LABORATORIO #6: PERSISTENCIA

DESARROLLO

A. Preparando

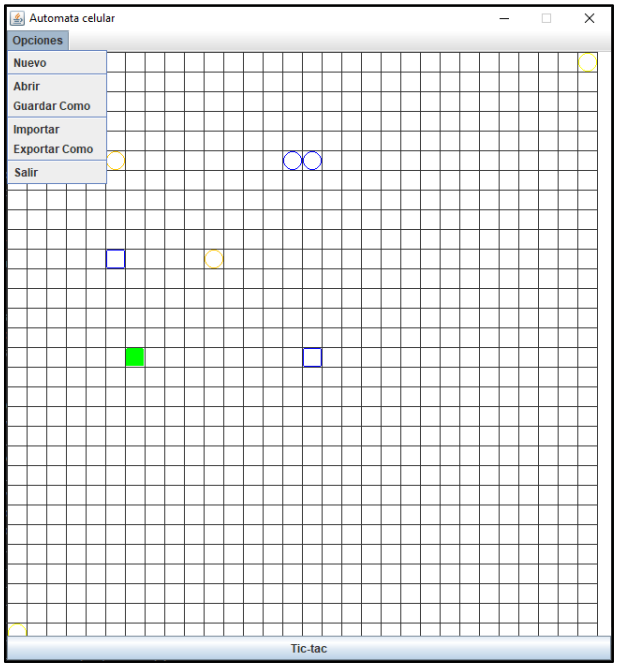
En este laboratorio vamos a extender el proyecto autómata adicionando un menú barra con las opciones básicas de entrada-salida y las opciones estándar nuevo y salir.

1. En su directorio descarguen la versión del proyecto realizado por ustedes para el laboratorio 03 y preparen el ambiente para trabajar desde CONSOLA
2. Ejecuten el programa, revisen la funcionalidad.

B. Creando la maqueta

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.

```
public void prepareAccionesMenu(){
    nuevo.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionNuevo();
        }
    });
    abrir.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionAbrir();
        }
    });
    guardar.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionGuardar();
        }
    });
    importar.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionImportar();
        }
    });
    exportar.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionExportar();
        }
    });
    salir.addActionListener( new ActionListener(){
        public void actionPerformed( ActionEvent event ){
            opcionSalir();
        }
    });
}
```

```
public void opcionNuevo() {
    automata = new CellularAutomata();
}

public void opcionAbrir() {
    automata.abra(archivo);
}

public void opcionGuardar() {
    automata.guarde(archivo);
}

public void opcionImportar() {
    automata.importe(archivo);
}

public void opcionExportar() {
    automata.exporte(archivo);
}

public void opcionSalir() {
    System.exit(0);
}
```

C. Implementando salir y nuevo

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.

```
public void opcionSalir() {
    System.exit(0);
}
```

2. Construyan el método opcionNuevo que crea un nuevo automata. Capturen una pantalla significativa.

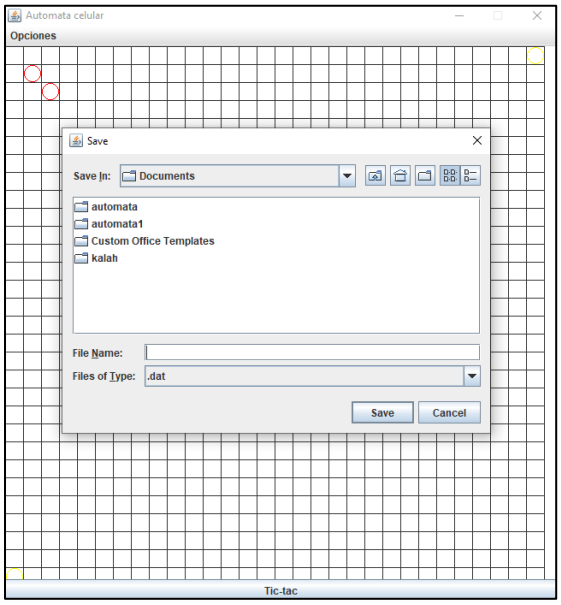
```
public void opcionNuevo() {
    automata = new CellularAutomata();
}
```

D. Implementando salvar y abrir

Las opciones salvar y abrir van a ofrecer servicios de persistencia del 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

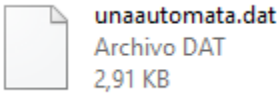
2. Construyan el método opcionGuardar que une de forma adecuada la capa de presentación con la capa de dominio. Usen un FileChooser y atiendan la excepción. Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una pantalla significativa.



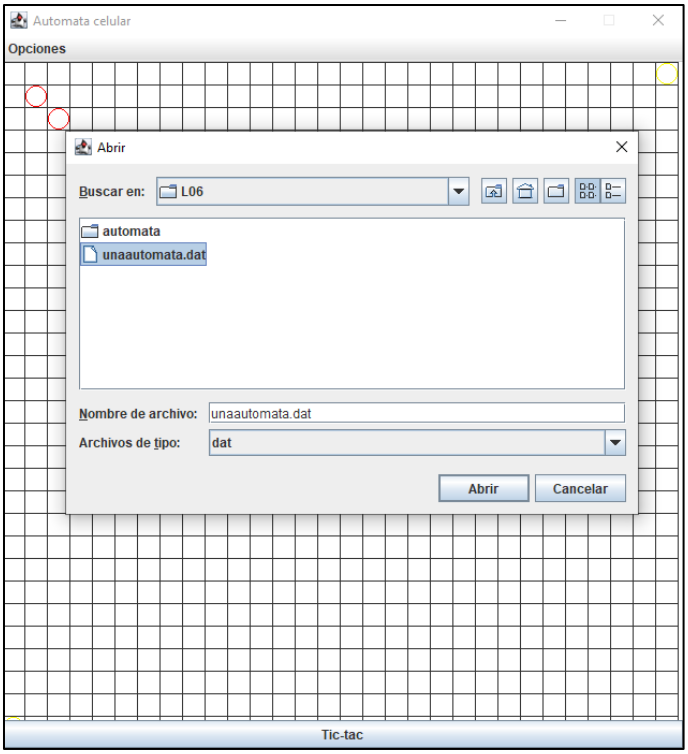
3. Construyan el método guarde que ofrece el servicio de guardar en un archivo el estado actual del autómata.

```
public void guarde(File archivo) throws AutomataException{
    try {
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(archivo));
        out.writeObject(this);
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

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?



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 autómata de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general.

```
public static CellularAutomata abra(File archivo) throws AutomataException{
    CellularAutomata automata = null;
    try {
        ObjectInputStream in = new ObjectInputStream(new FileInputStream(archivo));
        automata = (CellularAutomata) in.readObject();
        in.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return automata;
}
```

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

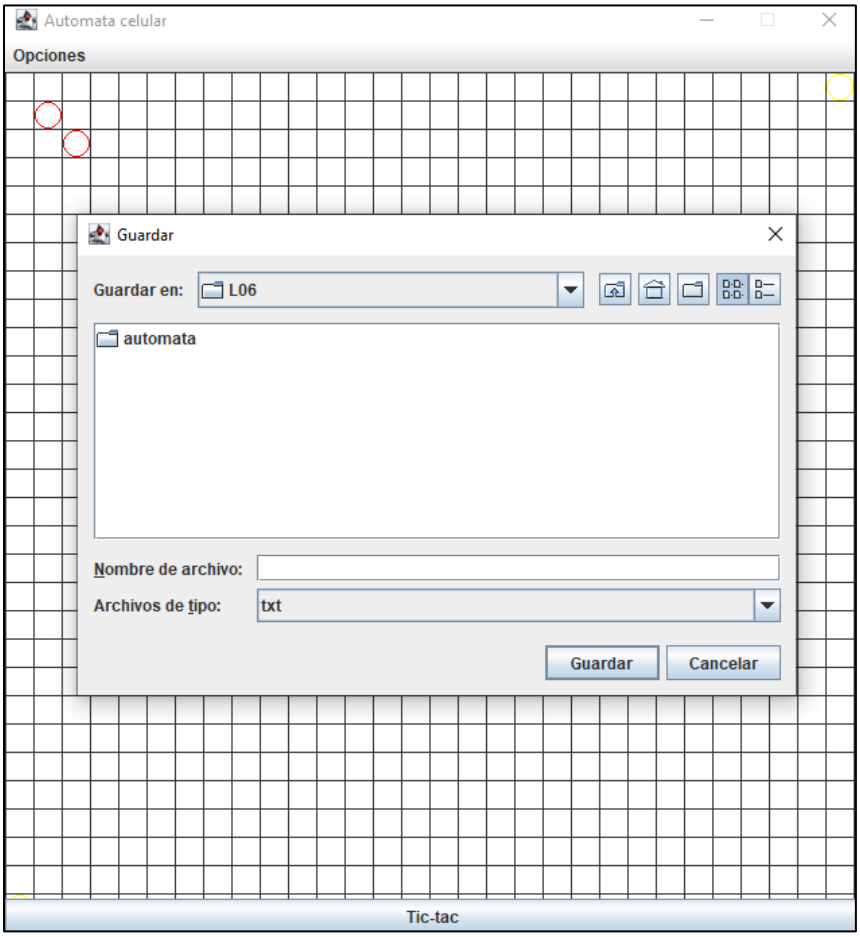
E. Implementando importar y exportar

Estas operaciones nos van a permitir importar información del autómata desde un archivo de texto y exportarlo. Los nombres de los archivos de texto deben tener como extensión .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.
Inquieta 100 100
Bombillo 500 500

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.

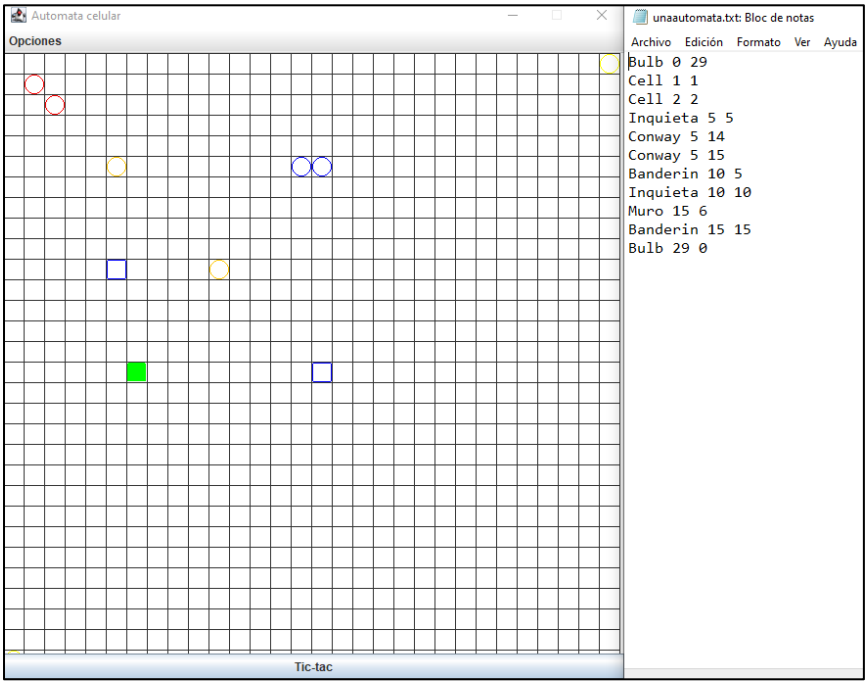


3. Construyan el método exporte que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado actual.

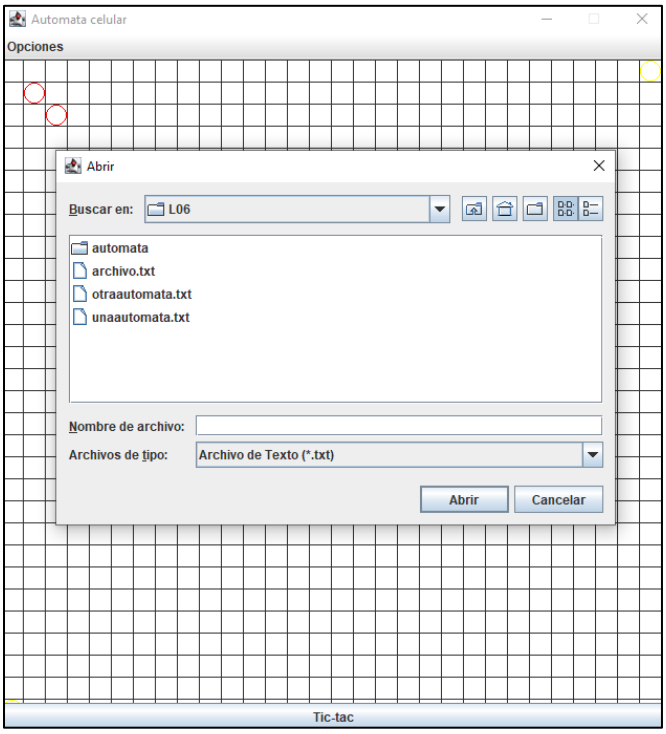
```
public void exporte(File archivo) throws AutomataException{
    try{
        FileWriter out = new FileWriter(archivo);

        for(int i=0; i<automata.length;i++){
            for(int j=0;j<automata.length;j++){
                if(getItem(i,j)!=null){
                    out.write(getItem(i,j).getClass().toString().replace("class","").replace(" domain.","")+
                            " "+String.valueOf(i)+" "+String.valueOf(j)+"\n");
                }
            }
        }
        out.close();
    }
    catch(Exception e){
        e.printStackTrace();
    }
}
```

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?



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.

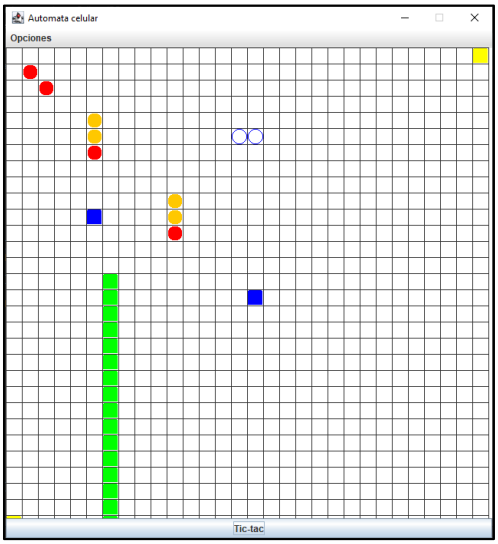


6. Construyan el método importe que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considere un mensaje de error general. (Consulten en la clase String los métodos trim y split)

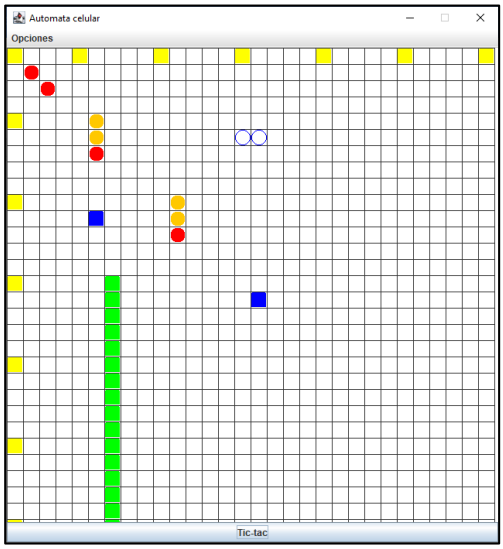
```
public static CellularAutomata importe(File archivo) throws AutomataException{
    CellularAutomata automata = new CellularAutomata();
    try {
        BufferedReader bIn = new BufferedReader(new FileReader(archivo));
        String line = bIn.readLine();
        while (line != null) {
            line = line.trim();
            String[] data = line.split(" ");
            construccion(automata, data[0], data[1], data[2]);
            line = bIn.readLine();
        }
        bIn.close();
    } catch (Exception e) {
        e.getStackTrace();
    }
    return automata;
}
```

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.
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

Antes de importar:



Después de importar:



F. Analizando comportamiento

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

   Después de hacer 3 Tic-tac, de guardarlo y abrirlo de nuevo, se puede observar el correcto funcionamiento de esta opción. El autómata abierto es igual al guardado.

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

   Se ejecuta y se exporta en un archivo txt el estado del autómata después de dar tres Tic-tac. Al importar dicho archivo, son las mismas células, pero no se guarda el estado en el que quedan (vivas, muertas o desconocido) sino que se reinician.

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

   La diferencia es que, al guardar y abrir, el estado de las células y su posición se mantiene y al exportar e importar, como se indicó anteriormente, se mantiene la posición, pero el estado cambia.

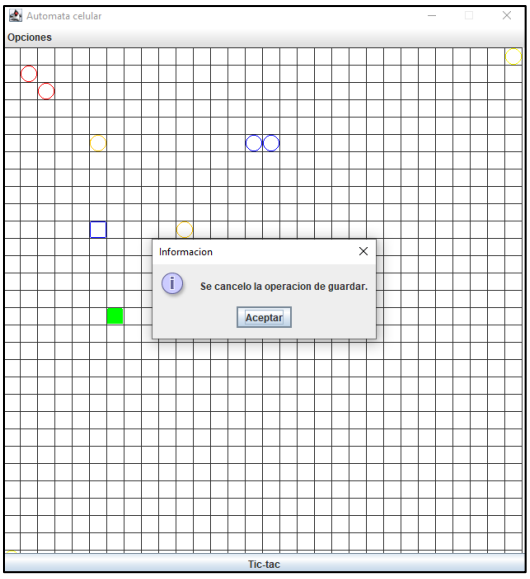
G. Perfeccionando salvar y abrir

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.

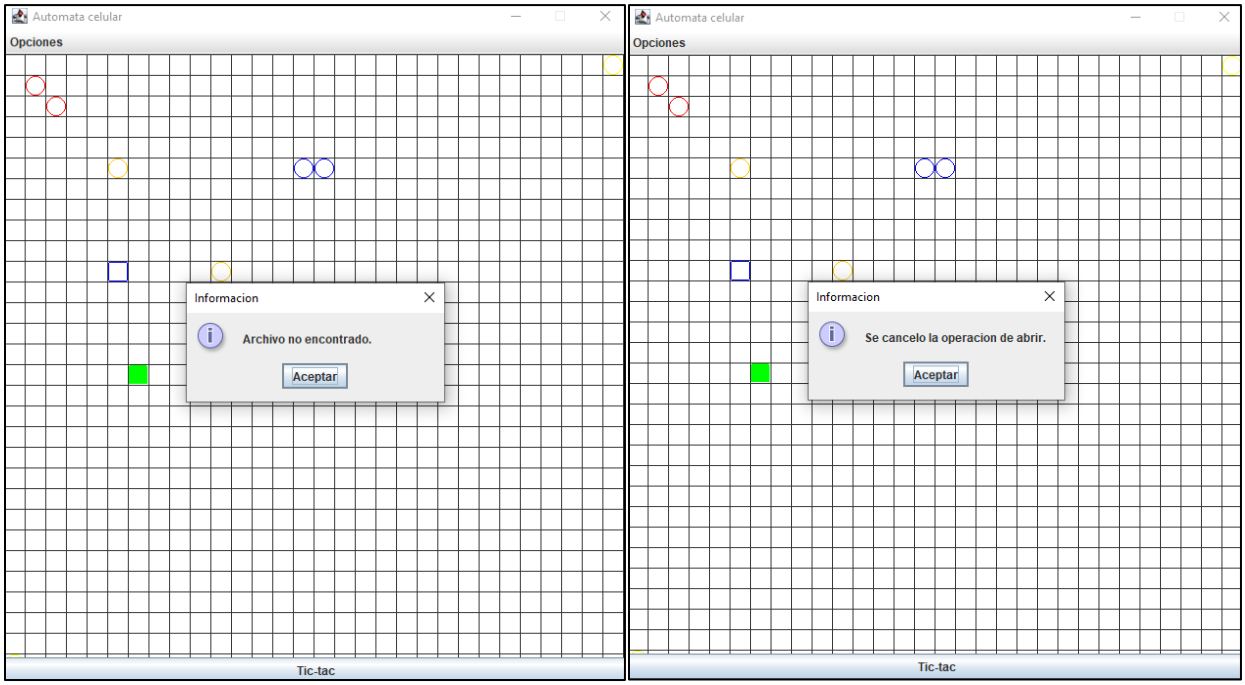
   Se tienen en cuenta los casos en el que el archivo no existe y se cancela la operación de abrir/guardar

3. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

   GUARDAR



ABRIR



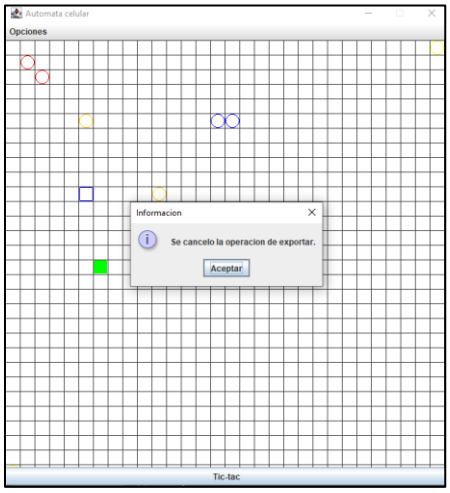
H. Perfeccionando importar y exportar.

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.

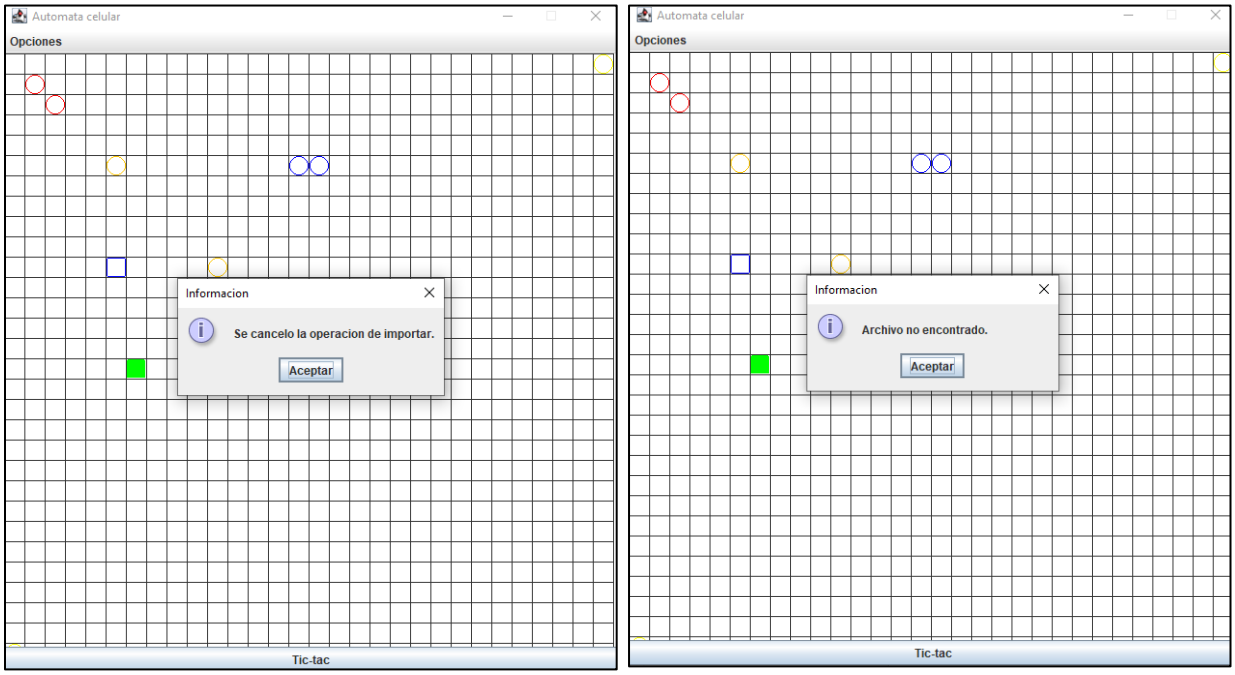
   Se tienen en cuenta los casos en el que el archivo no existe y se cancela la operación de abrir/guardar

3. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

EXPORTAR



IMPORTAR



I. Perfeccionando importar. Hacia un minicompilador.

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.

J. Perfeccionando importar. Hacia un minicompilador flexible.

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/Nombre)

   (12/Angie Natalia Mojica)
   (12/Daniel Antonio Santanilla)

2. ¿Cuál es el estado actual del laboratorio? ¿por qué? **(⊙_⊙;)**

   Se logró concretar hasta el item H, quedó faltando el último punto, pues por temas de tiempo y debido a que se debe continuar con el desarrollo del proyecto.

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

   Una vez más aplicamos en su mayoría de tiempo, la practica programación a pares en la que íbamos discutiendo la mejor forma de abarcar cada ciclo.

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

   El mayor logro fue hacer los métodos de persistencia y que funcionen.

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

   Hacer los minicompiladores dado que no sabíamos en que línea se podía presentar el error y como lo dijimos, se destinó más tiempo para el desarrollo del proyecto.

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

   Se destino tiempo para el desarrollo de este laboratorio, aportando cada uno ideas para el desarrollo del laboratorio.

   Siendo este el último laboratorio, nos dimos cuenta de que fuimos cumpliendo poco a poco los compromisos que hacíamos y los resultados fueron mejorando.

(☞ﾟヮﾟ)☞ FINAL ☜(ﾟヮﾟ☜)