# PROGRAMACIÓN ORIENTADA A OBJETOS

**I/O 2020-01**

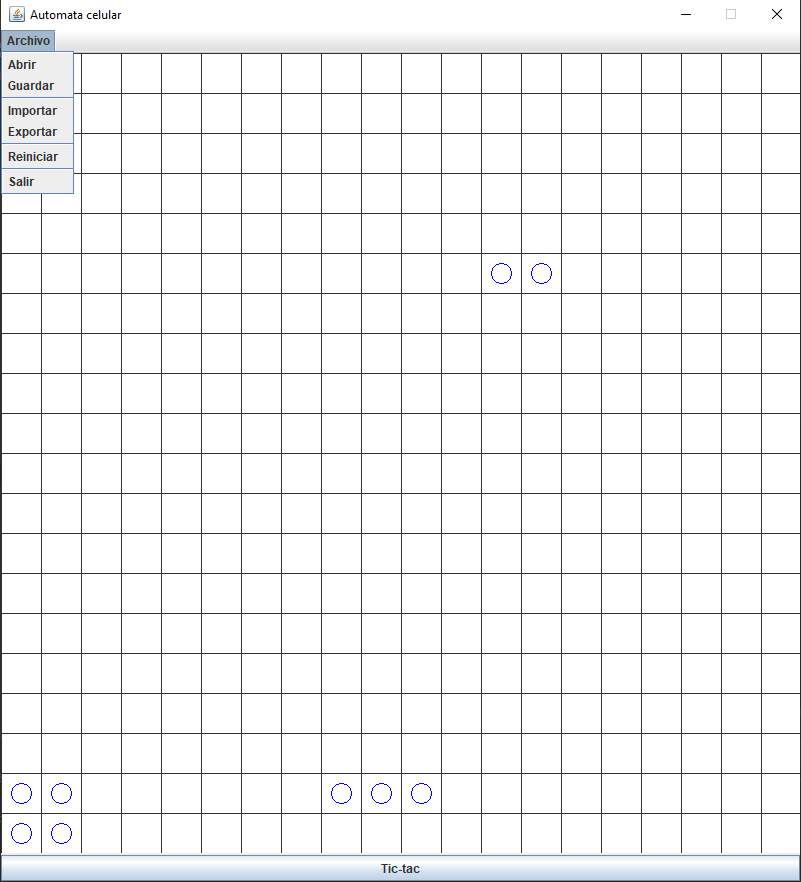
**Laboratorio 6/6 [ :) ]**

## Creando la maqueta

**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

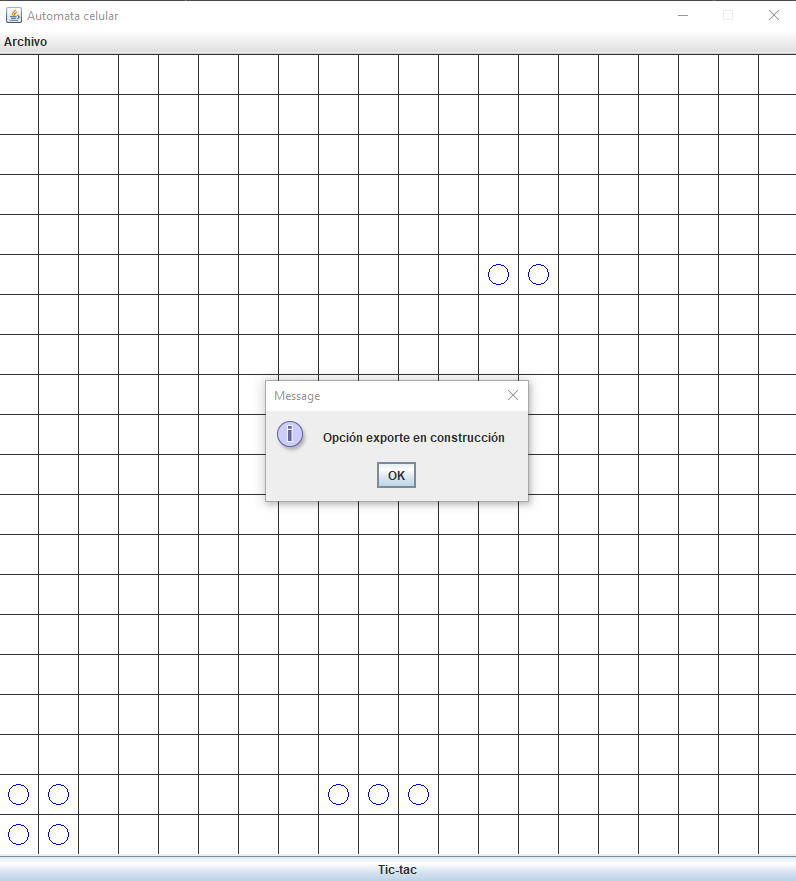
En este punto vamos a construir la maqueta correspondiente a esta extensión siguiendo el patrón MVC.

1. **MODELO:** Preparen los métodos correspondientes a reiniciar y a las cuatro opciones básicas de entrada-salida (salve,abra,exporte,importe). Los métodos deben simplemente propagar una automataExcepcion con el mensaje: “Opción … en construcción”. Los métodos de entrada salida deben tener un parámetro File.
2. **VISTA:** Construyan un menú barra que ofrezca, además de las opciones básicas de entrada-salida, las opciones estándar de iniciar y salir. Para esto creen el método prepareElementosMenu. Capturen la pantalla correspondiente.



1. **CONTROLADOR:** Construyan los controladores correspondientes a estas acciones. Para esto creen el método prepareAccionesMenu y los métodos base del controlador

( opcionSalver,opcionAbir, opcionExportar, opcionImportar, opcionReiniciar, opcionCerrar), Estos métodos, por ahora, **llaman directamente el método correspondiente de la capa de aplicación**. Capturen una pantalla significativa.



## Implementando salir e iniciar

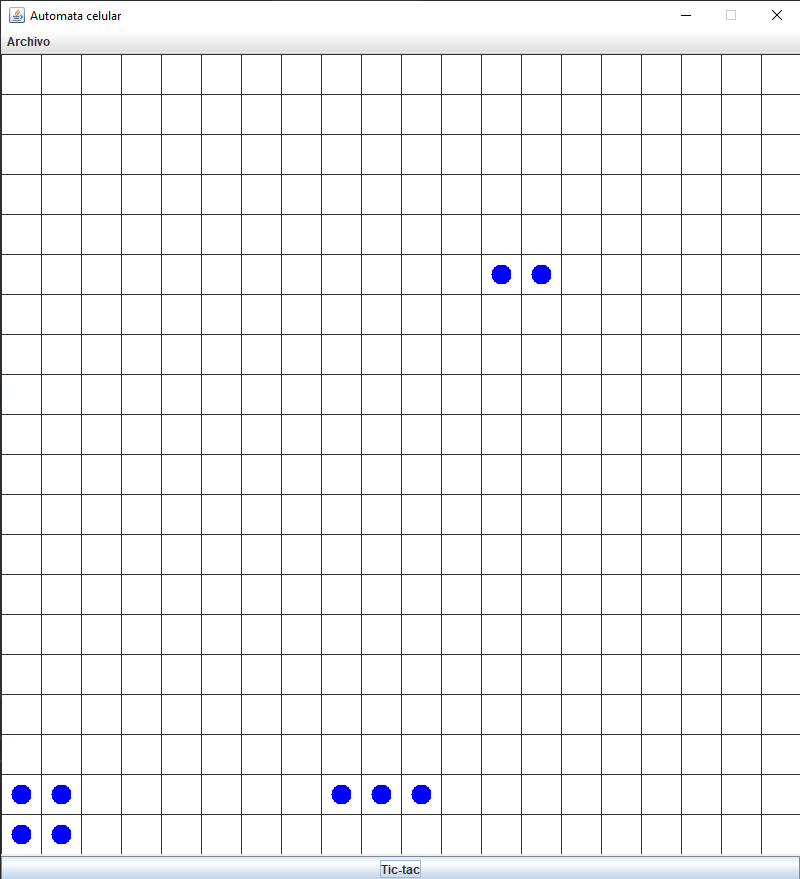
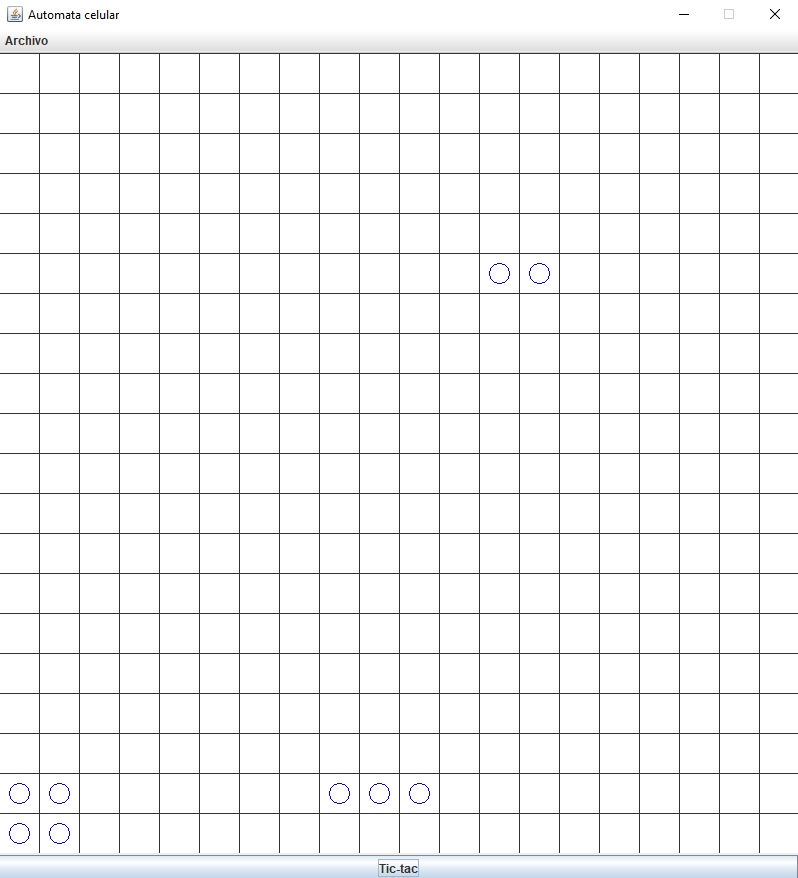
**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

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. Construyan el método opcionSalir que hace que se termine la aplicación. No es necesario incluir confirmación.

✓

1. Construyan el método opcionIniciar que crea un nuevo automata. Capturen una pantalla significativa.

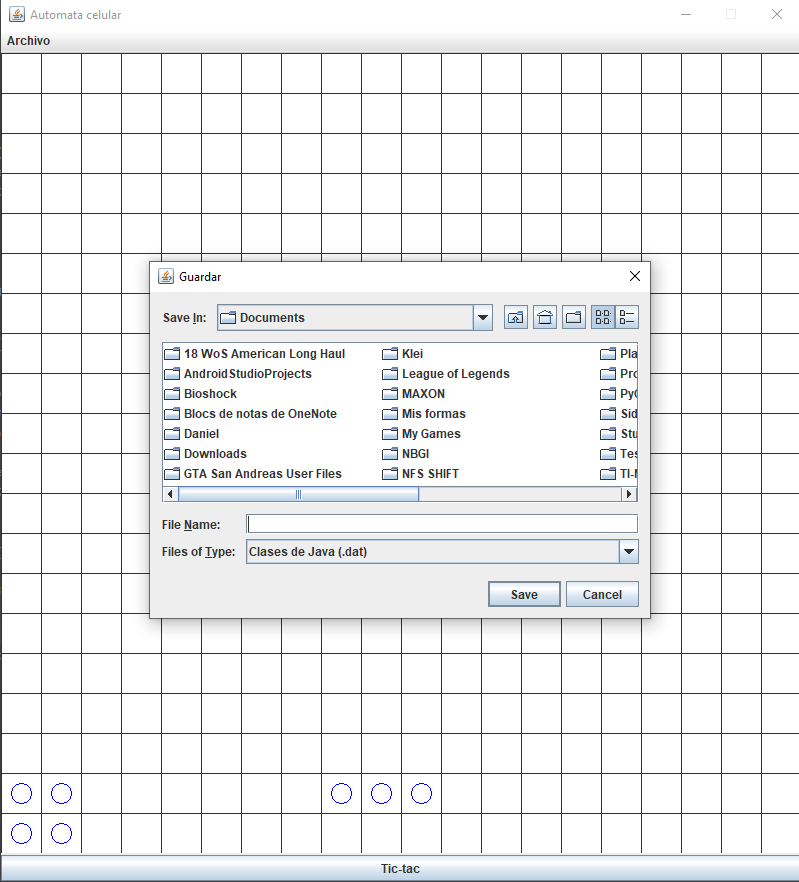


## Implementando salvar y abrir

**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

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. Construyan el método opcionSalvar que une de forma adecuada la capa de presentación con la capa de aplicación. Usen un FileChooser y atiendan la excepción. Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una pantalla significativa.

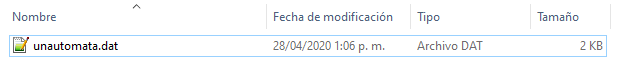


1. Construyan el método salve que ofrece el servicio de guardar en un archivo el estado actual del automata.

✔

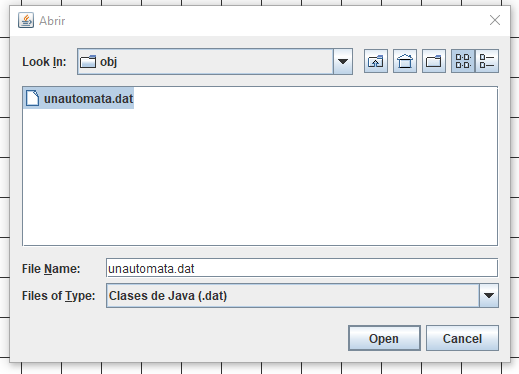
1. Validen este método guardando el automata inicial después de dos clics como

unautomata.dat. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?



El archivo ocupa 2KB.

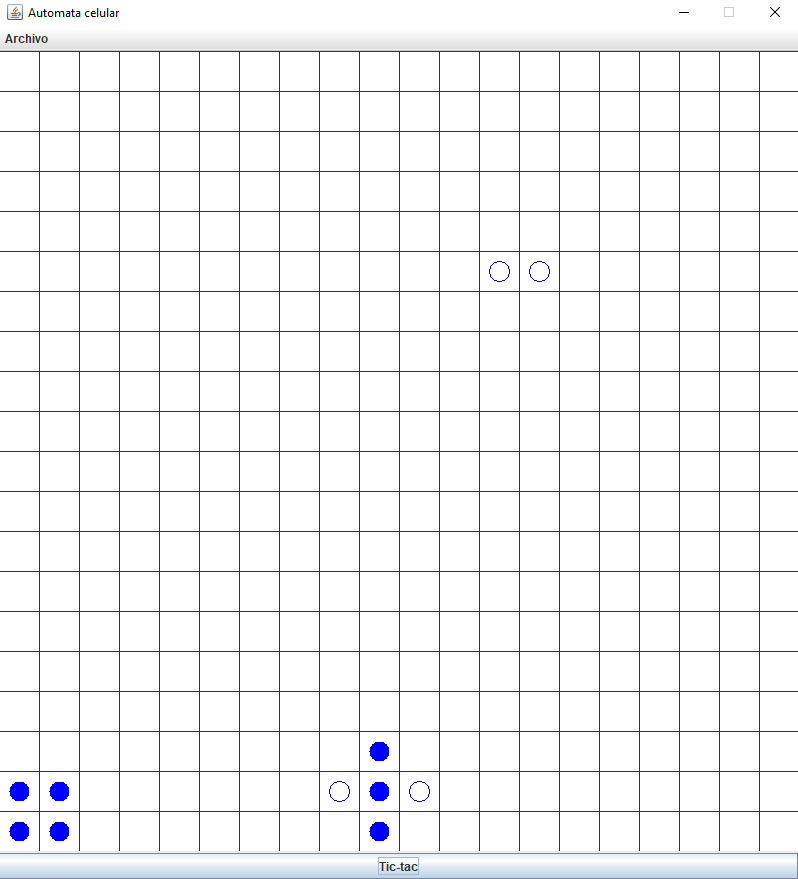
1. Construyan el método opcionAbrir que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una pantalla significativa.

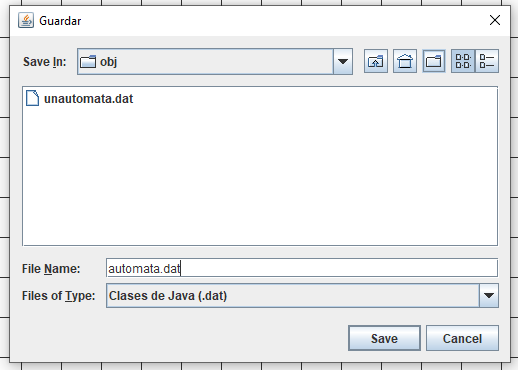


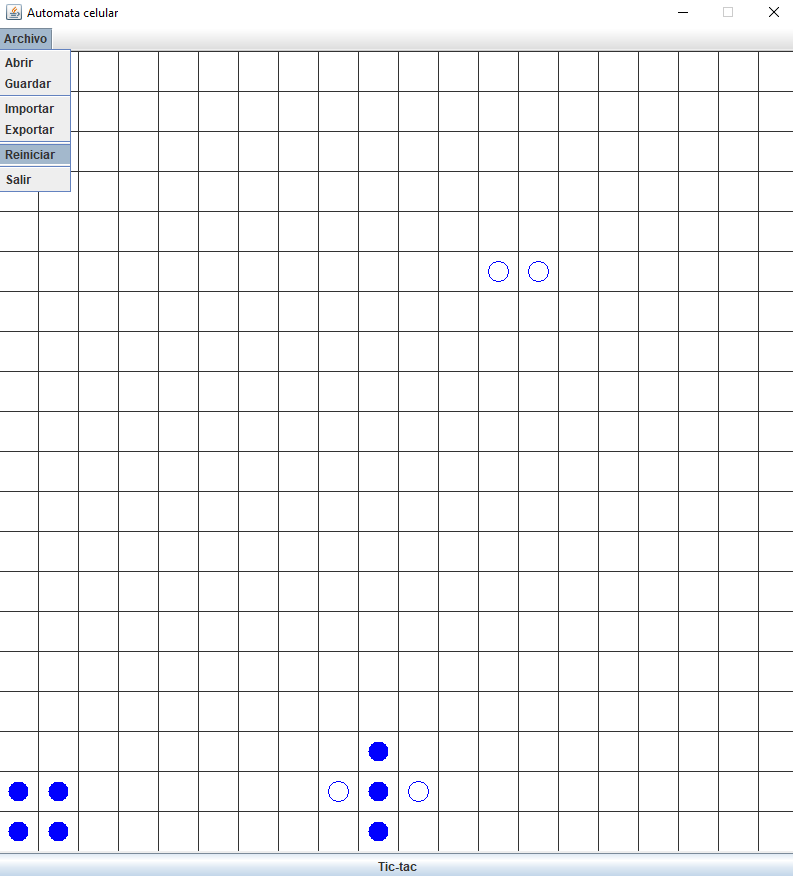
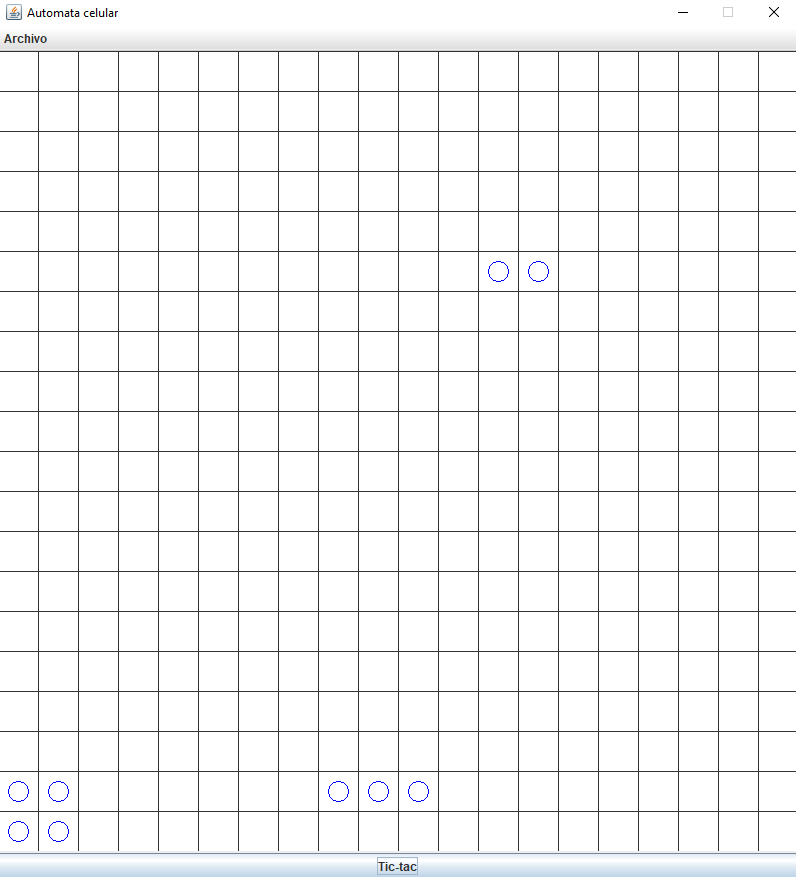
1. Construyan el método abra que ofrece el servicio de leer un automata de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general.

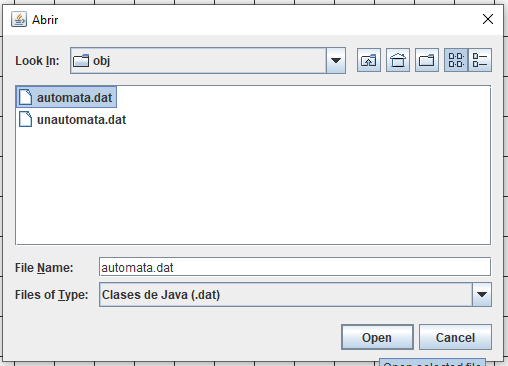
✔

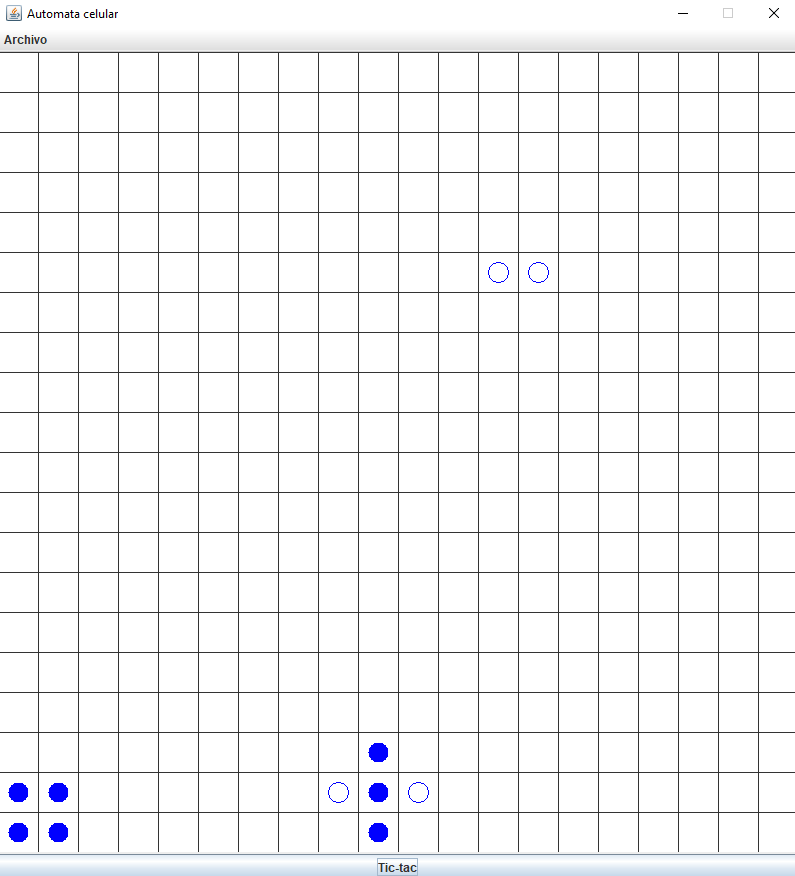
1. Realicen una prueba de aceptación para este método iniciando la aplicación, creando una nueva situación en el automata y abriendo el archivo unautomata.dat. Capturen imágenes significativas de estos resultados.

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

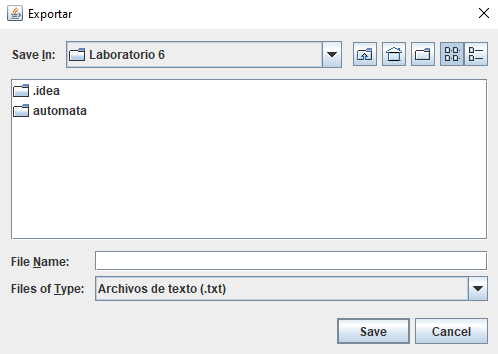
**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

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

Los archivos texto tienen una línea de texto por cada elemento

En cada línea asociada un elemento se especifica el tipo y la posición. Cedula 20 20

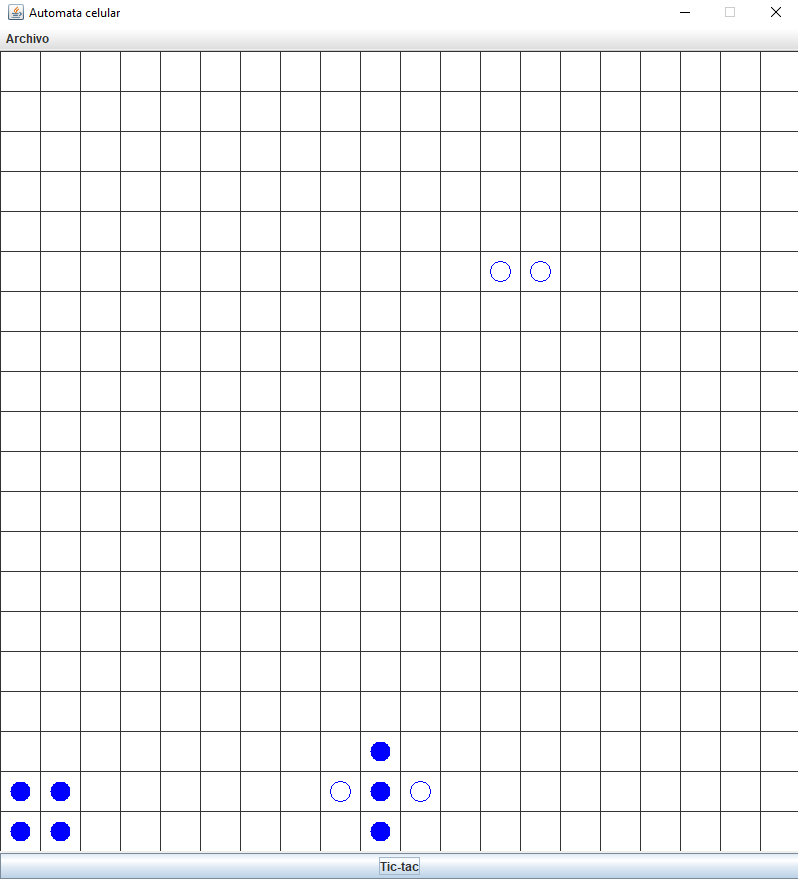
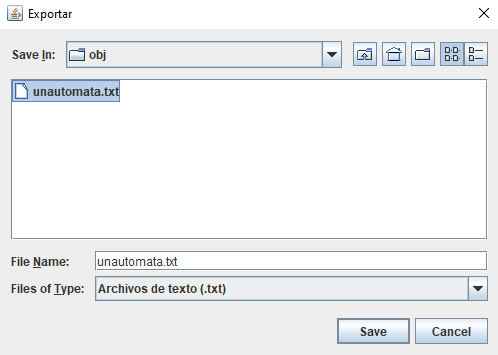
Barrera 50 50

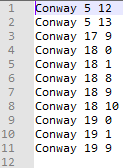
1. Construyan el método opcionExportar que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación y capturen una pantalla significativa.
2. Construyan el método exporte que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado actual.

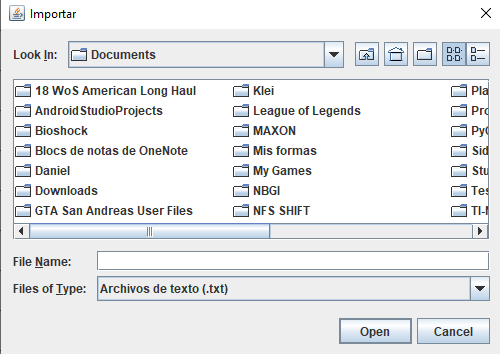
✔

1. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como unautomata.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?

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



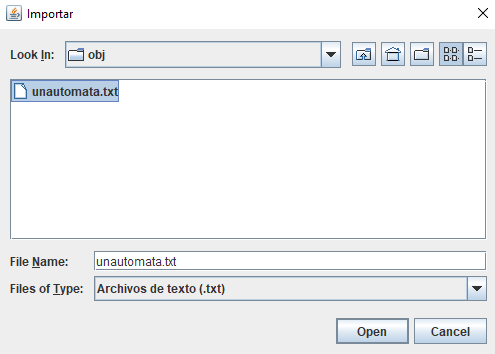
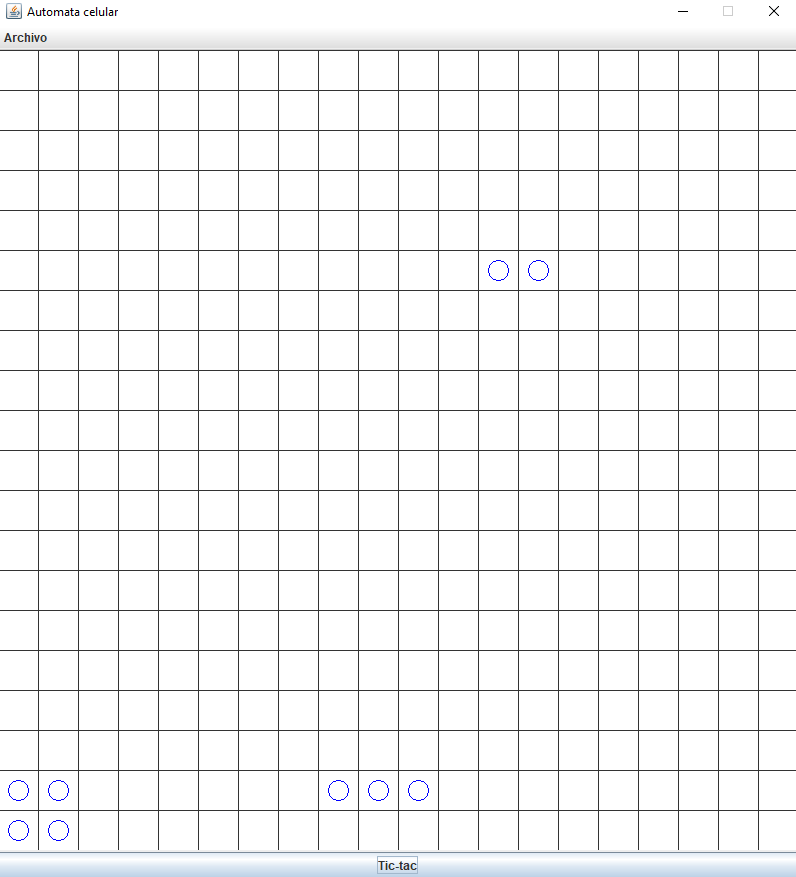
Podemos observar que todas las piezas se encuentran en el archivo

1. Construyan el método opcionImportar que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación y capturen una pantalla significativa.
2. Construyan el método importe que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considere un mensaje de error general.

(Consulten en la clase String los métodos trim y split)

✔

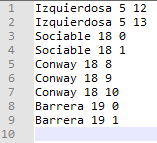
1. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a unautomata.txt. saliendo, entrando, creando un nuevo automata e importando el archivo unautomata.txt. ¿Qué resultado obtuvieron? Capturen la pantalla final.

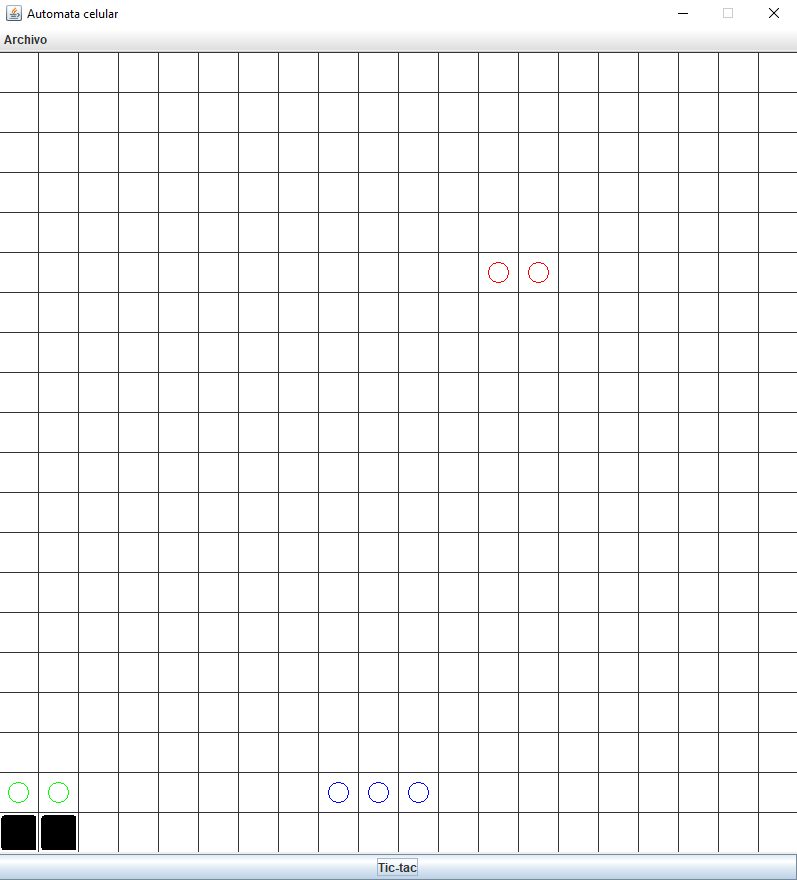
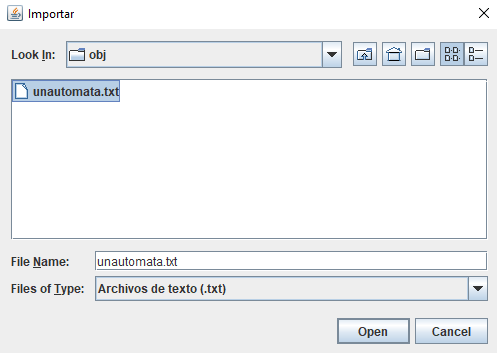
Ya tenemos el archivo el punto 3, llamado “unautomata.txt”, procederemos a importarlo y ver el resultado:

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. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto correcto en unautomata.txt, e importe este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.

Se creó el siguiente archivo de texto:



Y el resultado observado fue el siguiente:

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

## Analizando comportamiento

**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

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 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 le tipo de celulas y sus posiciones.

## Perfeccionando salvar y abrir

**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

1. Copien las versiones actuales de abra y salve y renómbrenlos como abra01 y salve01
2. Perfeccionen el manejo de excepciones de los métodos abra y salve detallando los errores.
3. Realicen una prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

## Perfeccionando importar y exportar.

**[En lab06.doc, \*.asta , automataErr.txt \*.java] [NO OLVIDEN BDD y MDD]**

1. Copien las versiones actuales de importe y exporte y renómbrenlos como importe01

y exporte01

1. Perfeccionen el manejo de excepciones de los métodos importe y exporte detallando los errores.
2. Realicen una prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

## Perfeccionando importar. Hacia un minicompilador.

**[En lab06.doc, \*.asta , automataErr.txt \*.java] [NO OLVIDEN BDD y MDD]**

1. Copien las versiones actuales de importe y exporte y renómbrenlos como importe02 y

exporte02

1. Perfeccionen el método **importe** para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador): número de línea donde se encontró el error, palabra que tiene el error y causa de error.
2. Escriban otro archivo con errores, llámelo automataErr.txt, para ir arreglándolo con ayuda de su “importador”. Presente las pantallas que contengan los errores.

## Perfeccionando importar. Hacia un minicompilador flexible.

**[En lab06.doc, \*.asta , automataFlex.txt \*.java] [NO OLVIDEN BDD y MDD]**

1. Copien las versiones actuales de importe y exporte y renómbrenlos como importe03 y

exporte03

1. Perfeccionen los métodos importe y exporte para que pueda servir para cualquier tipo de elementos creados en el futuro

(Investiguen cómo crear un objeto de una clase dado su nombre)

1. Escriban otro archivo de pruebas, llámelo automataErrG.txt, para probar la flexibilidad. Presente las pantallas que contenga un error significativo.

# RETROSPECTIVA

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

?(Horas/Hombre)

1. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
2. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
3. ¿Cuál consideran fue el mayor logro? ¿Por qué?
4. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?