**Laboratorio Nro. 6**

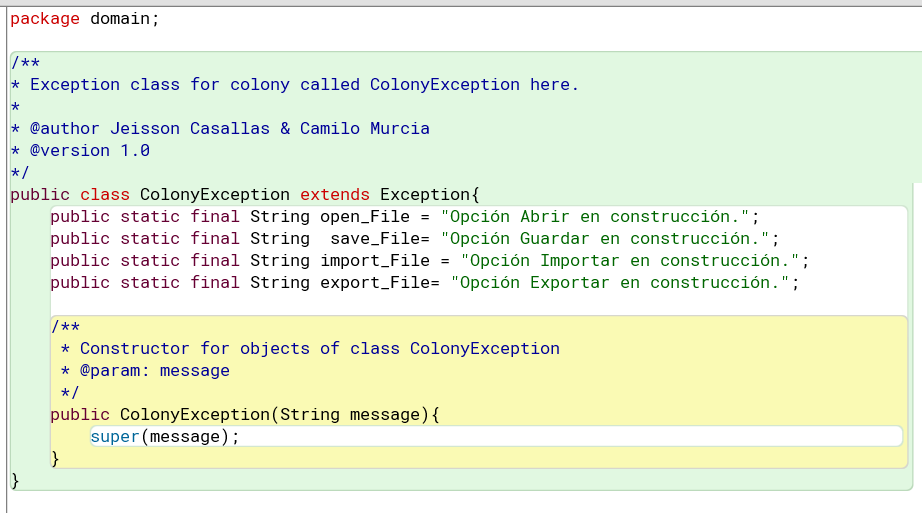
**Jeisson Casallas y Camilo Murcia**

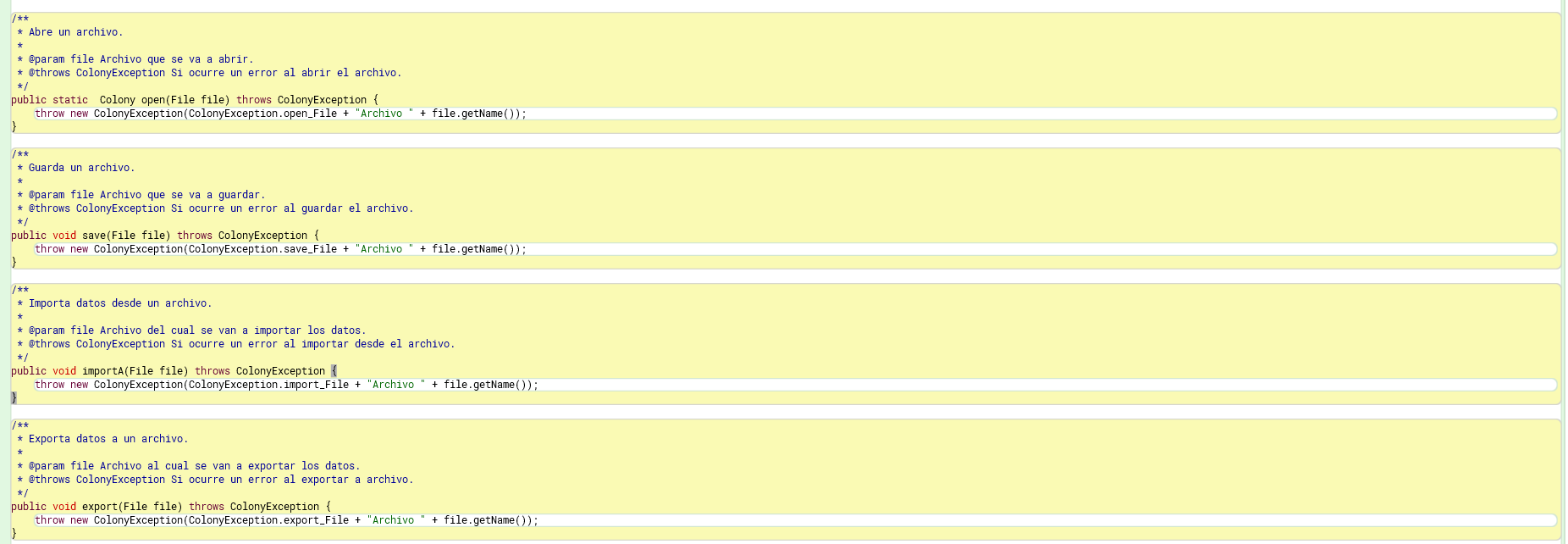
Creando la maqueta [En lab06.doc, \*.astah y \*.java]

[NO OLVIDEN BDD y MDD]

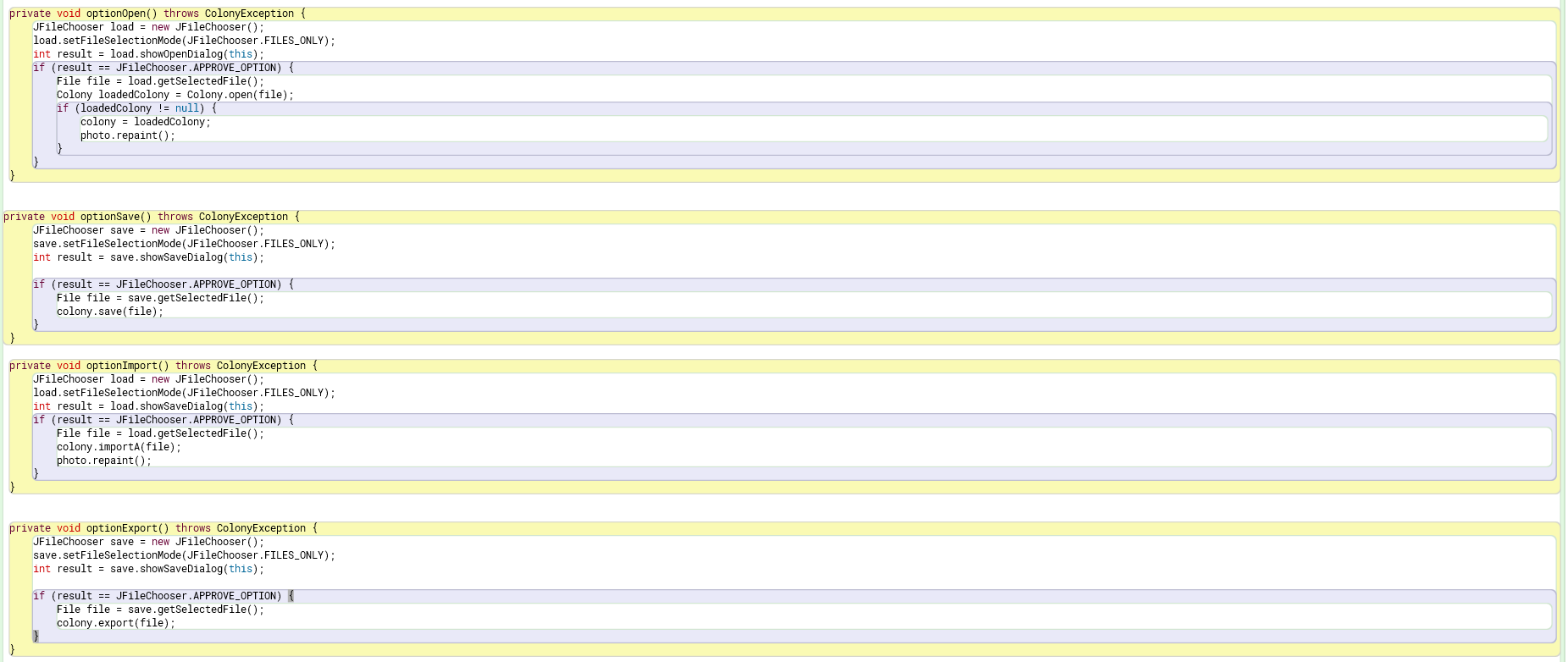
En este punto vamos a construir la maqueta correspondiente a esta extensión siguiendo el patrón MVC.

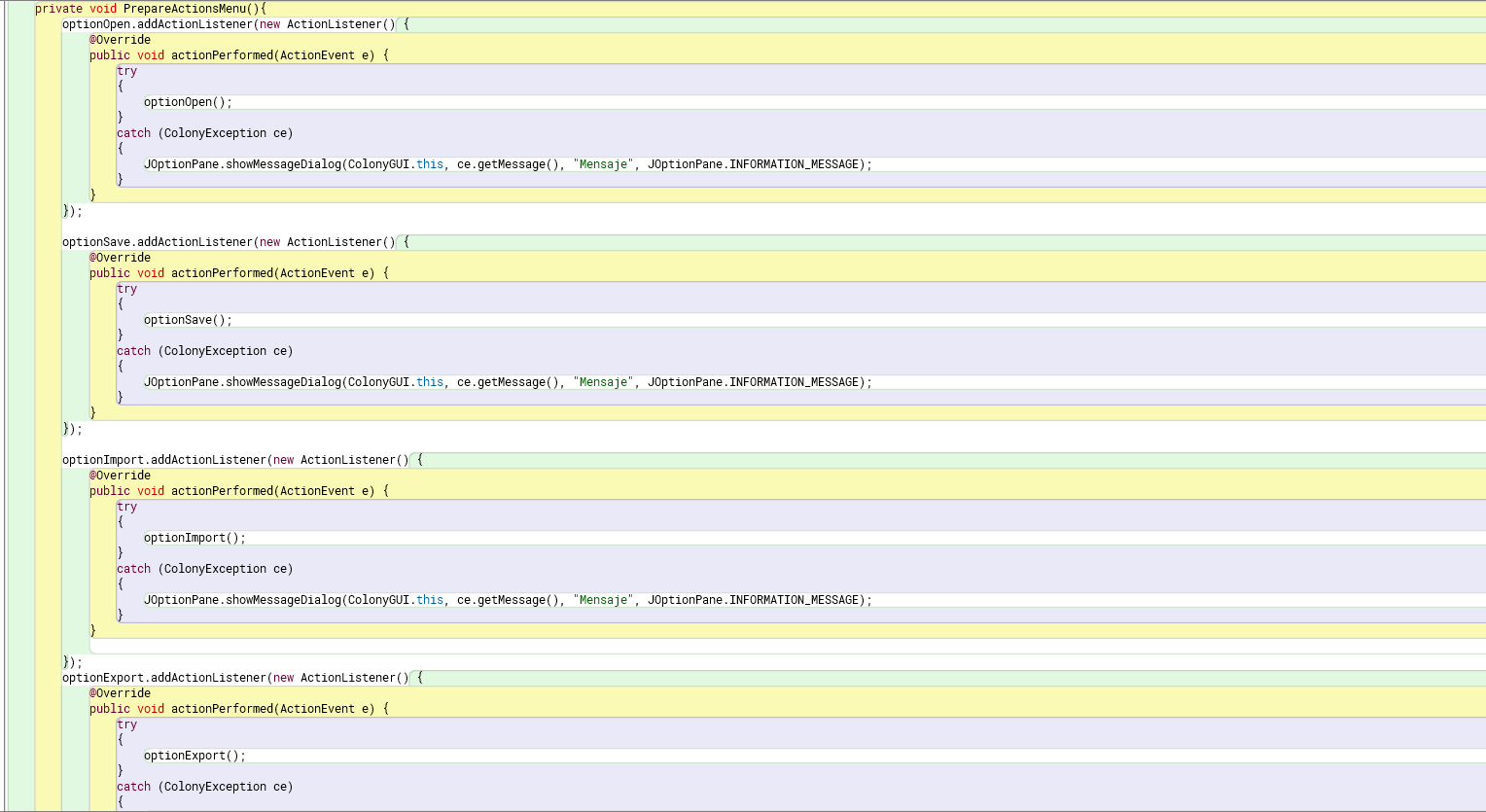
1. MODELO: Preparen en la clase fachada del modelo los métodos correspondientes a las cuatro optiones básicas de entrada-salida (open, save, import y export). Los métodos deben simplemente propagar una colonyException con el mensaje de “Opción nombreOpción en construcción. Archivo nombreArchivo”. Los métodos deben tener un parámetro File.



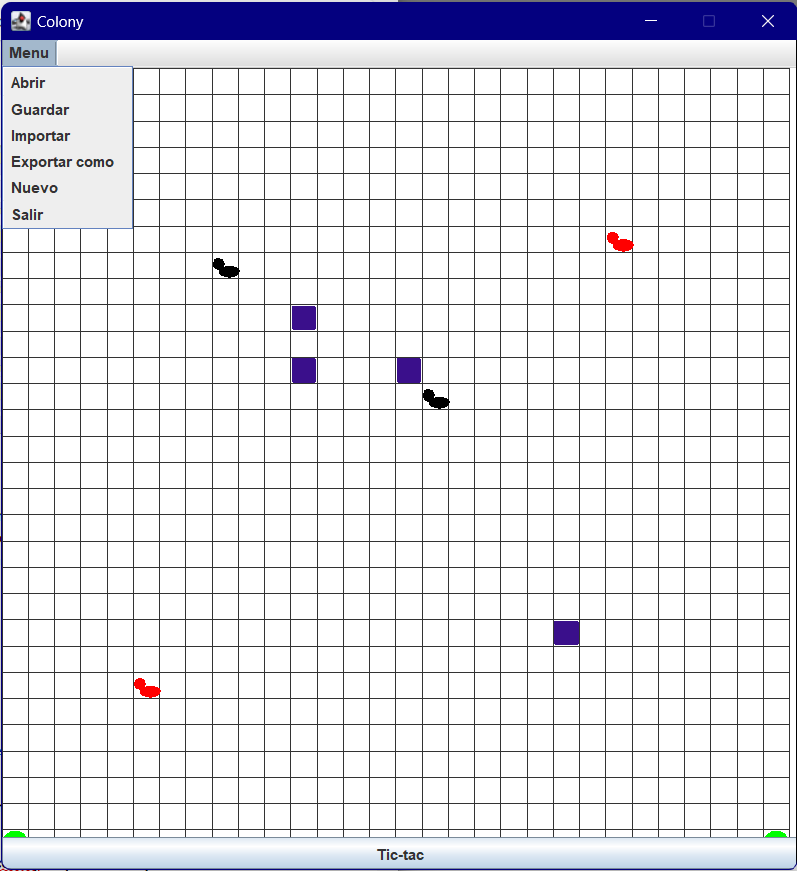


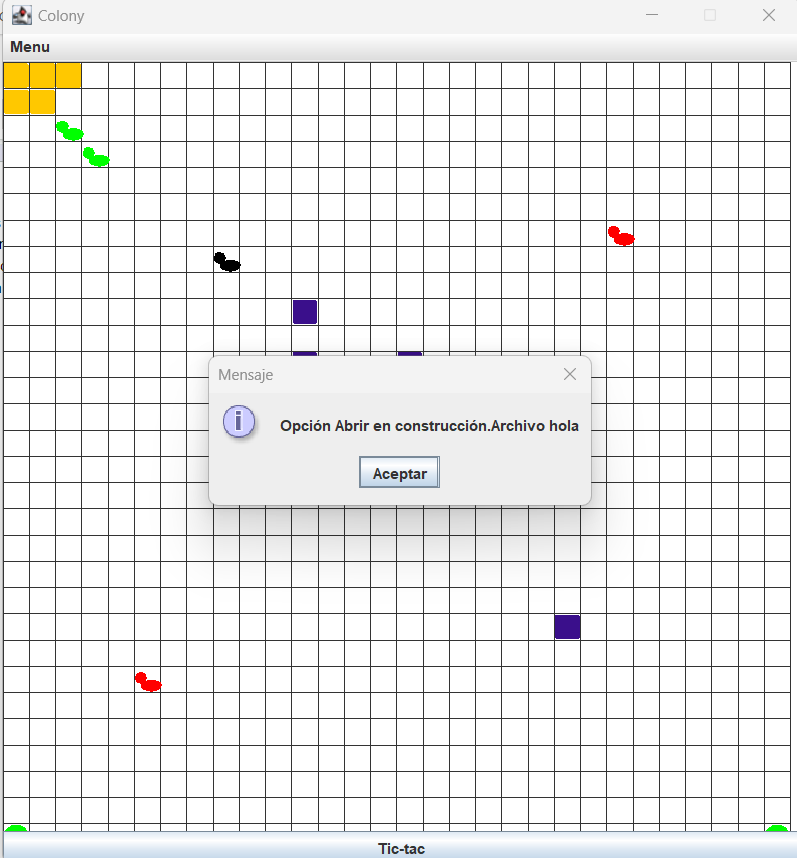
2. CONTROLADOR: Construyan los oyentes correspondientes a las seis optiones. Para esto creen el método prepareAccionesMenu y los métodos base del controlador (optionOpen, optionSave, optionImport, optionExport, optionNew, optionExit), En las optiones que lo requieran usen un FileChooser y atiendan la excepción. Estos métodos llaman el método correspondiente de la capa de dominio.

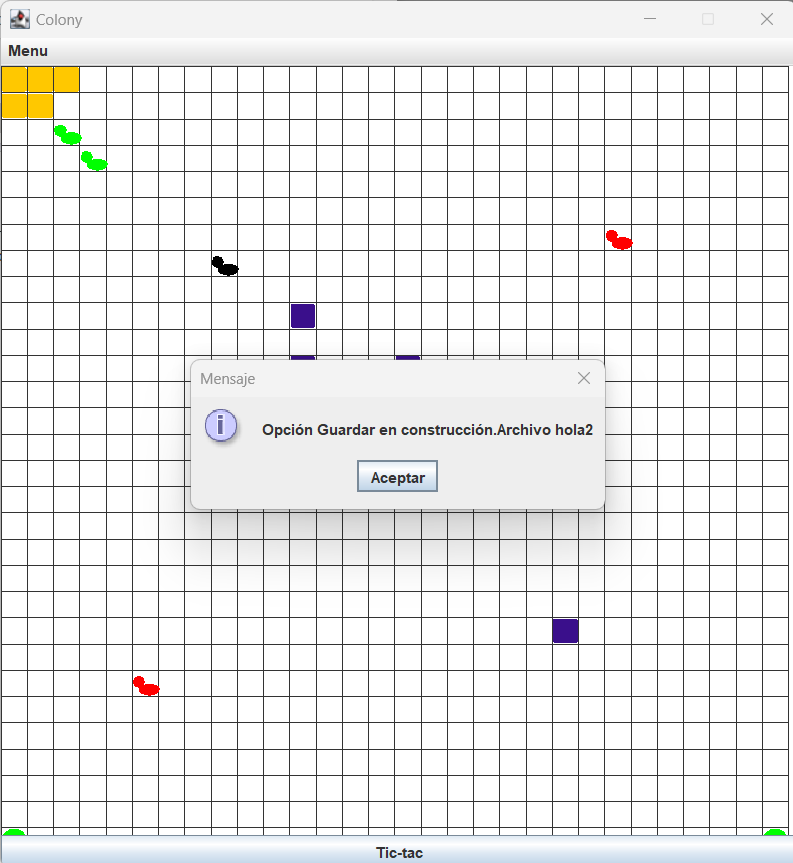


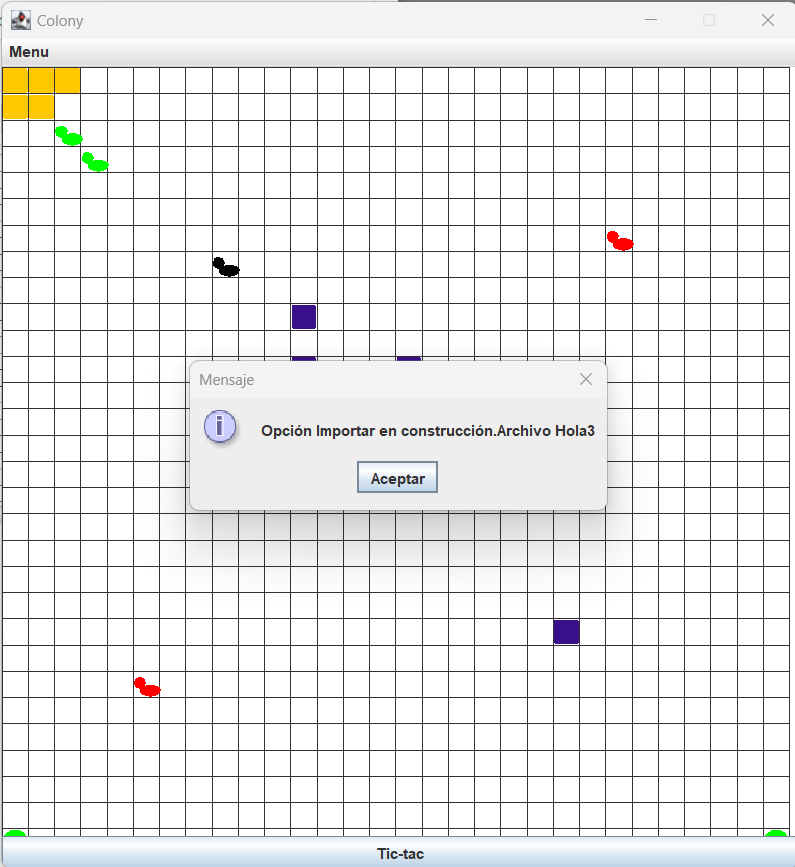


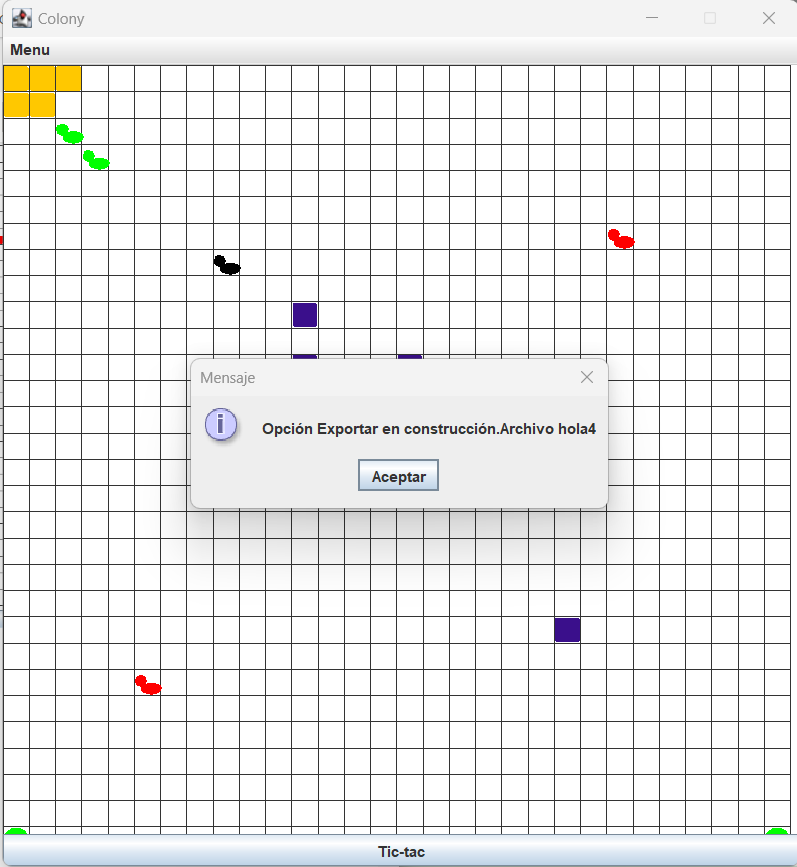
3. VISTA : Construyan un menú barra que ofrezca, además de las optiones básicas de entrada-salida, las optiones estándar de nuevo y salir (Nuevo, Abrir, Guardar como, Importar, Exportar como, Salir). No olviden incluir los separadores. Para esto creen el método prepareElementsMenu. Ejecuten las diferentes acciones del menú y para cada una de ellas capture una pantalla significativa.











**Implementando salir y nuevo**

**[En lab06.doc, \*.asta y \*.java]**

**[NO OLVIDEN BDD y MDD]**

Las optiones 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 optionExit que hace que se termine la aplicación. No es necesario incluir confirmación.**

**2. Construyan el método optionNew que crea un nuevo colony. Capturen una pantalla significativa.**



* Visualmente ya se ve desde el .jar.

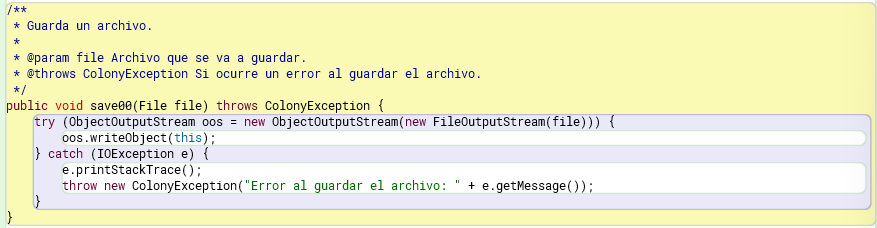
**Implementando salvar y abrir**

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

Las optiones salvar y abrir van a ofrecer servicios de persistencia del colony como objeto. Los nombres de los archivos deben tener como extensión .dat.

**1. Copien las versiones actuales de open y save y renómbrenlos como open00 y save00**

**2. Construyan el método save que ofrece el servicio de guardar en un archivo el estado actual de colony. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.**



* Pruebas en la carpeta de ColonyTest y diseño en el astah actualizado.

**3. Validen este método guardando el estado obtenido después de dos clics como oneColony.dat. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?**

* Efectivamente el archivo se creó en el Disco y ocupa 3KB.

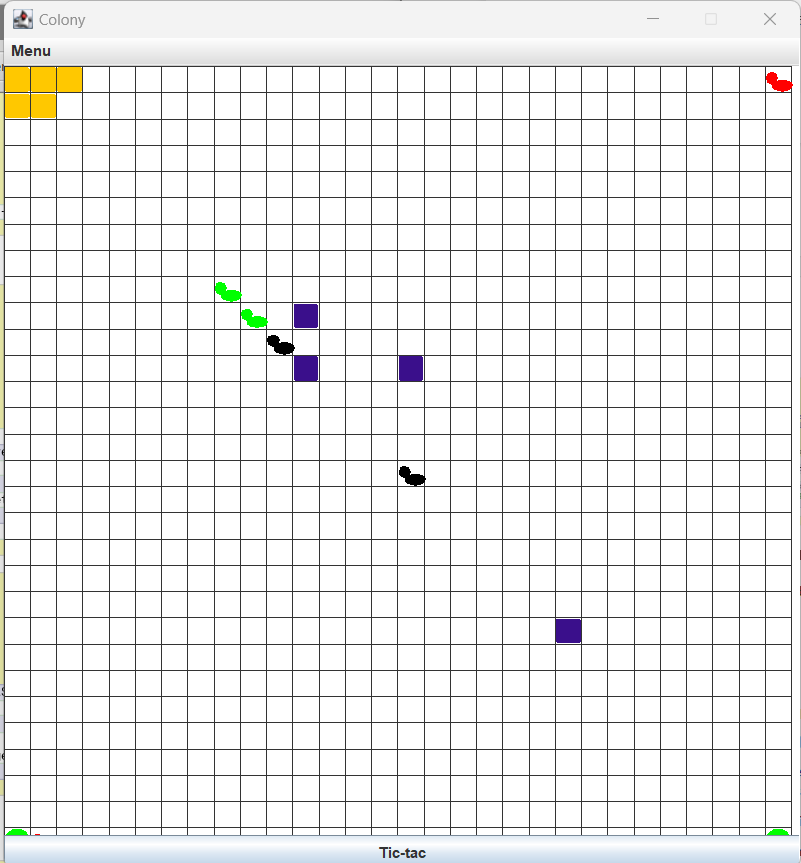


**4. Construyan el método open que ofrece el servicio de leer un colony de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.**

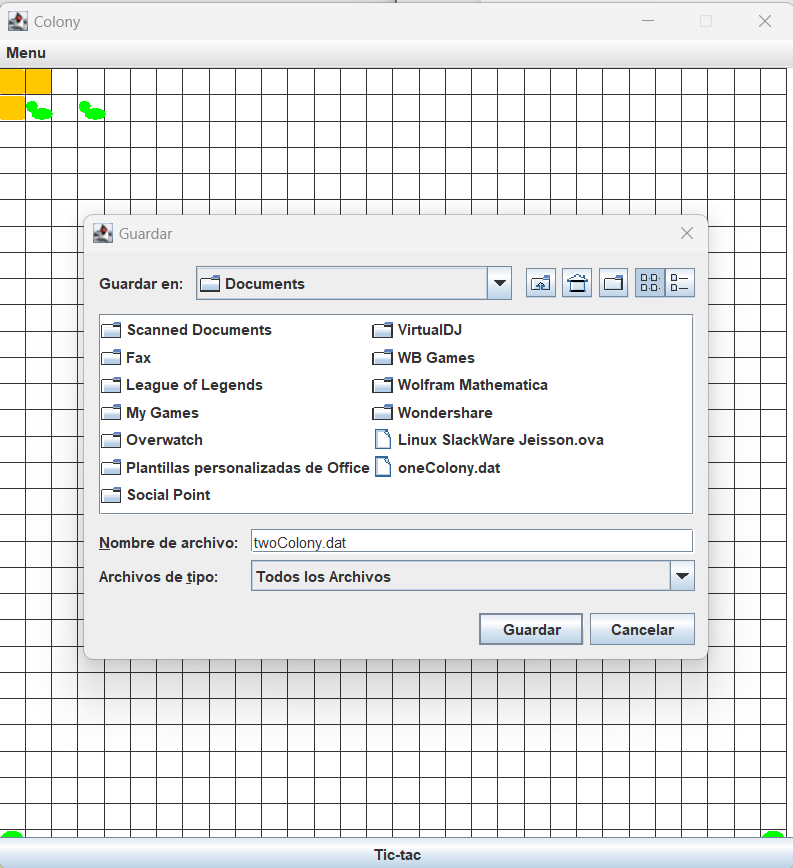


* Pruebas en la carpeta de ColonyTest y diseño en el astah actualizado.

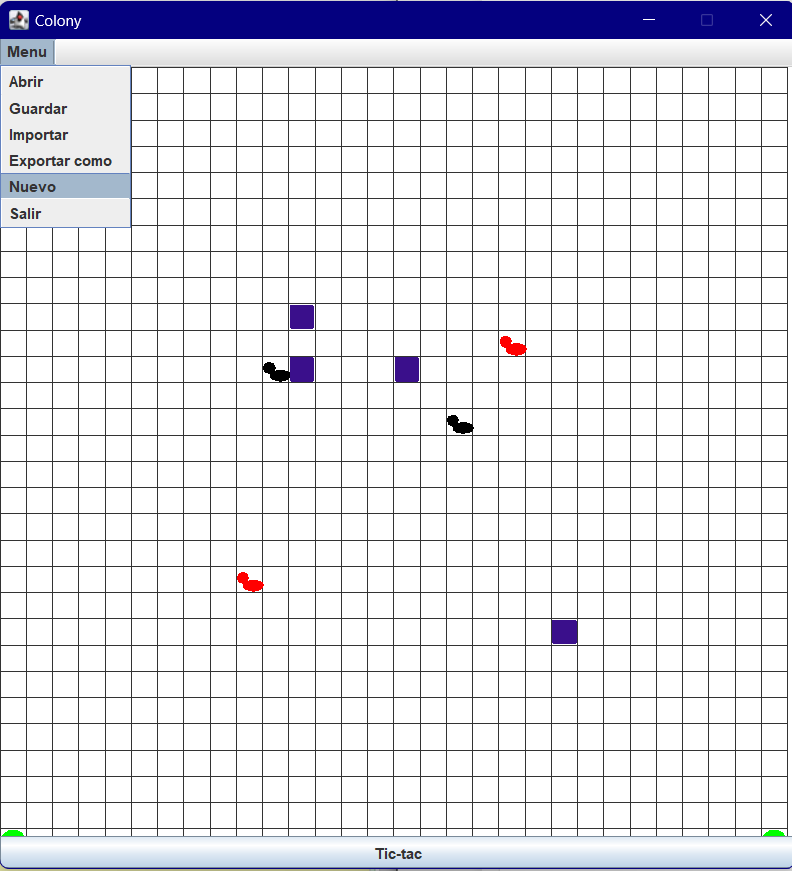
**5. Realicen una prueba de aceptación para este método iniciando la aplicación, creando un nueva situación en el colony y abriendo el archivo oneColony.dat. Capturen imágenes significativas de estos resultados.**

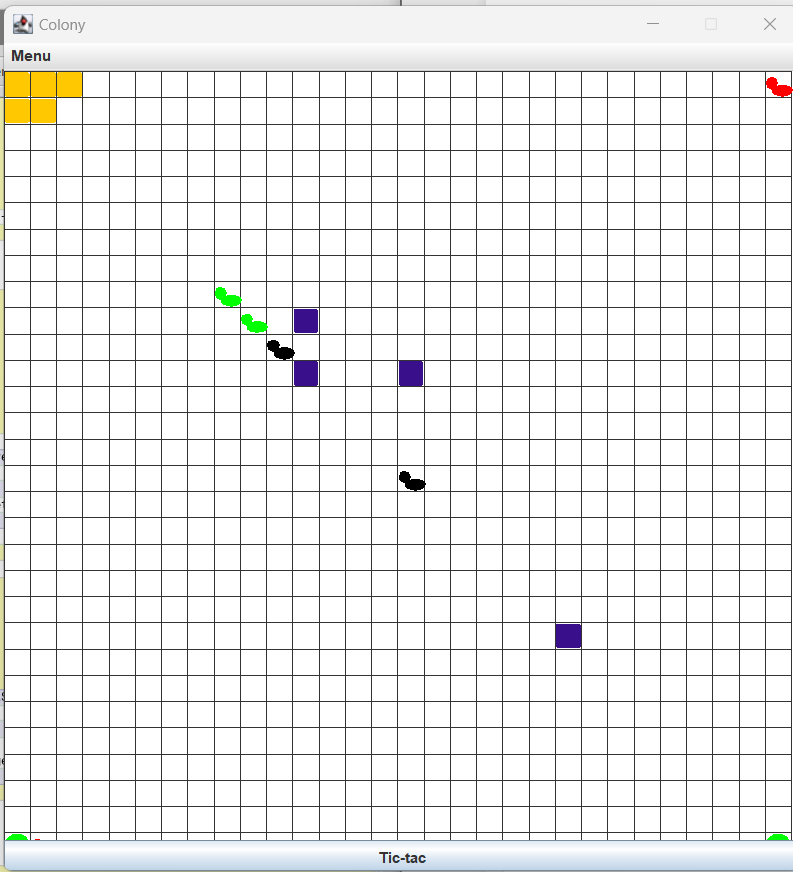


Tenemos el tablero inicial, procedemos a hacer 10 Tic-tac y luego a guardar.

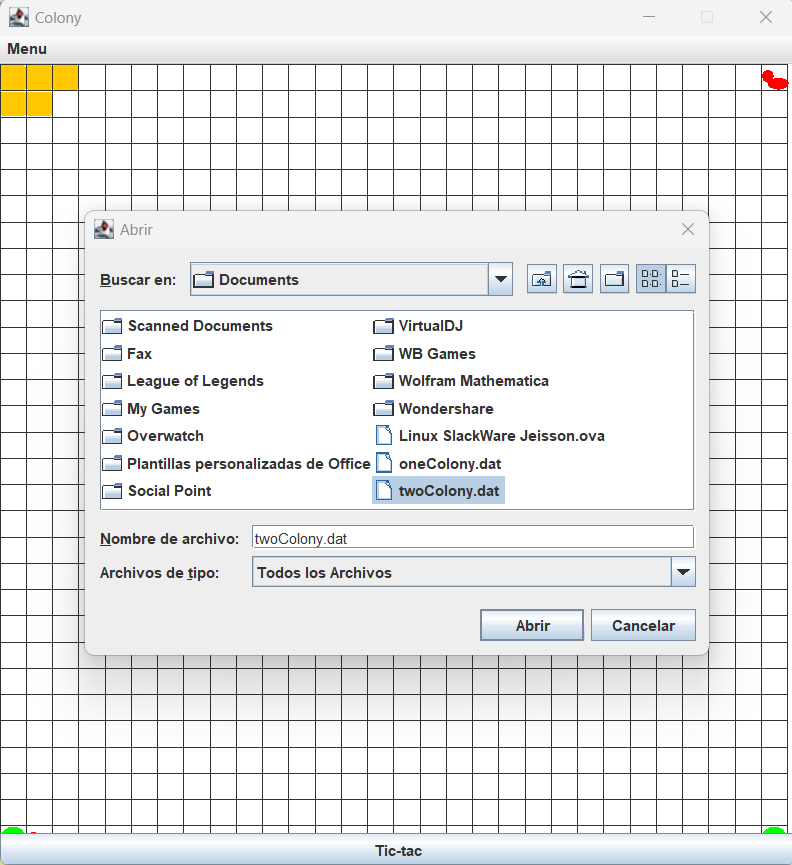


Ahora le damos a nuevo

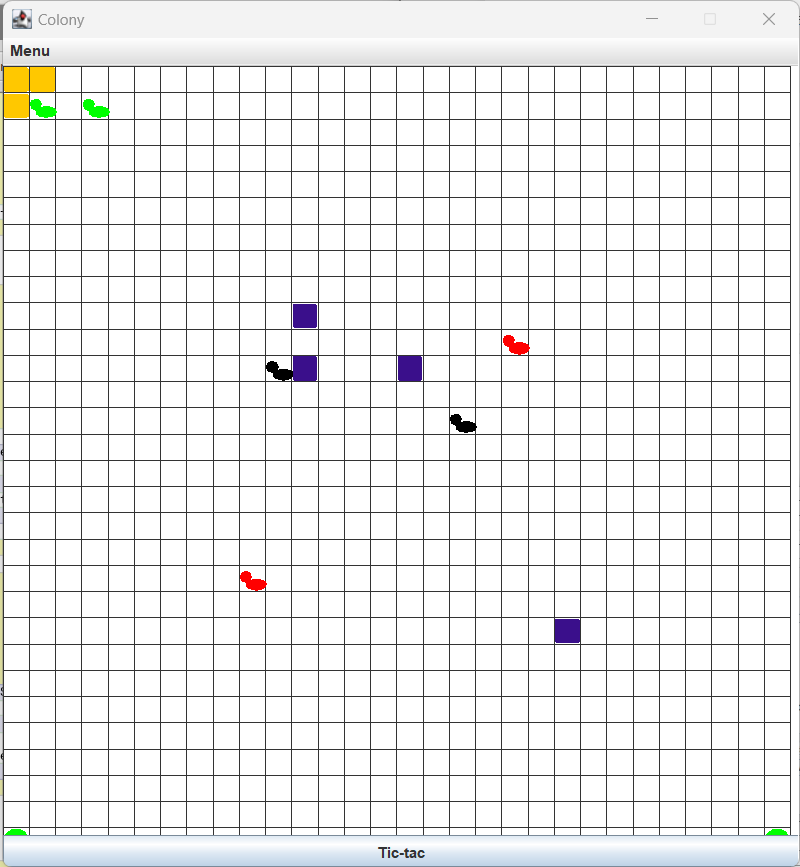




Abrir el twoColony.dat



Volvemos a la pantalla con los 10 tic’s



**Implementando importar y exportar**

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

Estas operaciones nos van a permitir importar información del colony 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.

Ant 100 100

Flower 500 500

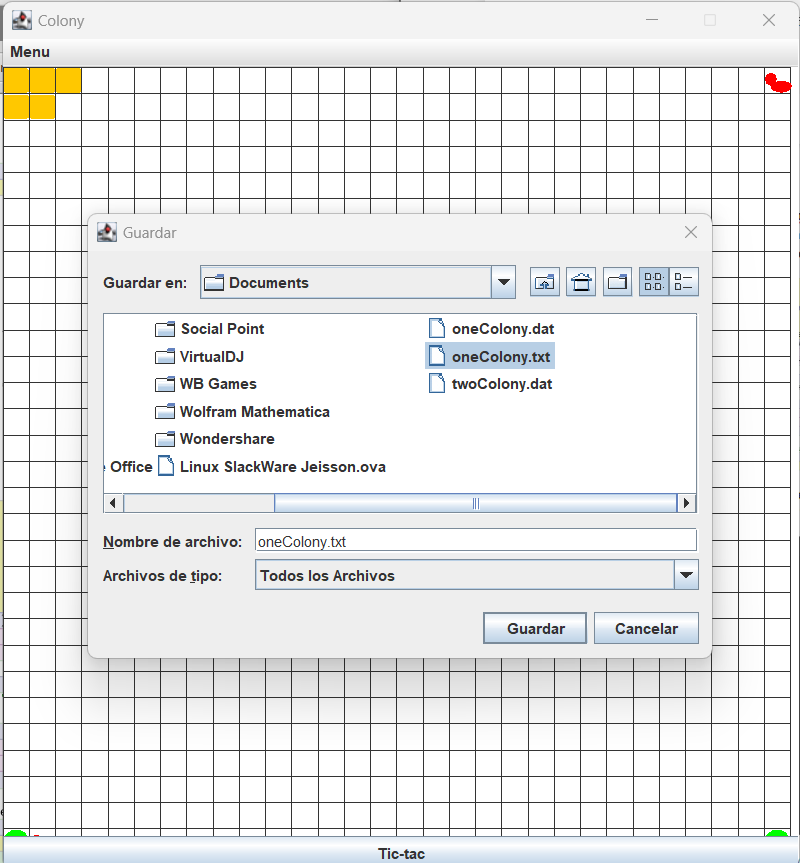
**1. Copien las versiones actuales de import y export y renómbrenlos como import00 y export00**

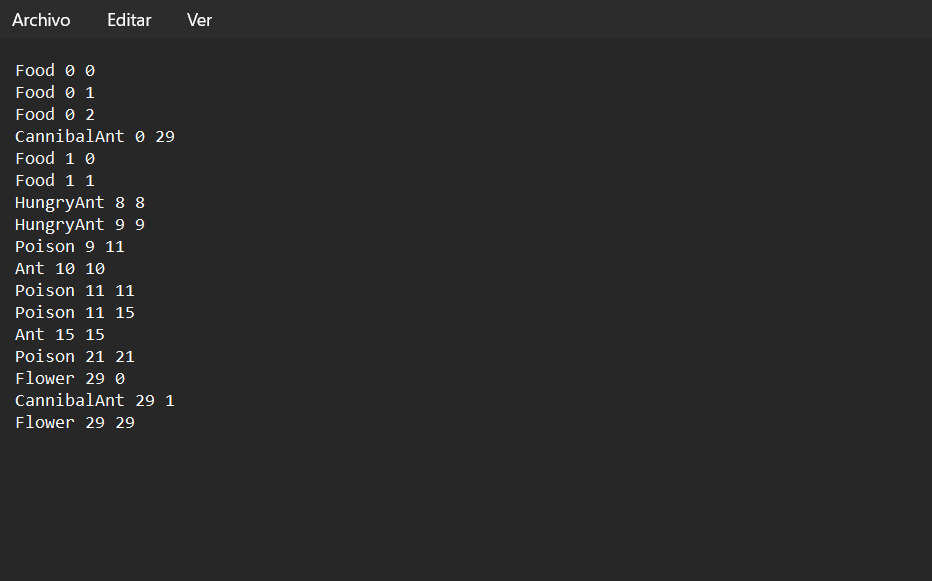
**2. Construyan el método export que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado actual. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.**



* Pruebas en la carpeta de ColonyTest y diseño en el astah actualizado.

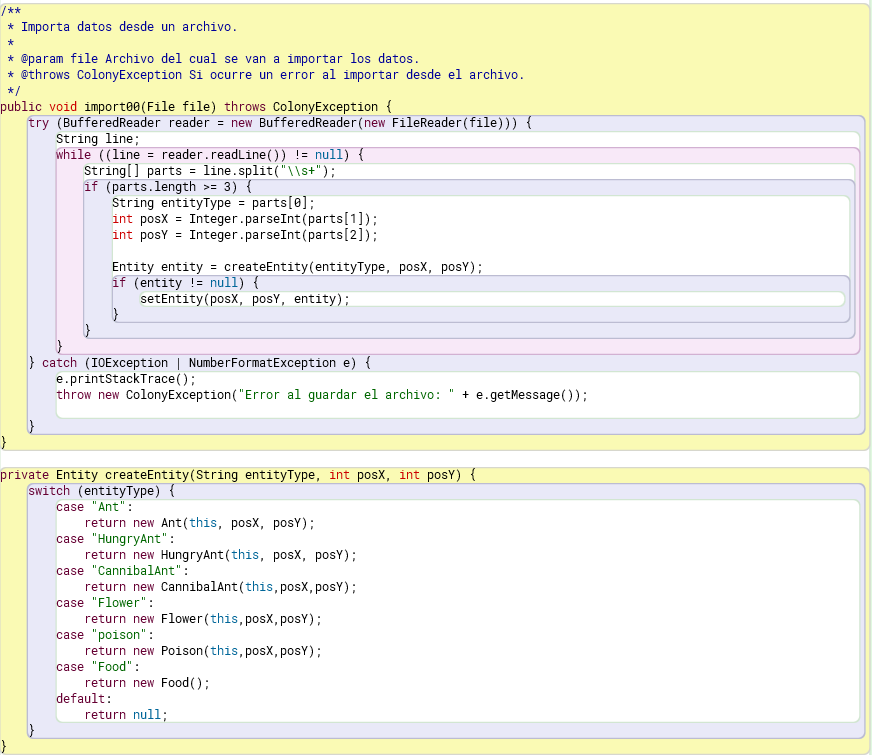
**3. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como oneColony.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?**





Se puede ver como se guarda en un archivo .txt todas las entidades que tenemos en el tablero junto con sus coordenadas en X y en Y.

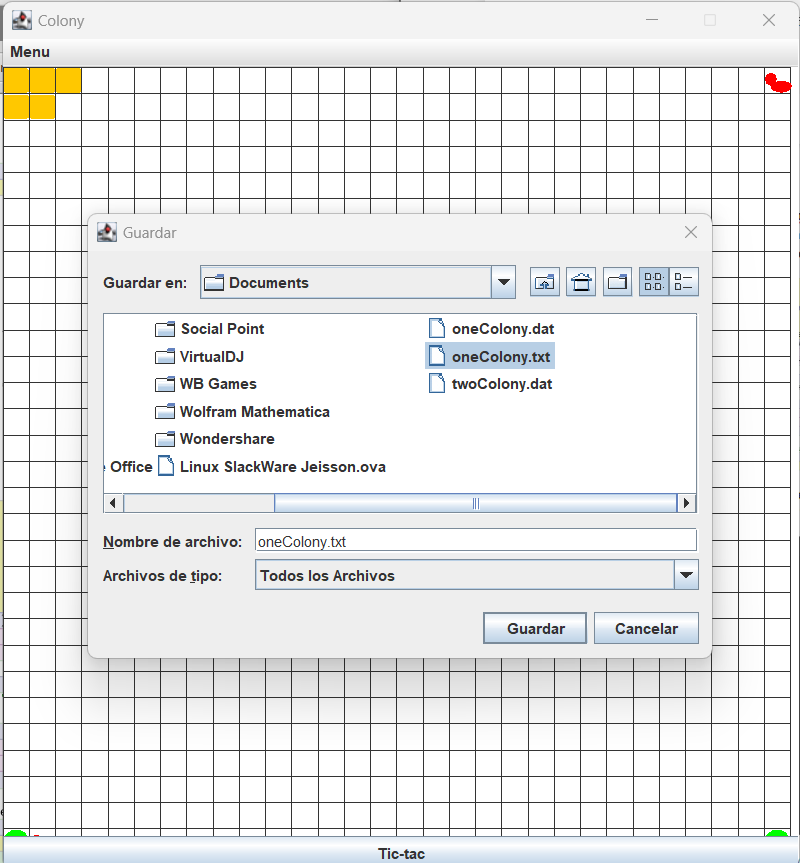
**4. Construyan el método import que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considere un mensaje de error general. No olviden diseño y pruebas de unidad. (Consulten en la clase String los métodos trim y split)**

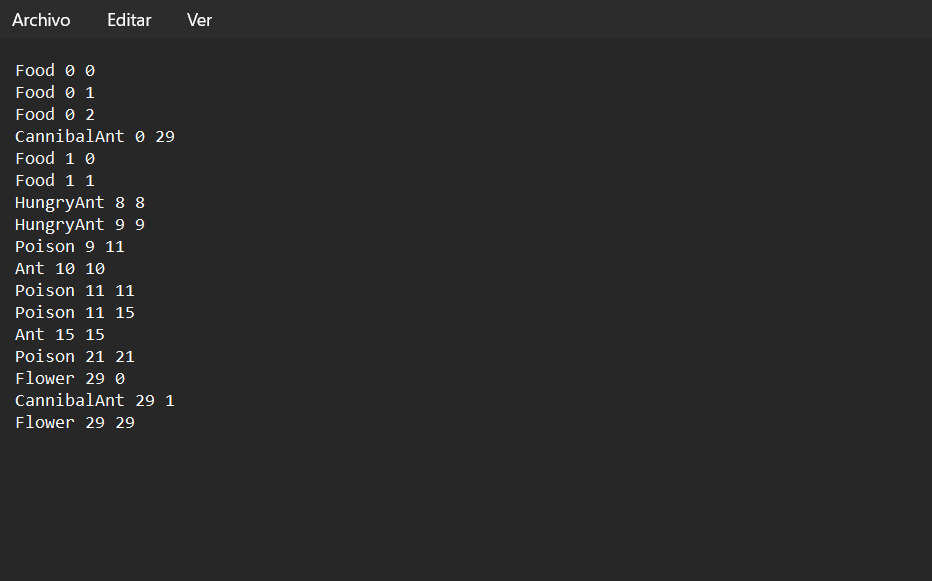


* Pruebas en la carpeta de ColonyTest y diseño en el astah actualizado.

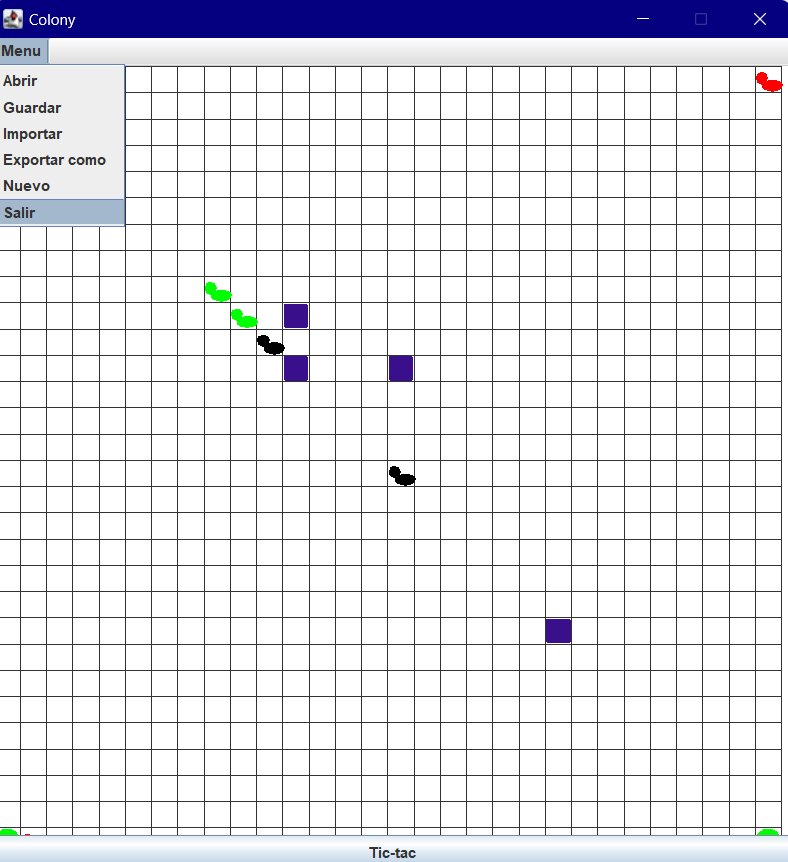
**5. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a oneColony.txt. saliendo, entrando, creando una nueva e importando el archivo otherColony.txt. ¿Qué resultado obtuvieron? Capturen la pantalla final.**

* Exportamos el Archivo OneColony.txt

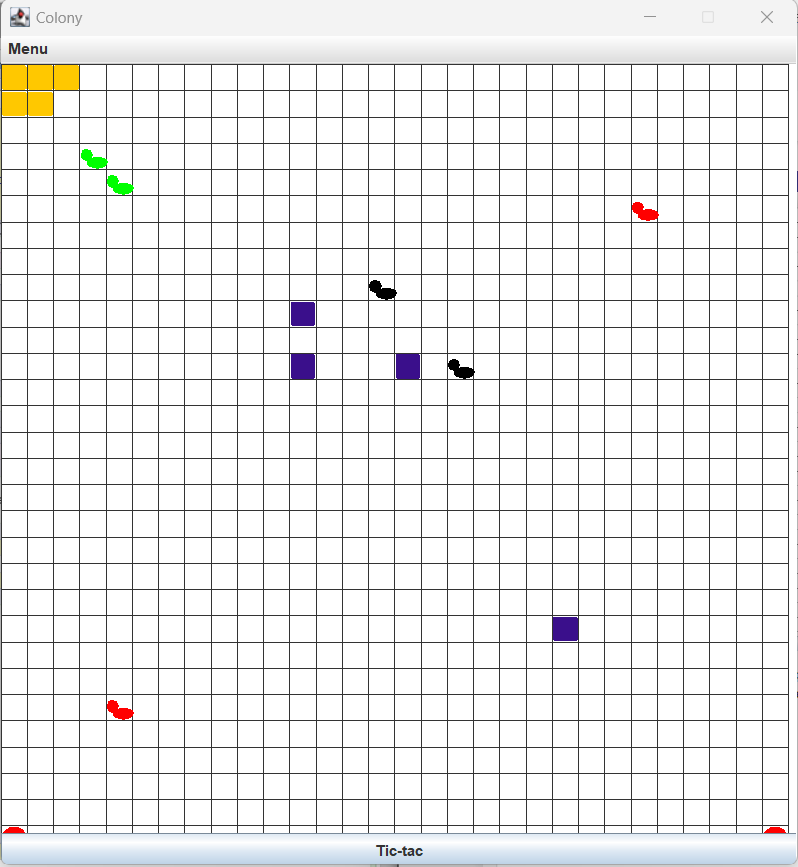




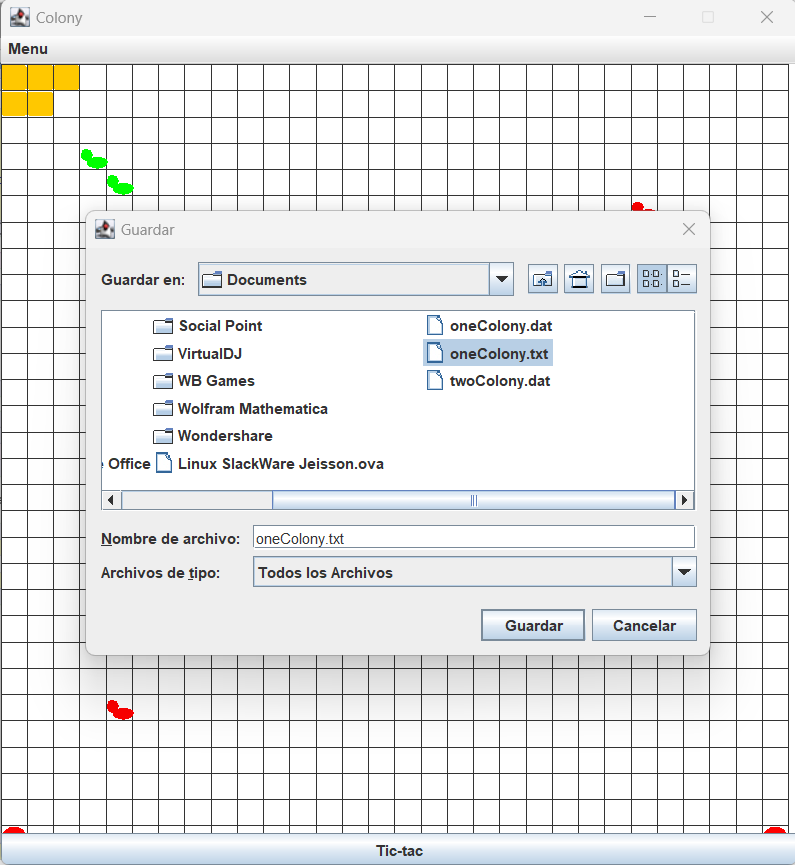
* Cerramos la aplicación



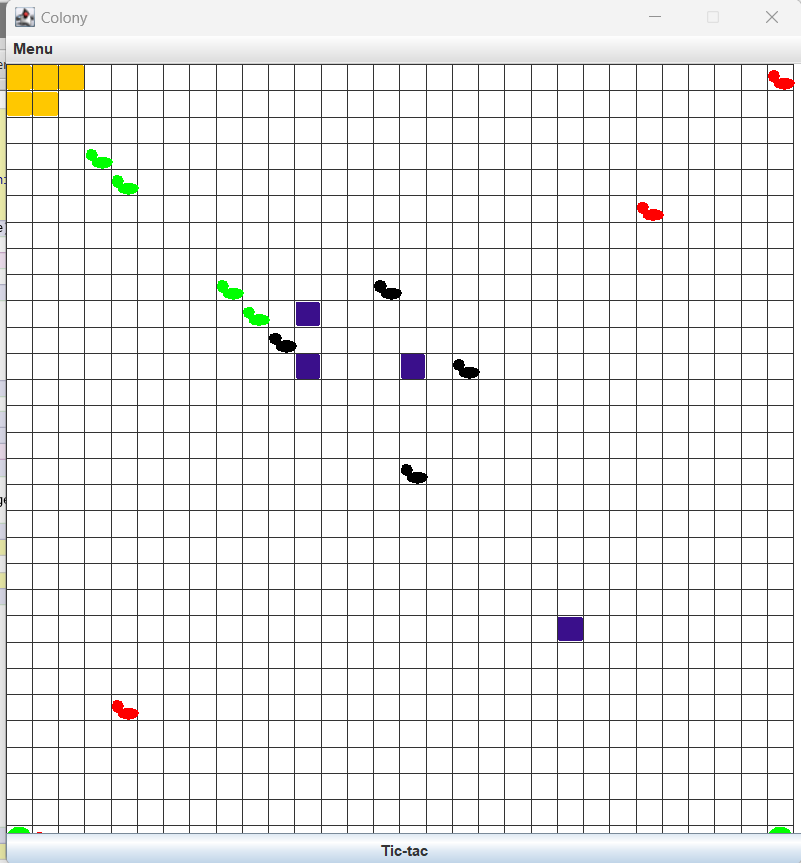
* La volvemos a abrir y hacemos un par de Tic’s para notar los cambios.



* Importamos el OneColony.txt

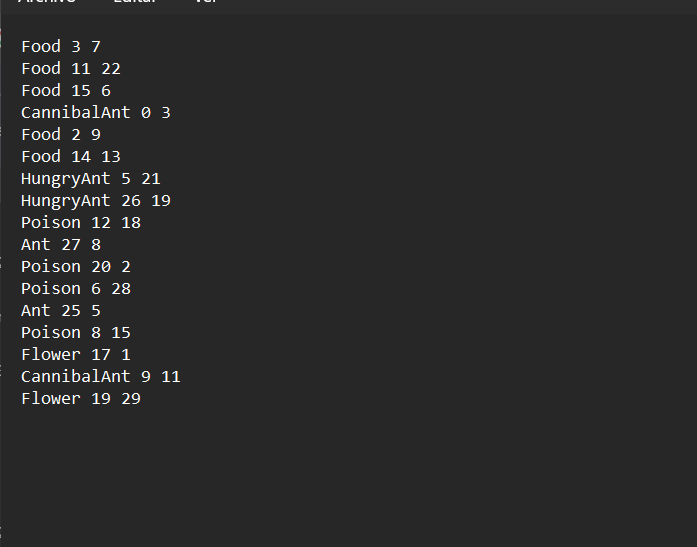


* Vemos que en las posiciones dadas en él .txt se han creado nuevas entidades.

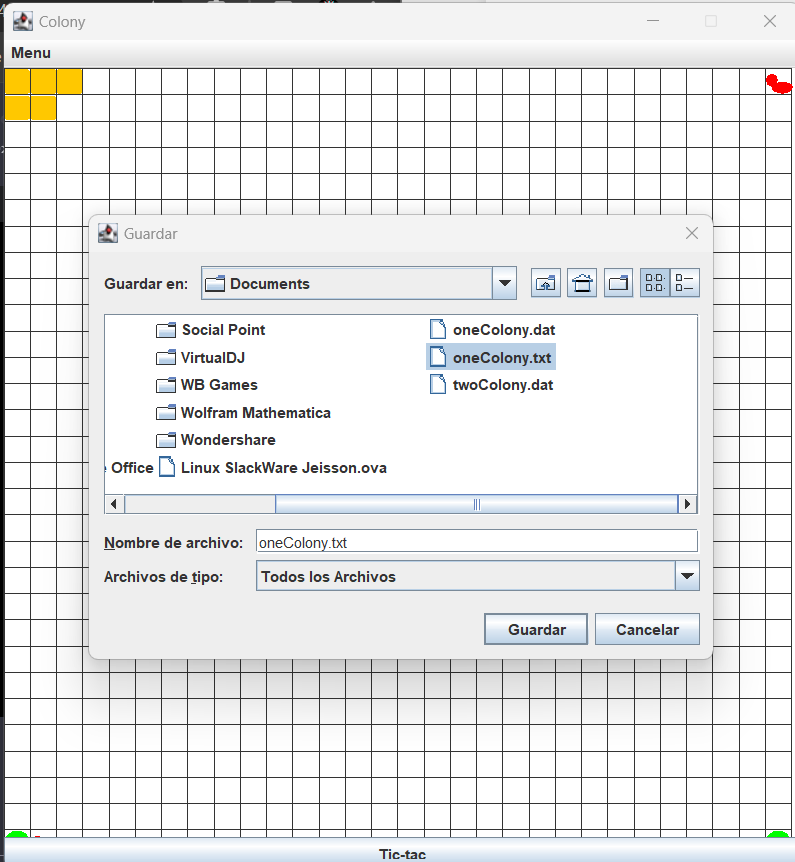


**6. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto correcto en oneColony.txt. e import este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.**

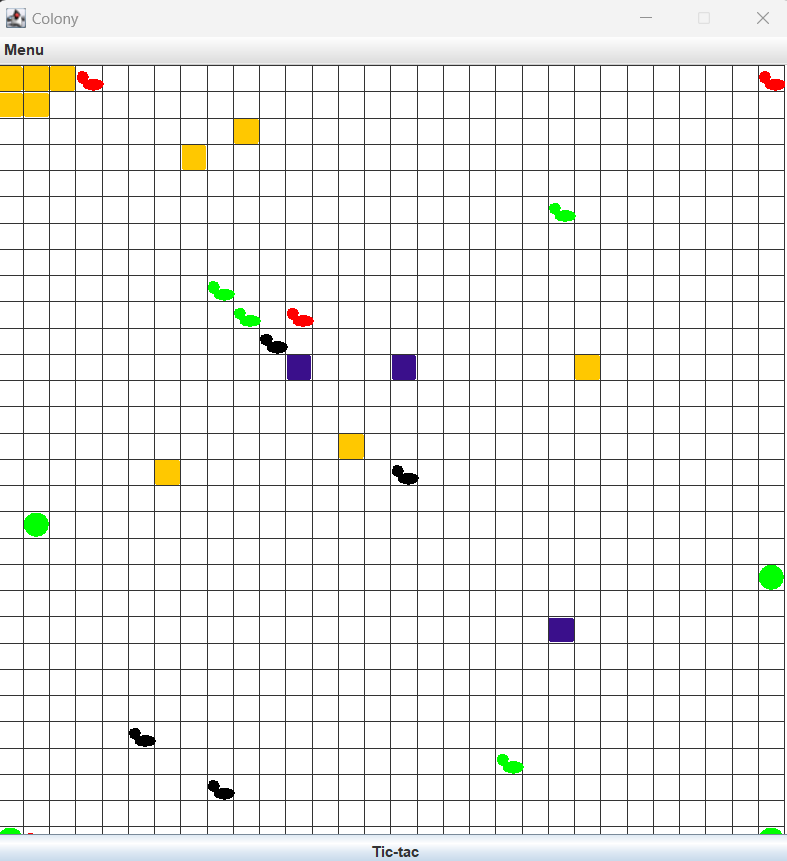
* Se modifica el archivo .txt de oneColony con nuevas coordenadas para cada entidad.



* Importamos el archivo



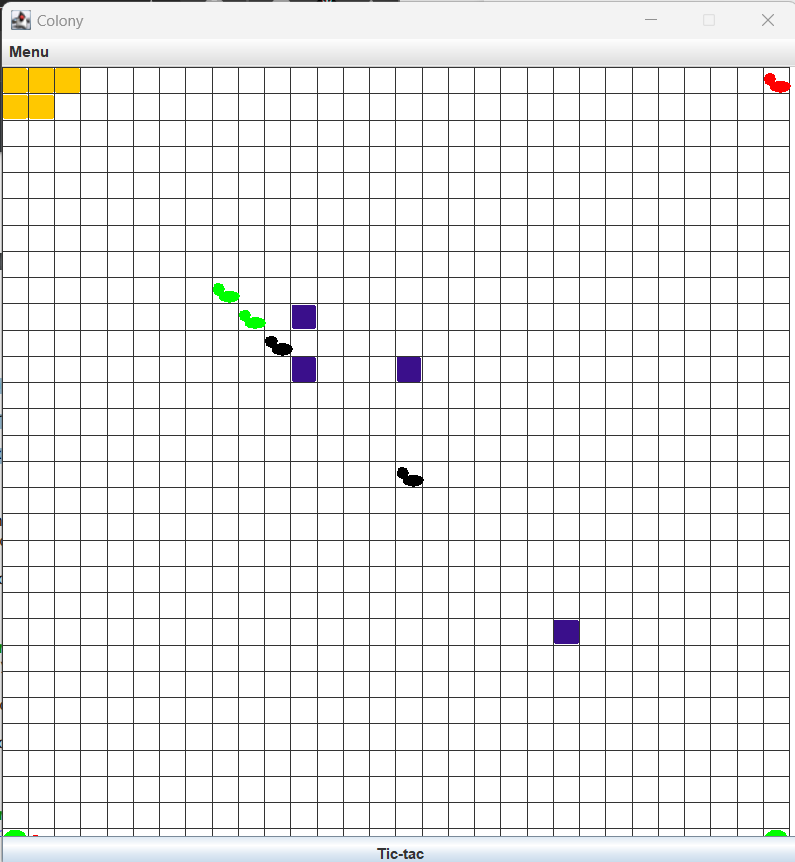
* Se crean nuevas entidades en el tablero.

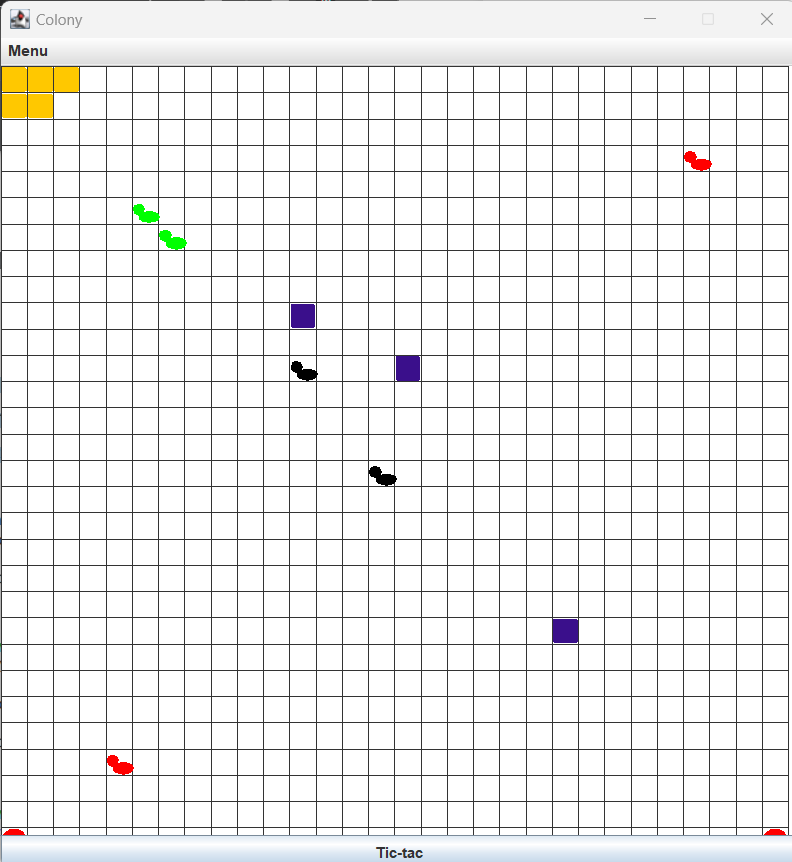


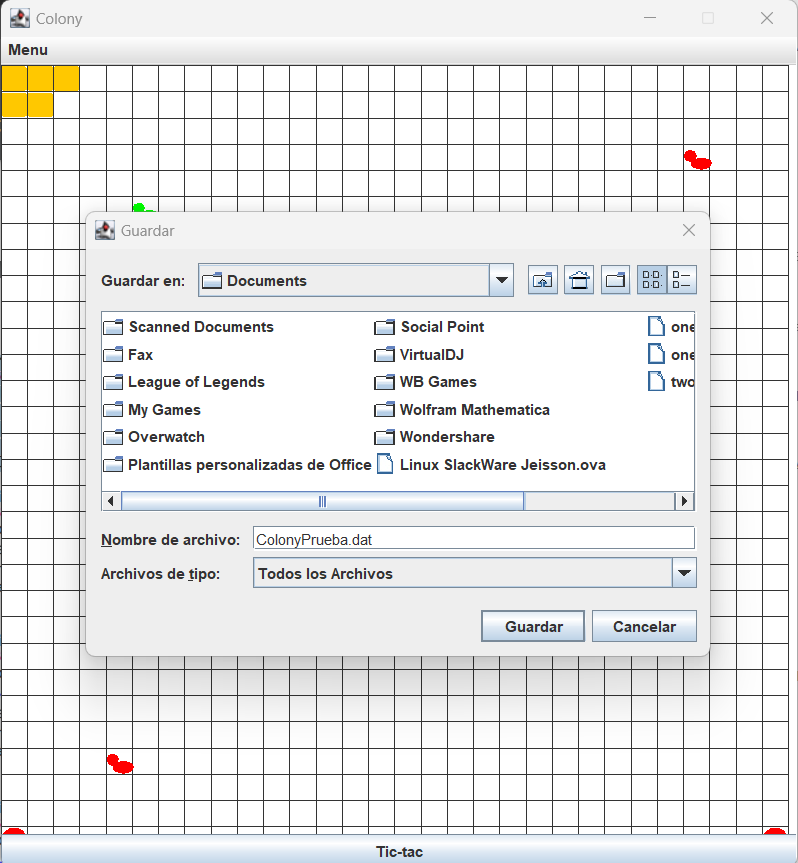
**Analizando comportamiento**

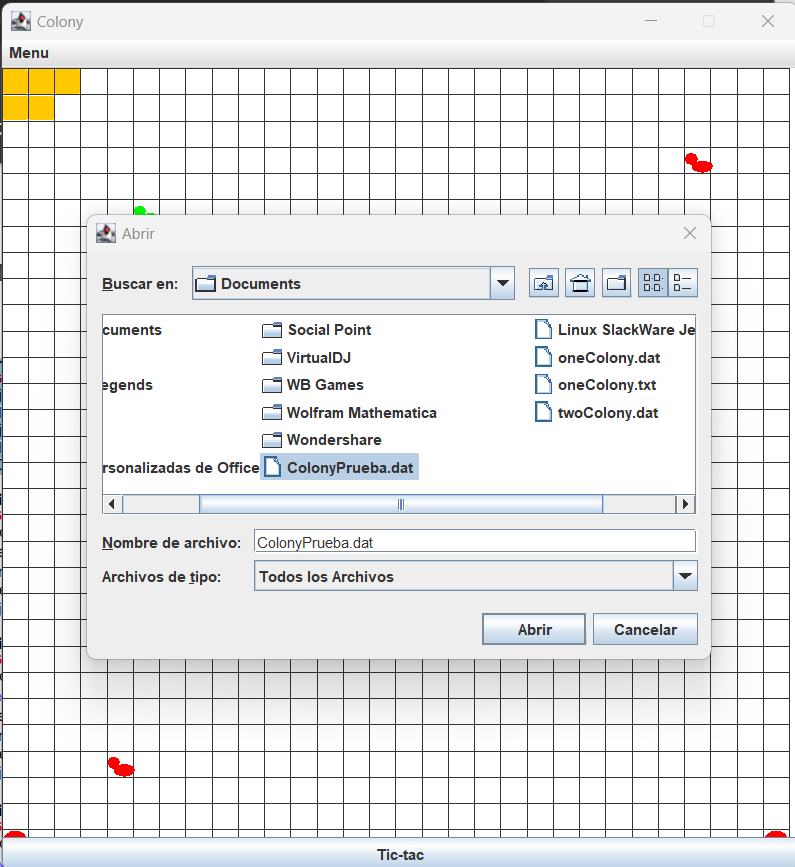
**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

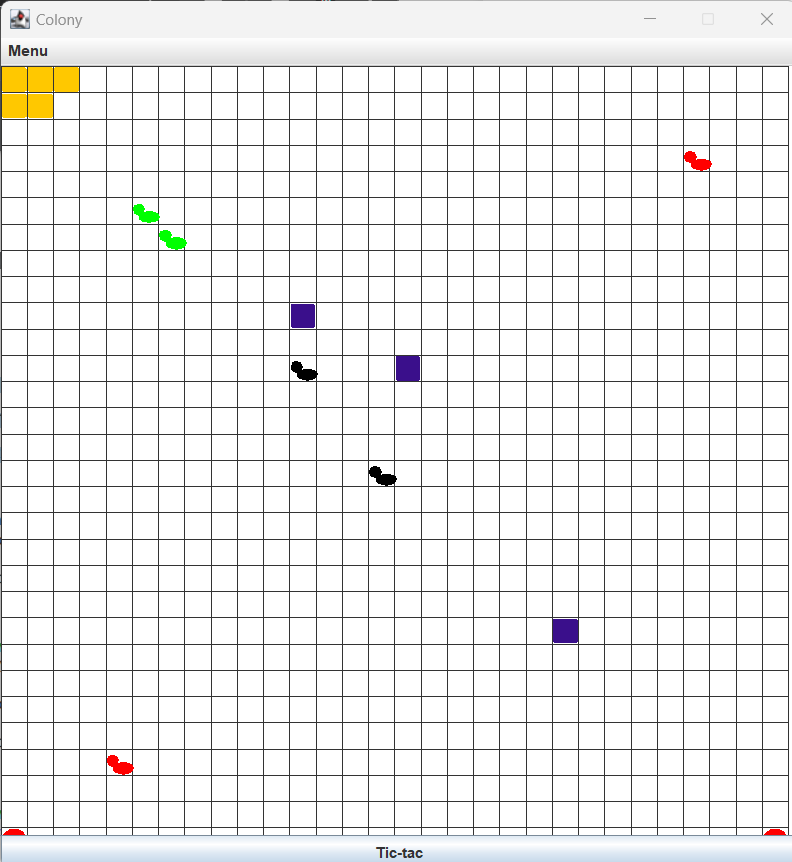
**1. Ejecuten la aplicación, den tres clics, salven a un archivo cualquiera y ábranlo. Describan el comportamiento**





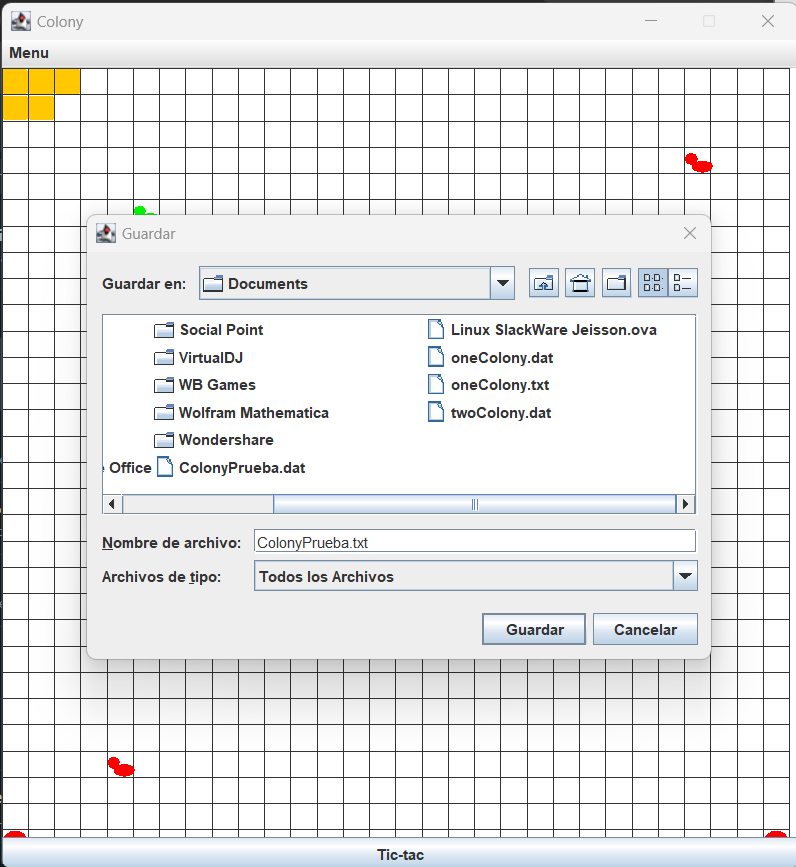




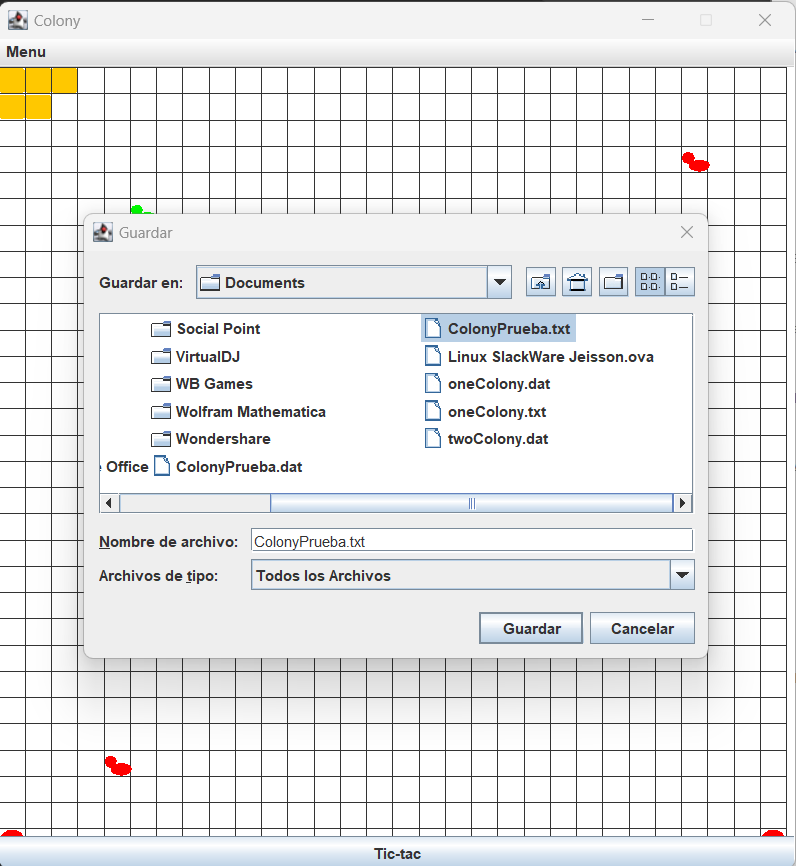


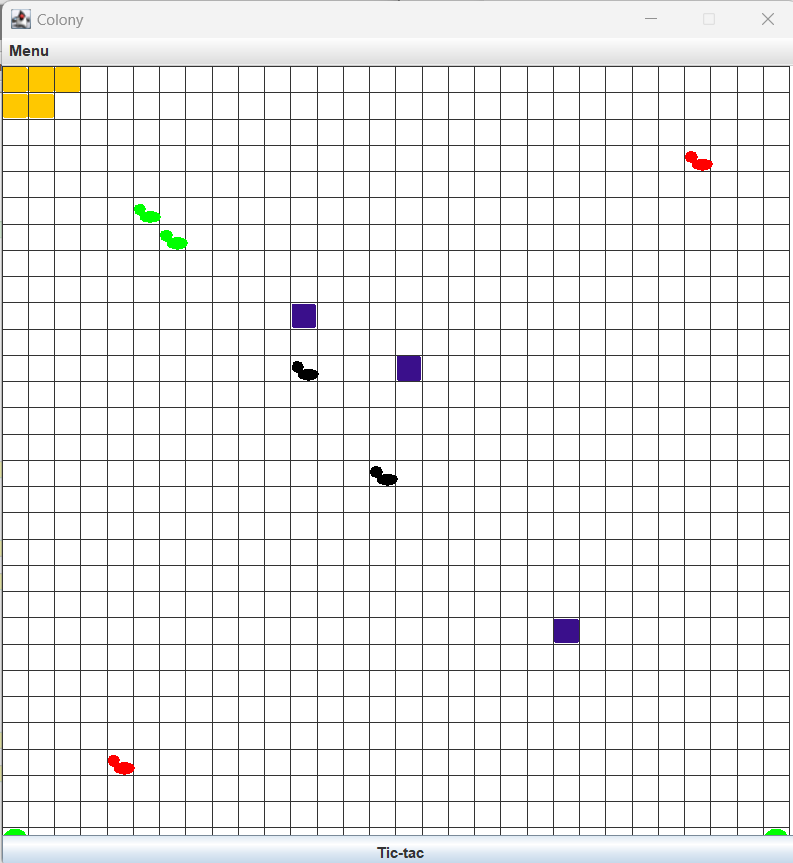
* Irónicamente, el tablero se mantiene exactamente igual, debido a que estamos guardando un archivo después de 3 tics y lo estamos abriendo sin realizar ningún tic adicional, por lo queda exactamente igual.

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



* Ahora lo importamos.





* De la misma forma, se mantiene el tablero exactamente igual, pues estamos exportando y luego volviendo a importar sin que haya modificaciones en el tablero, lo que hace internamente es sobrescribir las entidades que estaban en el tablero, que justamente estaban en las mismas posiciones.

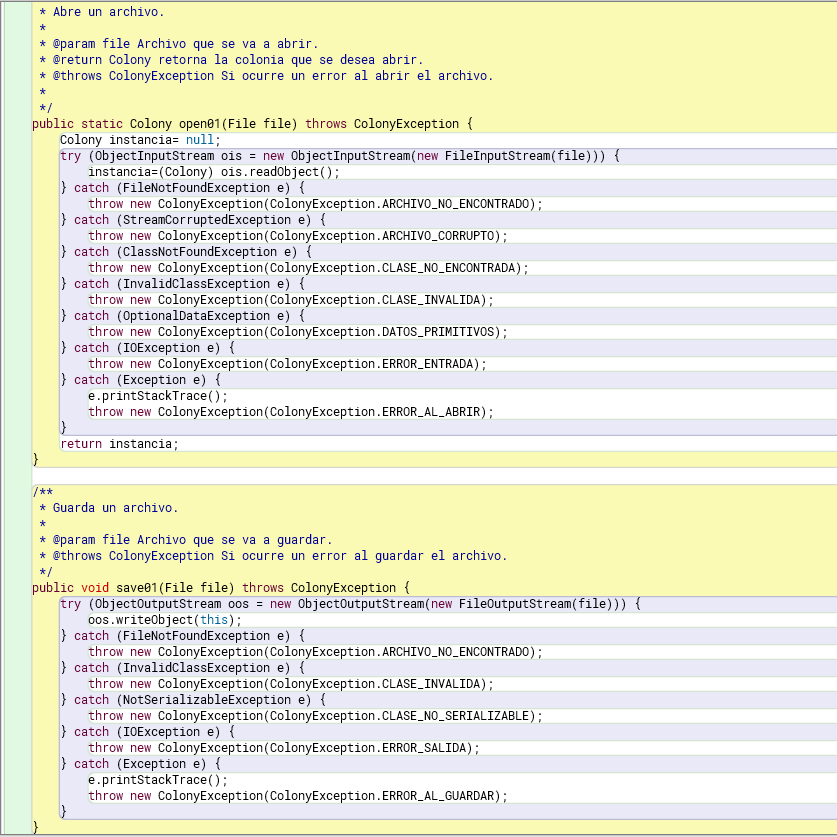
**3. ¿Qué diferencias ven el comportamiento 1? y 2.? Expliquen los resultados.**

Visualmente, no se ve diferencia en el comportamiento de estos 4 métodos, debido que para ver cambios tendríamos que realizar más tics, sin embargo, internamente si se ven los cambios, pues en el comportamiento 1, crea nuevas instancias, en el comportamiento 2 sobrescribe las entidades. Sin embargo, visualmente en esos ejemplos no es tan claro verlo.

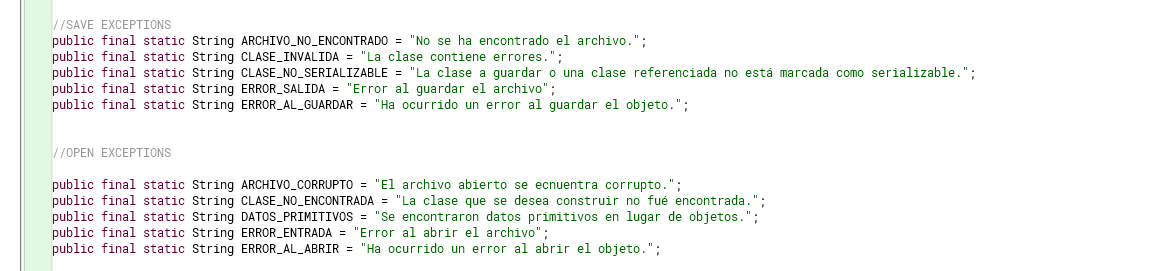
**Perfeccionando salvar y abrir**

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

**1. Copien las versiones actuales de open y save y renómbrenlos como open01 y save01**

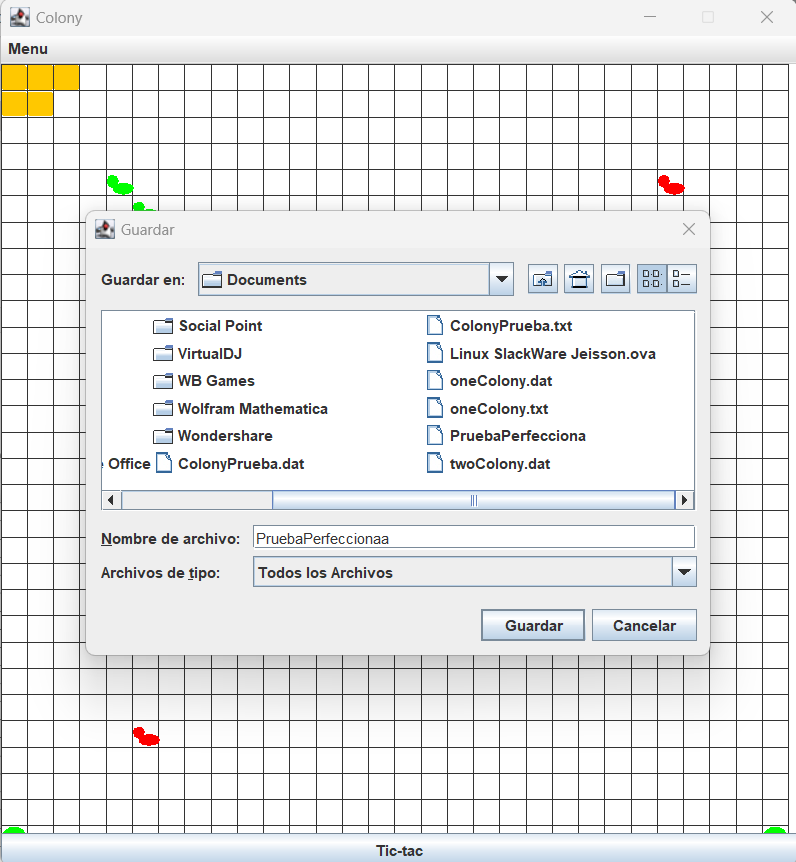


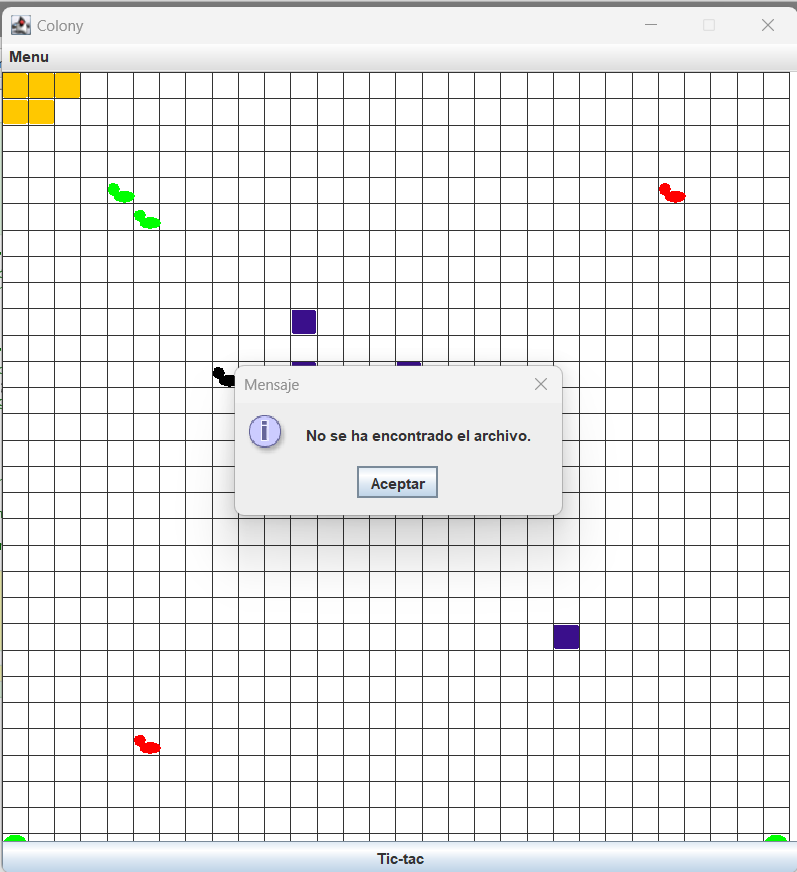
**2. Perfeccionen el manejo de excepciones de los métodos open y save detallando los errores.**



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

* Si tratamos de abrir un archivo el cual no existe, nos vamos a encontrar con el siguiente error.

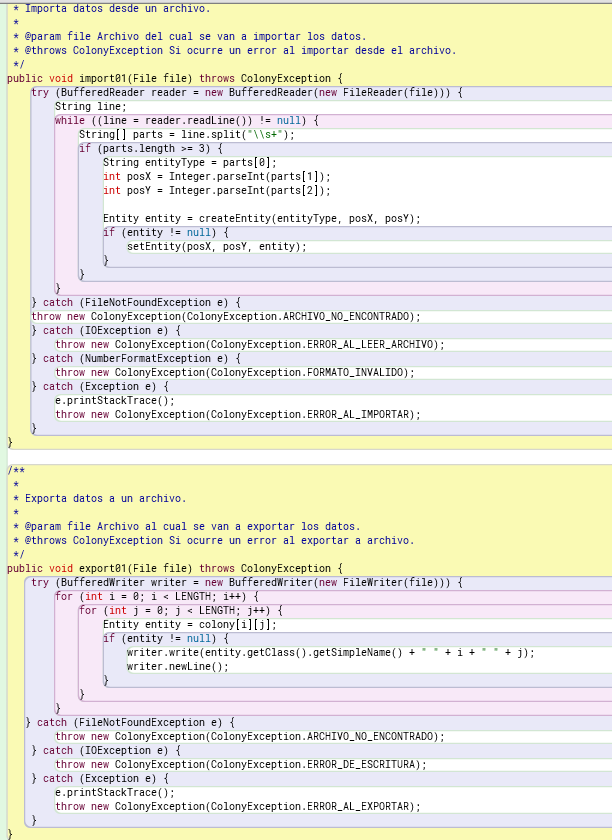


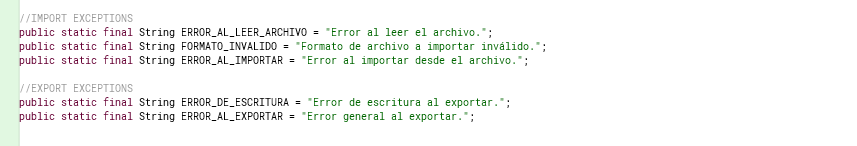


**Perfeccionando importar y exportar.**

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

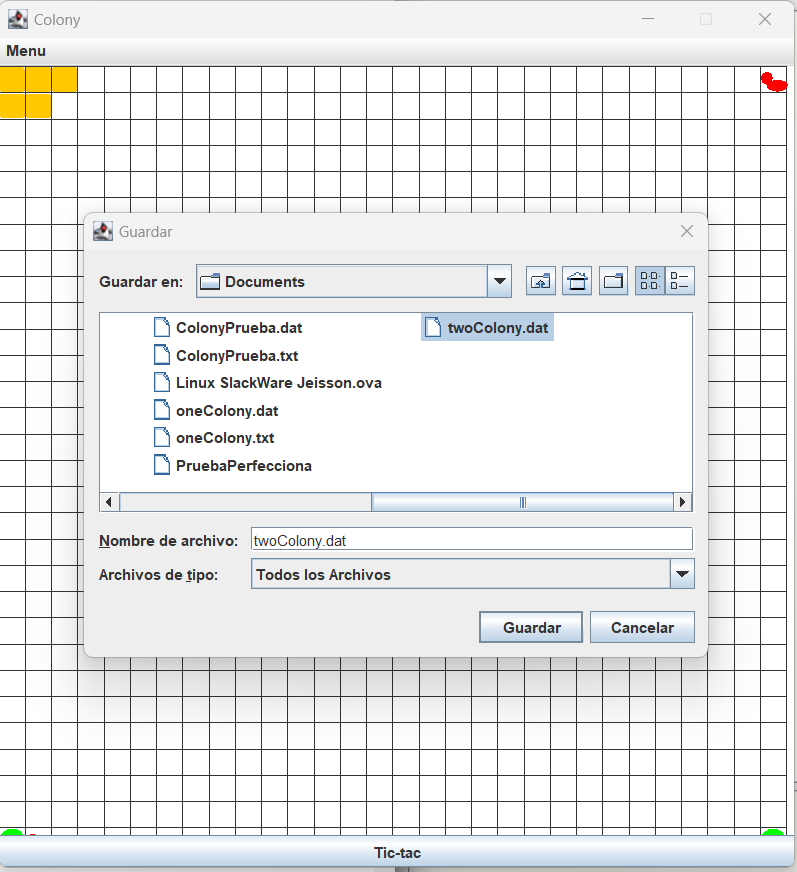
**1. Copien las versiones actuales de import y export y renómbrenlos como import01 y export01**

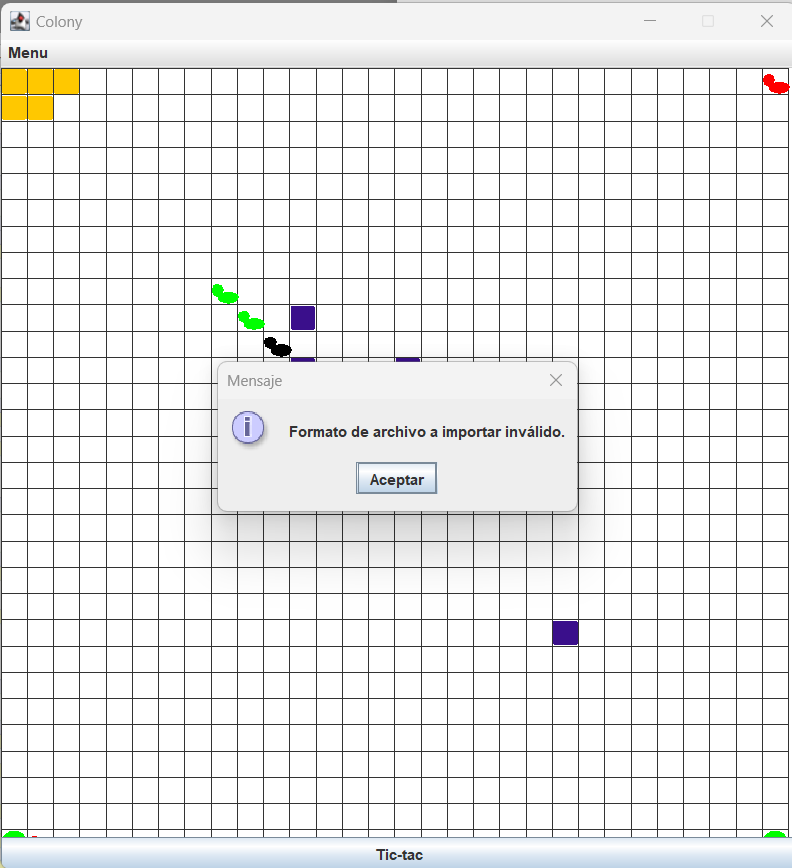


**2. Perfeccionen el manejo de excepciones de los métodos import y export detallando los errores.** 

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

* Si tratamos de importar un archivo con que no coincide con el formato .txt, vamos a obtener el siguiente mensaje de error.



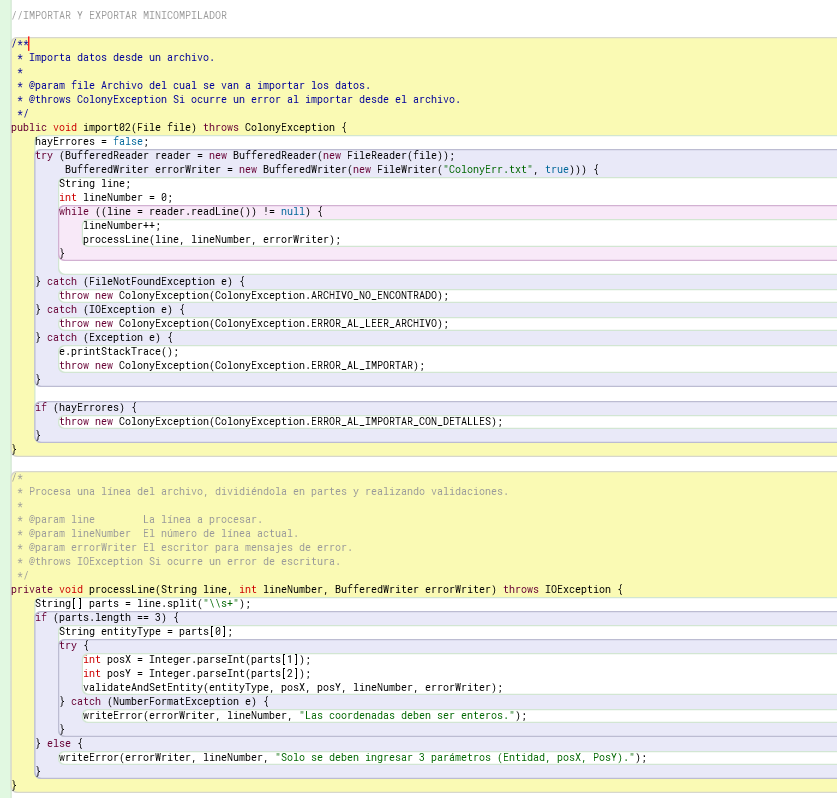


**Perfeccionando importar. Hacia un minicompilador.**

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

**1. Copien las versiones actuales de import y export y renómbrenlos como import02 y export02**

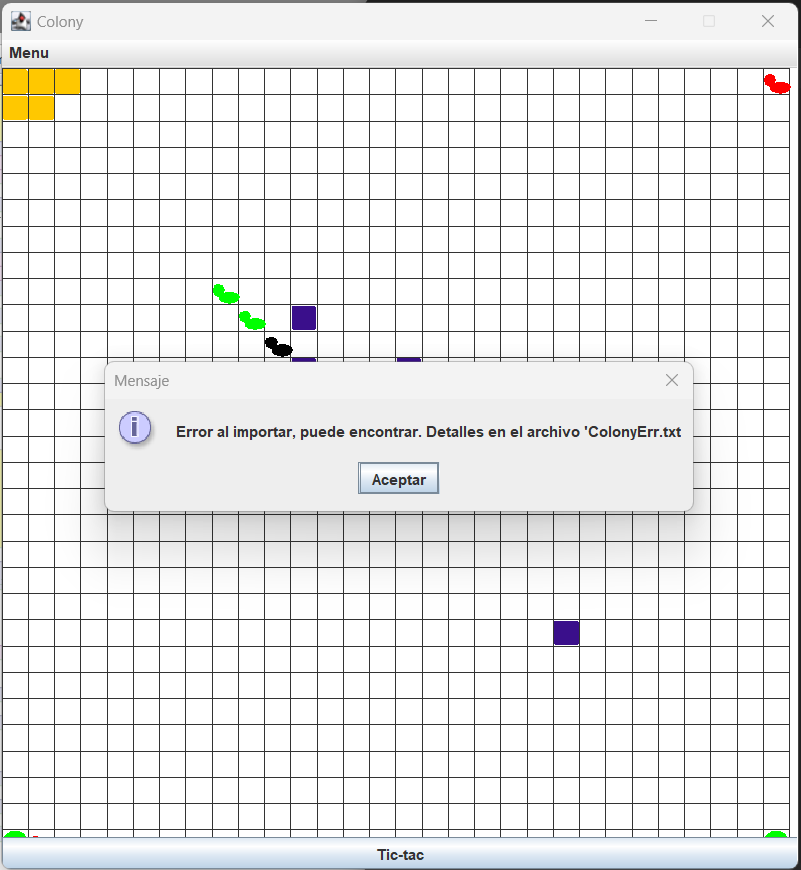
**2. Perfeccionen el método import para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palopen que tiene el error y causa de error.**

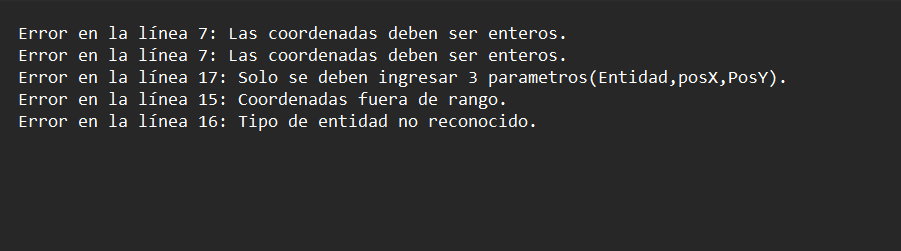




**3. Escriban otro archivo con errores, llámelo colonyErr.txt, para ir arreglándolo con ayuda de su “importador”. Presente las pantallas que contengan los errores.**

* Se verifica que no haya coordenadas mayores al tablero, que se ingresen números enteros, que línea tenga 3 elementos, y que la entidad sea reconocida por el programa. Las excepciones las podemos visualizar en el bloc de notas, mientras que visualmente en el simulador vamos a ver un mensaje que nos notificara que fue enviada al miniCompilador.





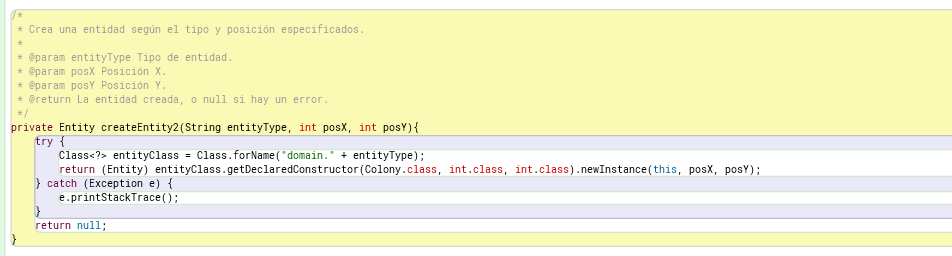
**BONO. Perfeccionando importar. Hacia un minicompilador flexible.**

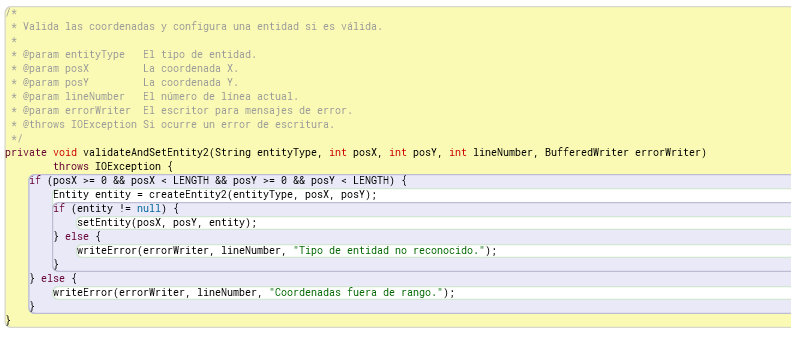
**[En lab06.doc, \*.asta , colonyFlex.txt \*.java] [NO OLVIDEN BDD y MDD]**

**1. Copien las versiones actuales de import y export y renómbrenlos como import03 y export03**

**2. Perfeccionen los métodos import y export para que pueda servir para cualquier tipo de elementos creados en el futuro (Investiguen cómo crear un objeto de una clase dado su nombre)**

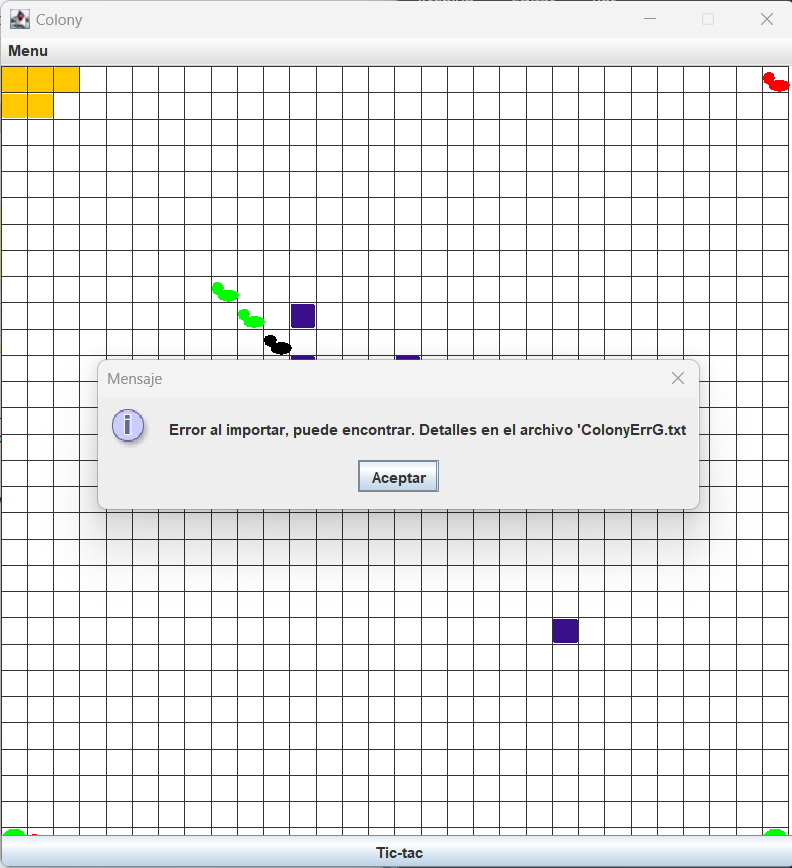


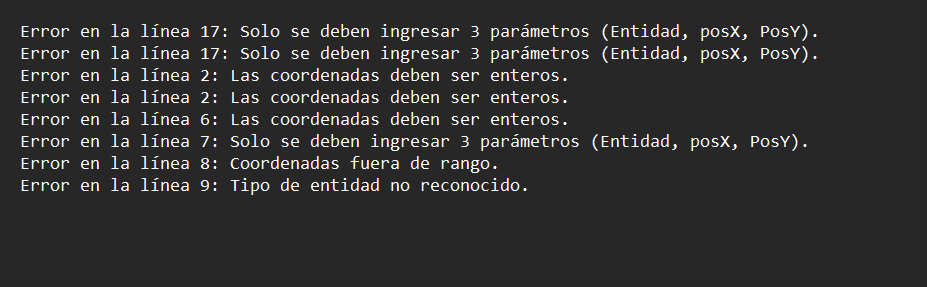




Cambiamos el método que crea las entidades, por el metodo de reflexividad en java, el cual nos permite Instanciar una clase a partir de su nombre. Una vez usamos este método en Import03, veremos que funciona de igual forma y que en caso de tener que extender las entidades, no tendremos problema.

**3. Escriban otro archivo de pruebas, llámelo colonyErrG.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)**

12 horas por persona.

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

Se encuentra realizado en su totalidad.

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

Refactorización del código, pues así pudimos simplificar métodos muy grandes.

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

Entender el funcionamiento de los métodos de persistencia, debido a que nunca los habíamos visto.

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

La investigación de los métodos y el lograr que estos quedaran funcionales, se resolvió dedicándole tiempo al laboratorio

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

Lo que hicimos bien como equipo fue hacer el laboratorio con tiempo.