LABORATORIO 6

PROGRAMACION ORIENTADA A OBJETOS

ELABORADO POR:

DIEGO FERNANDO RUIZ ROJAS

JUAN FELIPE AGUAS PULIDO

PRESENTADO A: MARIA IRMA DIAZ ROZO

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

PROGRAMACIÓN ORIENTADA A OBJETOS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D. C.

30 DE ABRIL DE 2021

**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 en la clase fachada del modelo los métodos correspondientes a las

cuatro opciones básicas de entrada-salida (abra, guarde, importe y exporte). Los

métodos deben simplemente propagar una automataExcepcion con el mensaje genérico

de “Opción en construcción”. Los métodos deben tener un parámetro File.

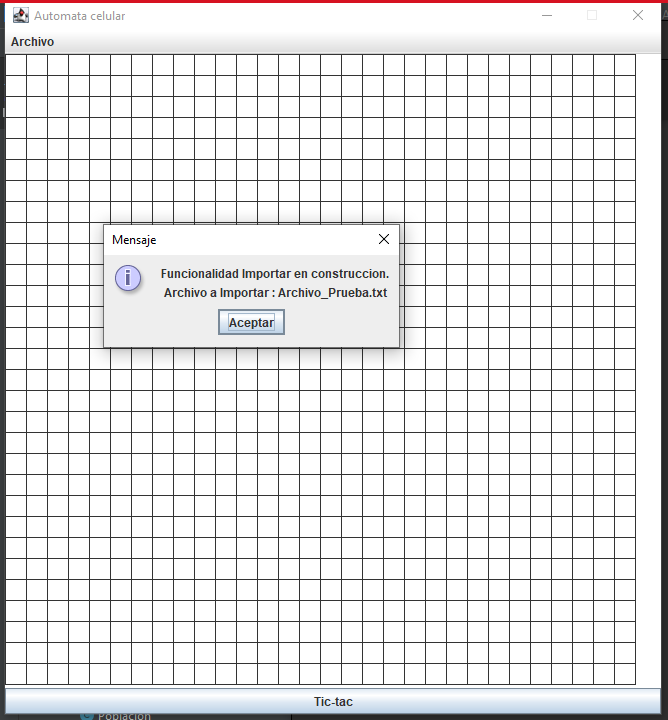
2. VISTA : Construyan un menú barra que ofrezca, además de las opciones básicas de

entrada-salida, las opciones estándar de nuevo y salir (Nuevo, Abrir, Guardar como,

Importar, Exportar como, Salir). No olviden incluir los separadores. Para esto creen

el método prepareElementosMenu. Únicamente debe funcionar la vista. Capturen la

pantalla correspondiente.



3. CONTROLADOR: Construyan los oyentes orrespondientes a las seis opciones. Para esto

creen el método prepareAccionesMenu y los métodos base del controlador

( opcionAbir,opcionGuardar, opcionExportar, opcionImportar, opcionNuevo,

opcionSalir), Estos métodos, por ahora, llaman directamente el método

correspondiente de la capa de dominio. No incluyan todavía el FileChooser.

Capturen una pantalla significativa.



**Implementando salir y nuevo**

[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 dominio y el segundo sí.

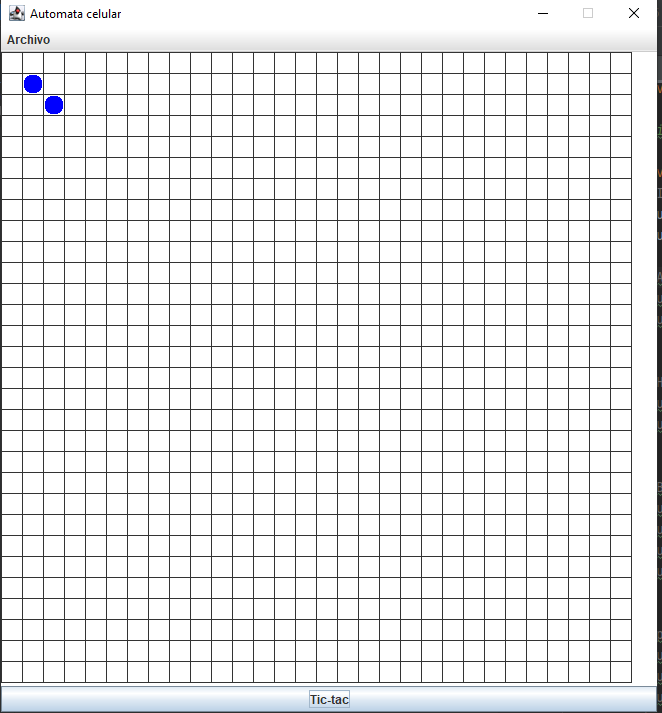
1. Construyan el método opcionSalir que hace que se termine la aplicación. No es

necesario incluir confirmación.

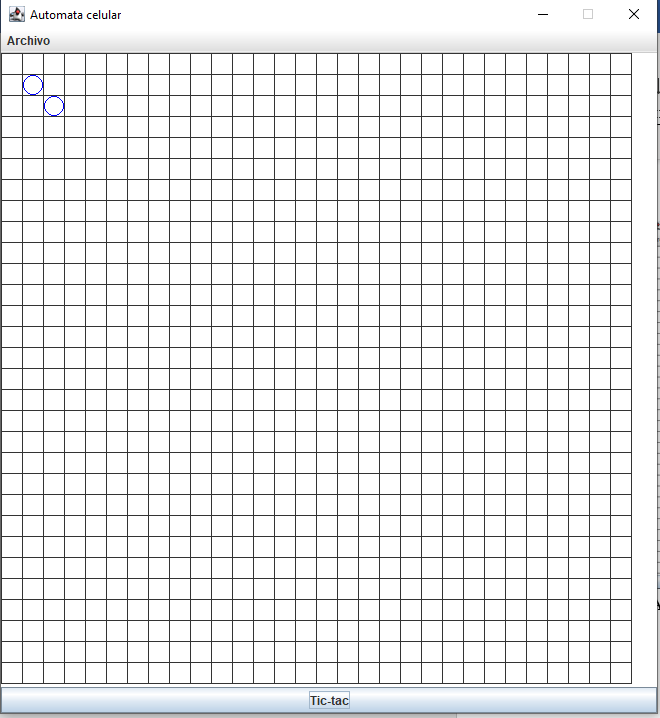
2. Construyan el método opcionNuevo que crea una nueva automata. Capturen una

pantalla significativa.

* Autómata anterior



* Autómata Nuevo



**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 una automata como

objeto. Los nombres de los archivos deben tener como extensión .dat.

1. Copien las versiones actuales de abra y guarde y renómbrenlos como abra00 y

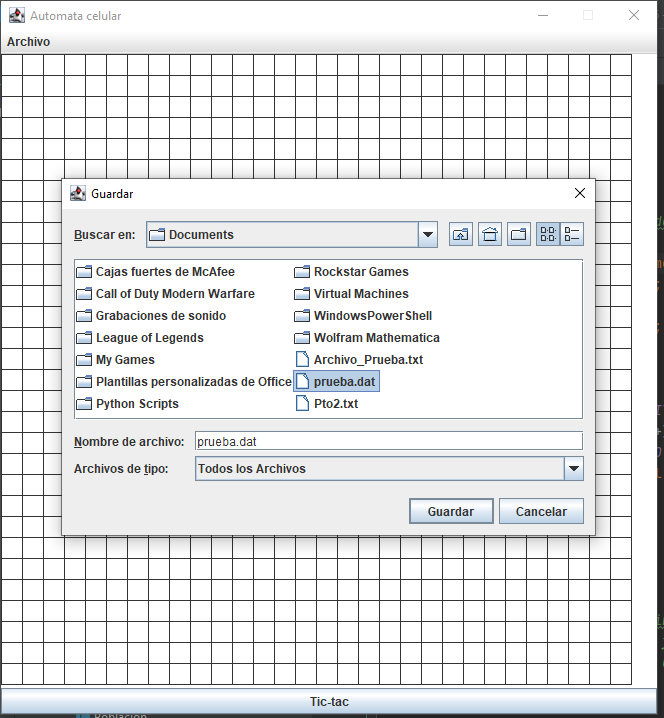
guarde00

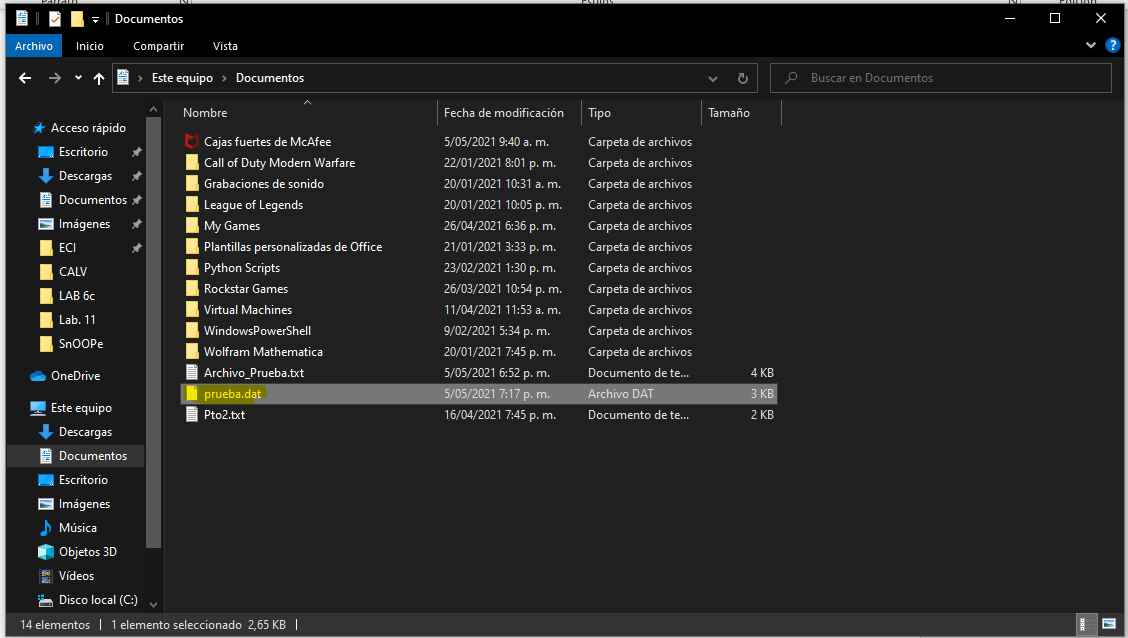
2. Construyan el método opcionGuardar que une de forma adecuada la capa de

presentación con la capa de dominio. Usen un FileChooser y atiendan la excepción.

Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una

pantalla significativa.





3. Construyan el método guarde que ofrece el servicio de guardar en un archivo el estado

actual del automata.

4. Validen este método guardando la isla inicial después de dos clics como

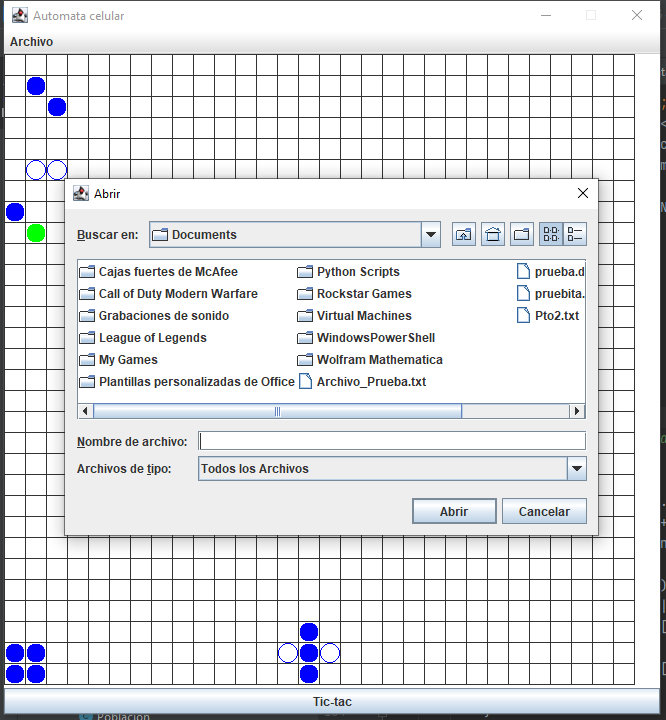
unaautomata.dat. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?

Si el archivo se creo en la carpeta seleccionada y ocupa 4Kb de espacio.

5. Construyan el método opcionAbrir que une de forma adecuada la capa de

presentación con la capa de dominio. Ejecuten la aplicación probando las diferentes

opciones del FileChooser y capturen una pantalla significativa.



6. 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.

7. Realicen una prueba de aceptación para este método iniciando la aplicación, creando

una nueva situación en el automata y abriendo el archivo unaautomata.dat. Capturen

imágenes significativas de estos resultados.

**Implementando importar y exportar**

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

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

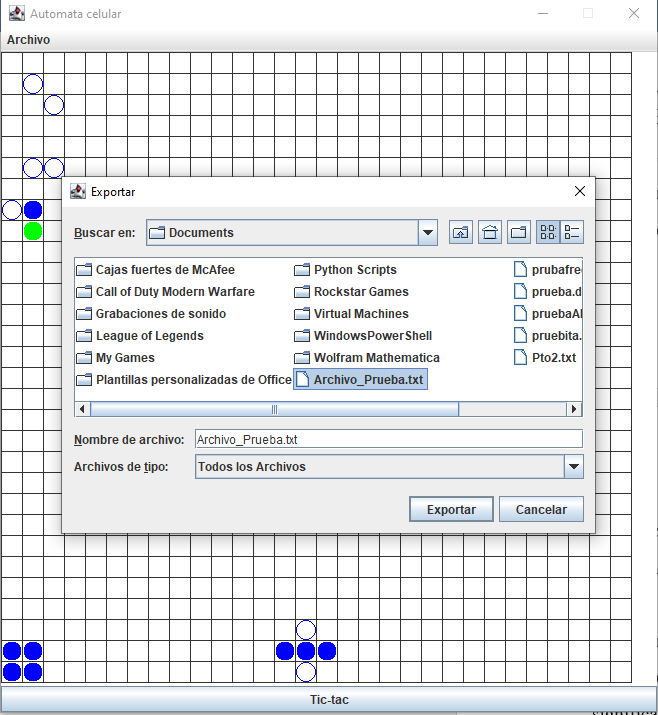
1. Copien las versiones actuales de importe y exporte y renómbrenlos como importe00 y

exporte00

2. Construyan el método opcionExportar que une de forma adecuada la capa de

presentación con la capa de dominio. Ejecuten la aplicación y capturen una pantalla

significativa.



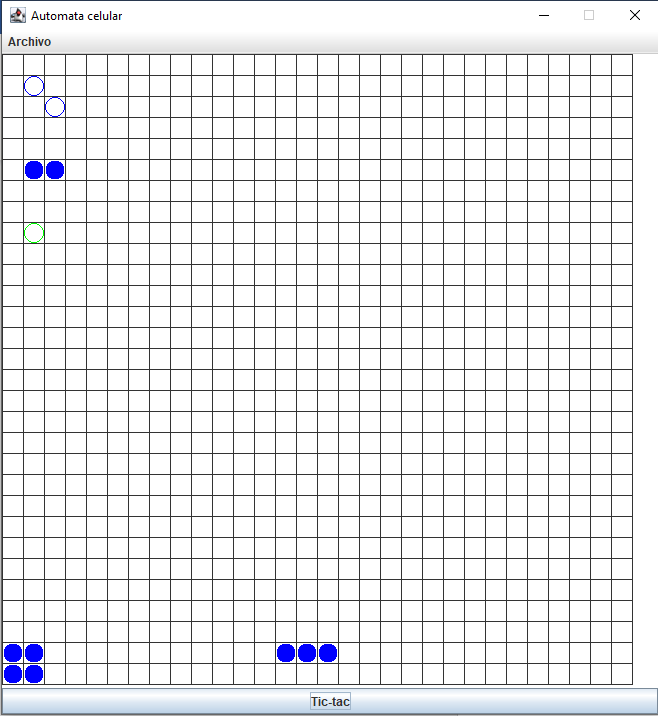
3. Construyan el método exporte que ofrece el servicio de exportar a un archivo texto,

con el formato definido, el estado actual.

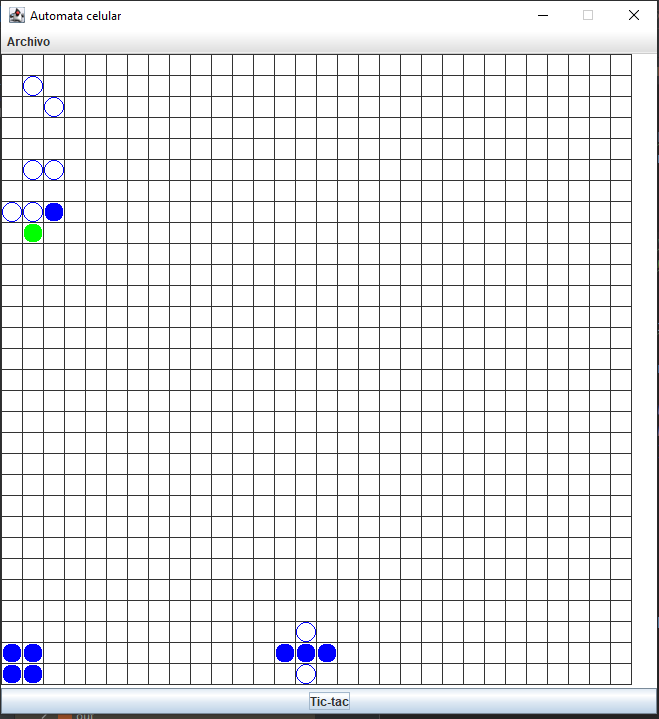
4. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando

como unaautomata.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?

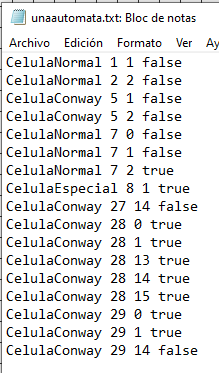
Inicializamos el autómata y hacemos 10 tictacs.



Luego de los 10 tictacs



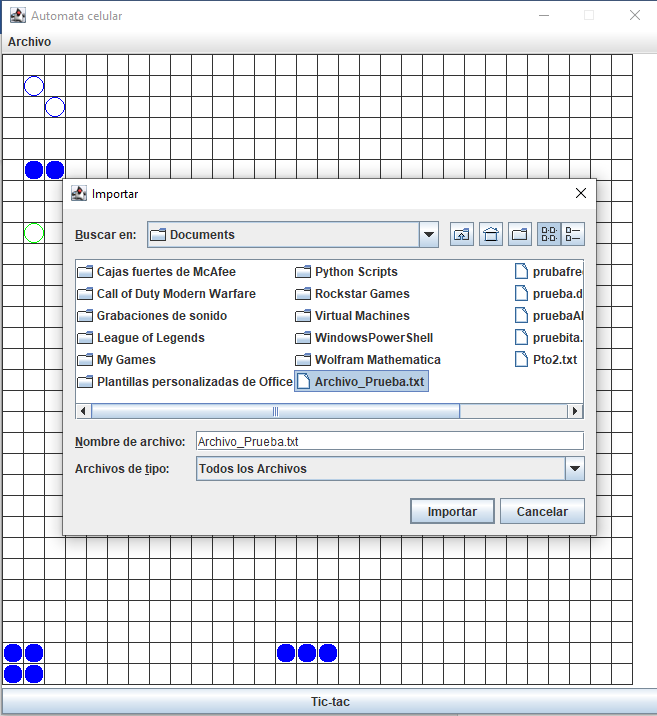
Archivo .txt generado con la información del autómata



5. Construyan el método opcionImportar que une de forma adecuada la capa de

presentación con la capa de dominio. Ejecuten la aplicación y capturen una pantalla

significativa.



6. 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)

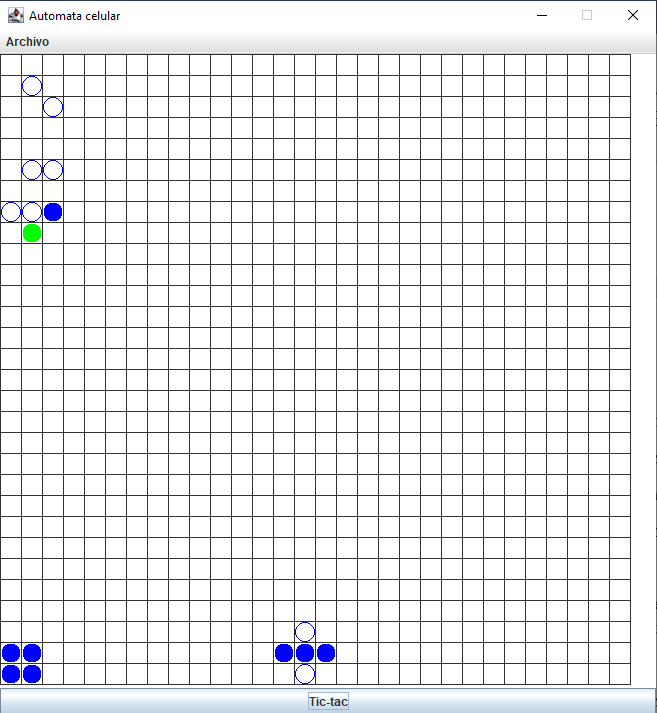
7. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación

exportando a unaautomata.txt. saliendo, entrando, creando un nuevo autómata e

importando el archivo otraautomata.txt. ¿Qué resultado obtuvieron? Capturen la

pantalla final.

Luego de realizar la exportación del .txt generado en el punto 6 del autómata, creamos un nuevo autómata importamos este .txt y nos da el siguiente resultado.

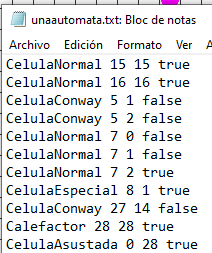


8. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto

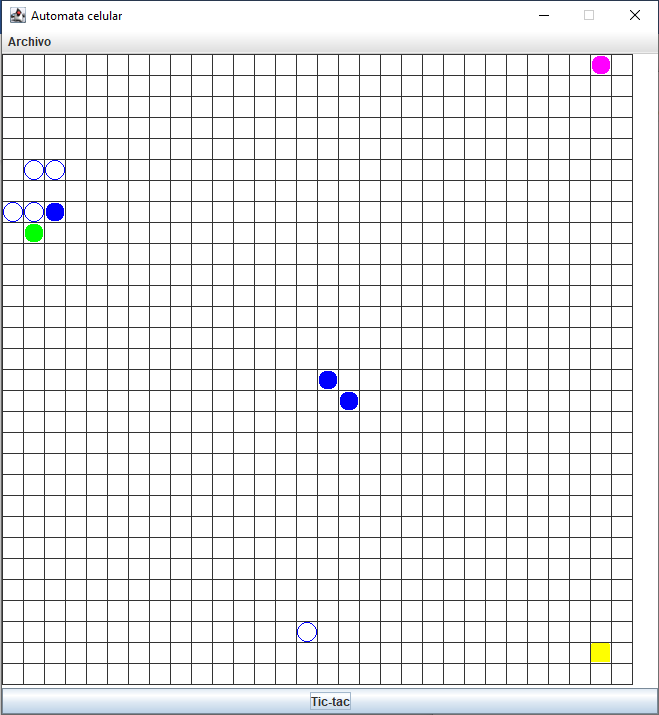
correcto en unaautomata.txt. e importe este archivo. ¿Qué resultado obtuvieron?

Capturen la pantalla.

Escribimos el siguiente archivo



Al importarlo obtuvimos este resultado



**Analizando comportamiento**

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

1. Ejecuten la aplicación, den tres clics, guarden a un archivo cualquiera y ábranlo.

Describan el comportamiento

Al guardarlo luego de tres clics y abrirlo vemos que la funcionalidad esta bien implementada pues lo abre en el mismo estado que se guardó, con respecto al archivo .dat este no es entendible para una persona.

2. Ejecuten la aplicación, tres clics, exporten a un archivo cualquiera e importen. Describan

el comportamiento

Al exportarlo luego de tres clics e importarlo vemos que la funcionalidad está bien implementada pues lo abre en el mismo estado que se guardó, con respecto al archivo .txt este es claro con las células que hay en el autómata informando su tipo, posición y estado.

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

En cuestión de funcionabilidad son iguales pero los archivos que se generan son muy diferentes puesto que el .txt esta enfocado en el entendimiento del usuario y el .dat solo es para que lo entienda el programa.

**Perfeccionando salvar y abrir**

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

1. Copien las versiones actuales de abra y guarde y renómbrenlos como abra01 y

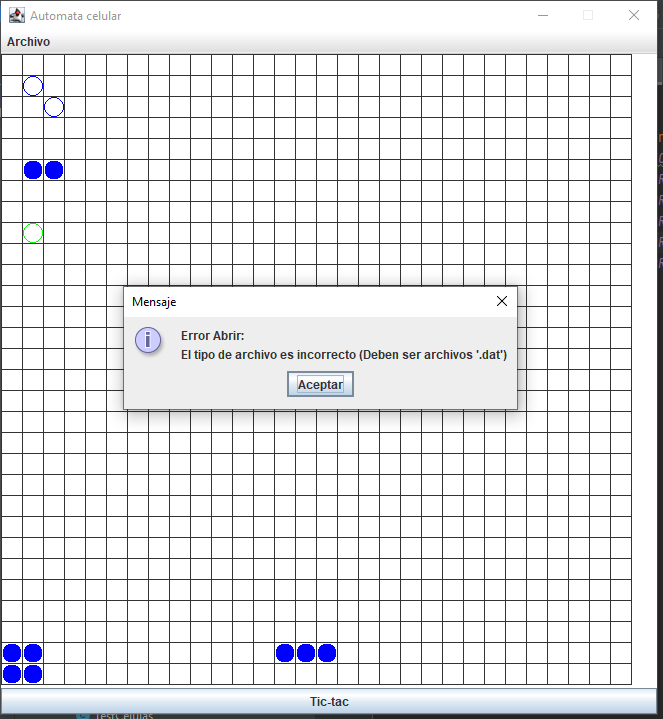
guarde01

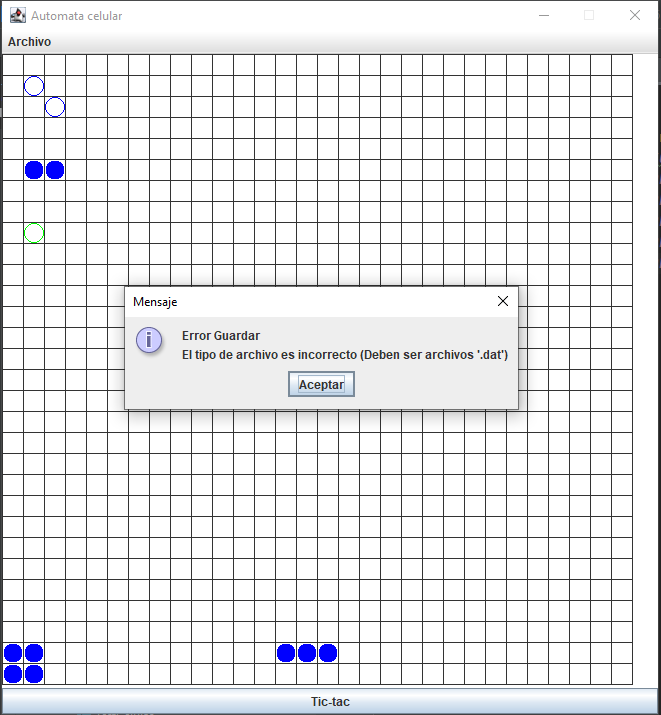
2. Perfeccionen el manejo de excepciones de los métodos abra y guarde detallando los

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

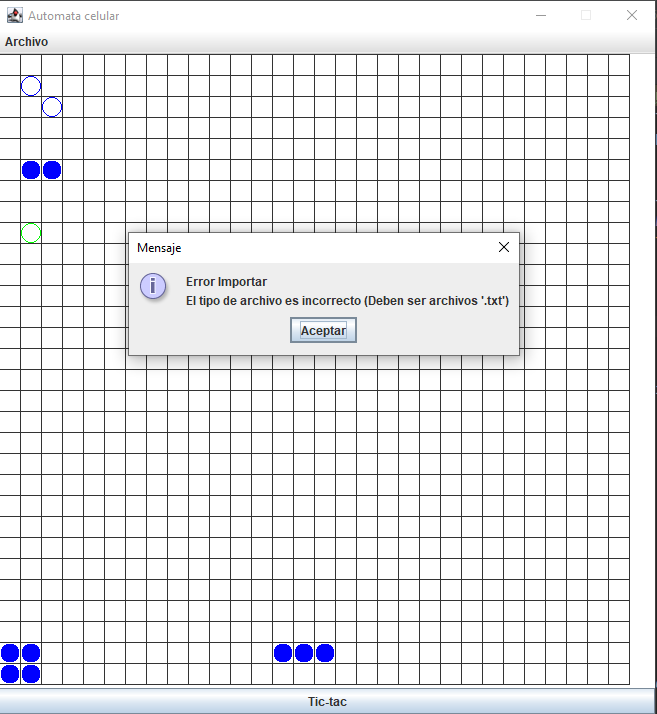
y exporte01

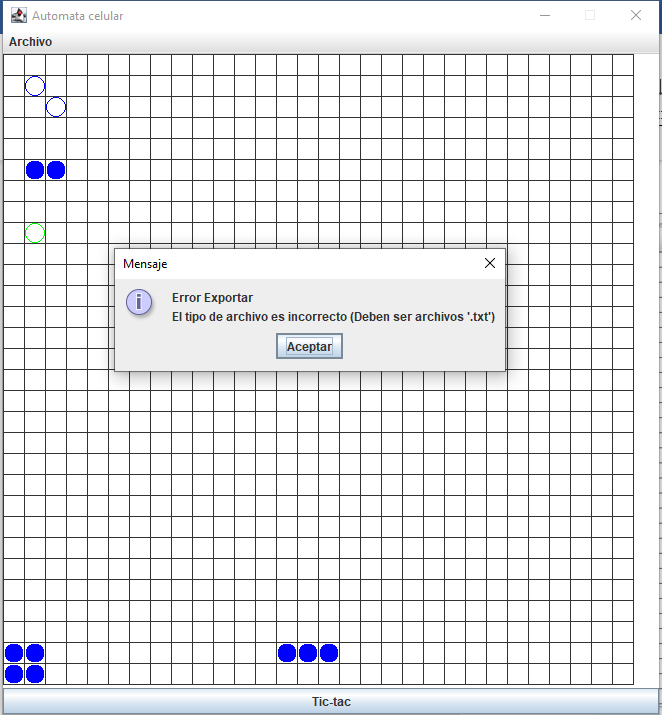
2. Perfeccionen el manejo de excepciones de los métodos importe y exporte detallando

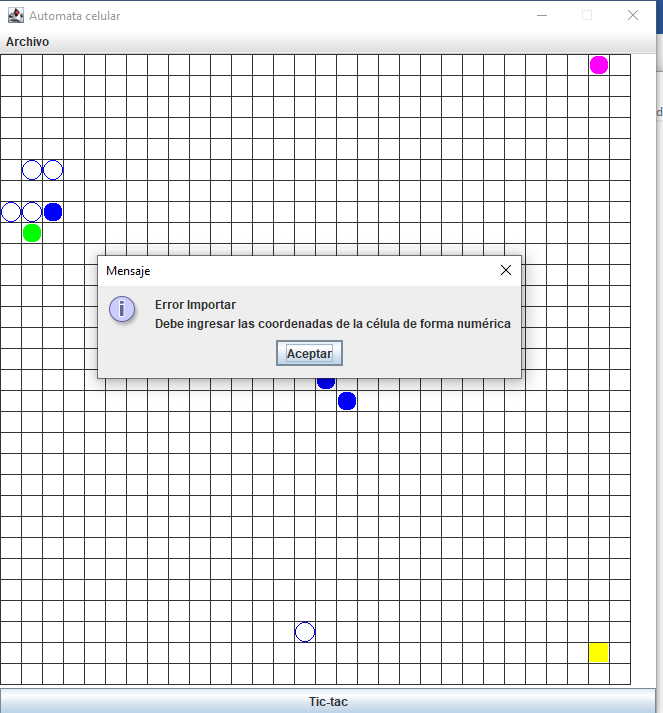
los errores.

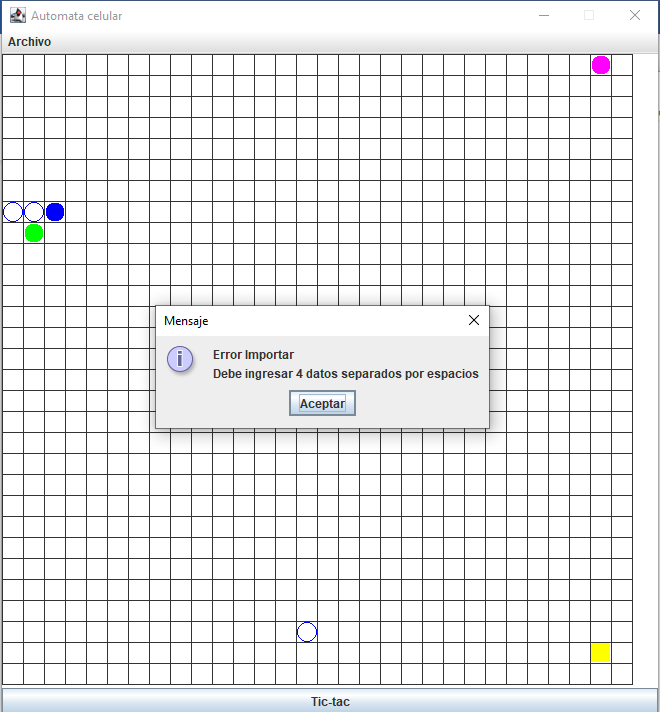
3. Realicen una prueba de aceptación para validar cada una de los nuevos mensajes

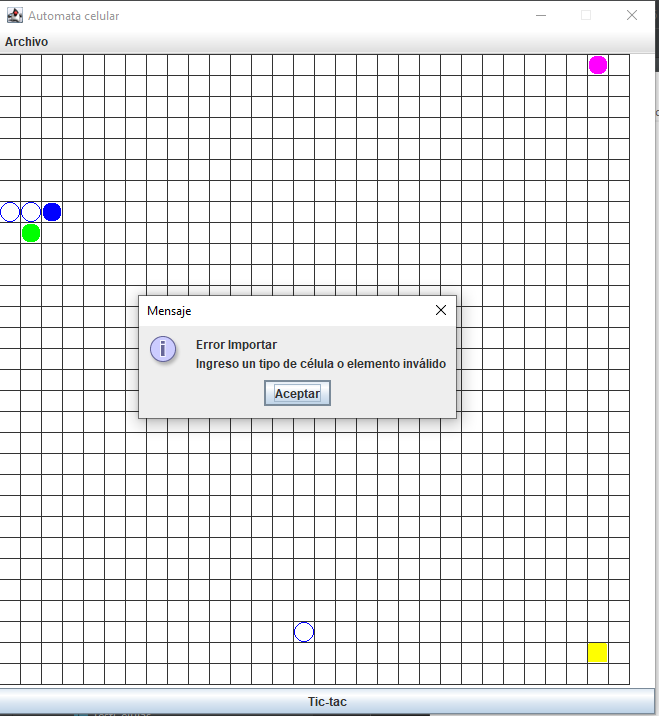
diseñados, ejecútenla y capturen la pantalla final.











**Perfeccionando importar. 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

2. Perfeccionen el método importe para que, además de los errores generales, en las

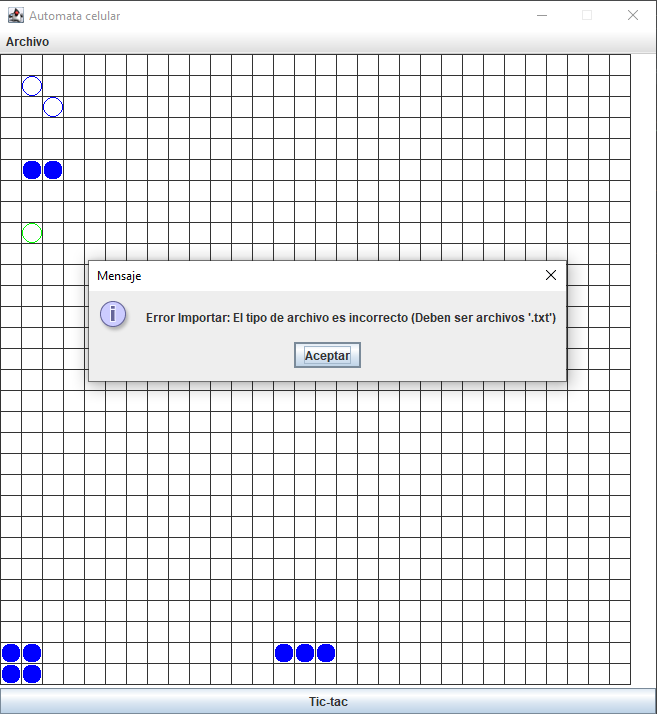
excepciones indique el detalle de los errores encontrados en el archivo (como un

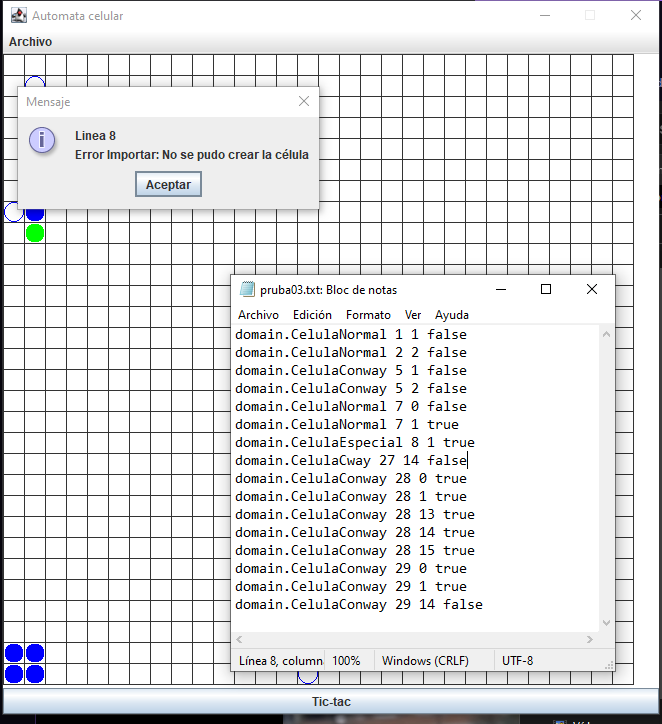
compilador) : número de línea donde se encontró el error, palabra que tiene el error y

causa de error.

3. 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

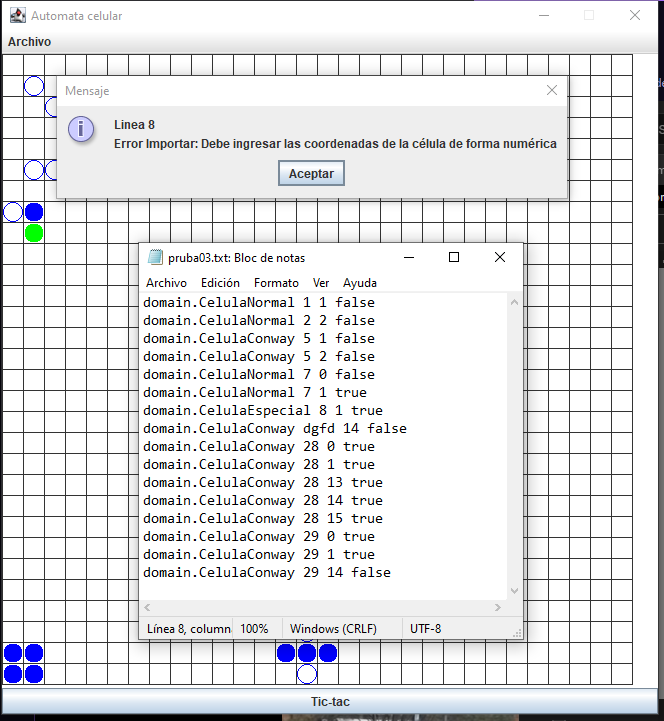
2. Perfeccionen los métodos importe y exporte para que pueda servir para cualquier tipo

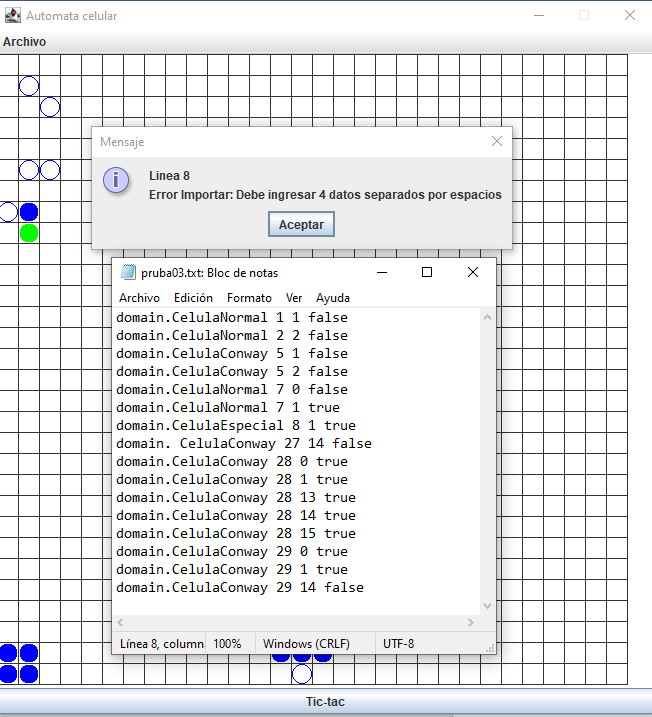
de elementos creados en el futuro

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

3. 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)

Aguas – 10 horas

Ruiz – 9 horas

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

El laboratorio esta finalizado porque cumplimos con los requerimientos de todos los ciclos

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

Refactor puesto que al realizar el laboratorio y avanzar en los ciclos tuvimos que refactorizar parte del código para poder realizar los otros ciclos.

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

Poder terminar el laboratorio.

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

Poder guardar correctamente el autómata puesto que no teníamos muy claro cuales eran las clases que debían extender de Serializable

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

Trabajar en equipo para poder terminar todo correctamente y conocer todo el código que realizo el otro.