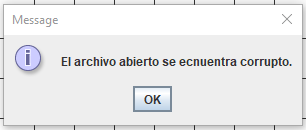
Al guardar un archivo, si la ruta es inválida obtenemos el siguiente error

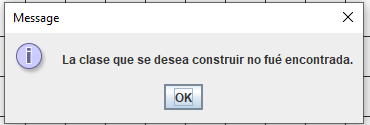


 Si la clase que queremos guardar no está marcada como serializarle obtenemos el siguiente error

Si queremos abrir un archivo, pero le damos una ruta inválida obtendremos el siguiente error



 Si intentamos abrir un archivo que no se guardó correctamente obtendremos el siguiente error:

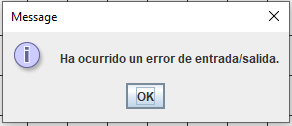
 Por último, si guardamos una clase, y luego al intentar cargarla, está ya no existe, obtendremos el siguiente error:

Como se dijo anteriormente, no se pudieron comprobar todos los errores, pero esto se debe a que hay algunos que ocurren a muy bajo nivel y no es fácil desencadenarlos.

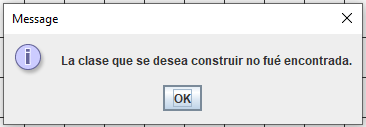
### Perfeccionando importar y exportar.

1. *–código--*
2. *–código--*
3. *prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.*

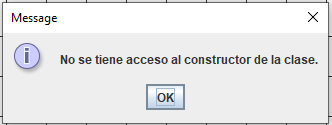
Cuando vamos a exportar, si la ruta especificada no existe, se produce un error de entrada/salida



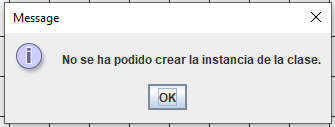
Si al exportar referenciamos una clase que en el momento de importar ya no existe, obtendremos el siguiente error:



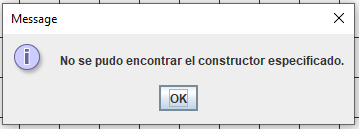
Si intentamos crear una clase cuyo constructor es privado, obtendremos la siguiente excepción:



Si intentamos instanciar una clase con parámetros que no corresponden a ninguno de sus constructores, obtendremos el siguiente error:

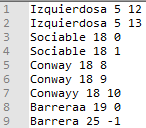
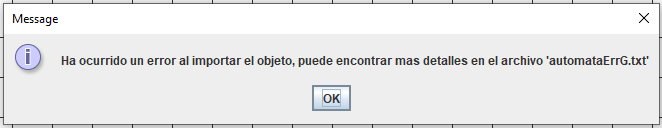


Si no se encuentra el constructor de una clase, obtendremos el siguiente error:



### Perfeccionando importar. Hacia un minicompilador.

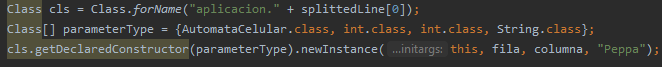
1. *–código--*
2. *--* *código--*
3. *archivo con errores, automataErr.txt.*





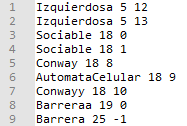
### Perfeccionando importar. Hacia un minicompilador flexible.

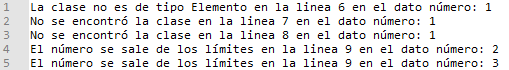
1. *-- código--*
2. *-- código--*
3. *archivo de pruebas, llámelo automataErrG.txt, para probar la flexibilidad.*



Este fragmento de código nos permite instanciar cualquier clase que implemente la interfaz Elemento y que tenga el constructor por defecto de Célula.

Extendimos nuestro compilador, permitiendo que verifique si una Clase existe pero no implementa la interfaz Elemento.





### RETROSPECTIVA

1. *¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes*

*?(12/12)*

1. *¿Cuál es el estado actual del laboratorio? ¿Por qué?*

Completo

1. *Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?*

Para este laboratorio consideramos que fue útil la práctica de refactorización, debido a que toco realizar algunos cambios en los parámetros de algunas clases, para hacer una reestructuración más adecuada para el correcto funcionamiento.

1. *¿Cuál consideran fue el mayor logro? ¿Por qué?*

Lograr la importación, debido a debimos crear una clase a partir de un solo un String con su nombre

1. *¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?*

Lograr la importación, debido a que no se tenia una claridad clara para su correcta implementación, se realizó una investigación para poder lograr su adecuado funcionamiento.

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

Plantear diferentes soluciones he implementaciones para el correcto funcionamiento.