PROGRAMACIÓN ORIENTADA A OBJETOS

Persistencia

Laboratorio 6/6

Integrantes:

Daniel Useche

Daniel Patiño

2025-1

DESARROLLO

Preparando

En este laboratorio vamos a extender el proyecto schelling 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

Abrimos la consola y ubicamos el directorio con el proyecto realizado en el laboratorio 3

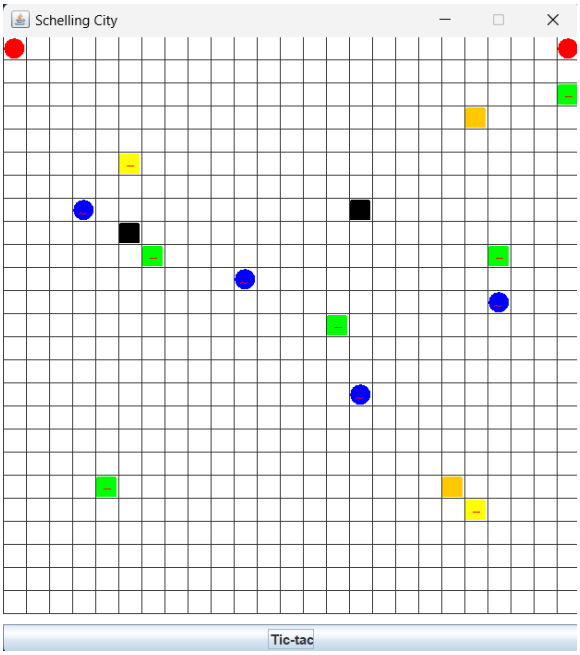
```
C:\Users\danpa>cd "C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\LAB 03\Consola\schelling"

C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\LAB 03\Consola\schelling>
```

2. Ejecuten el programa, revisen la funcionalidad.

Ejecutamos el .jar que teniamos previamente para revisar su funcionalidad

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\LAB 03\Consola\schelling>schelling.jar
```



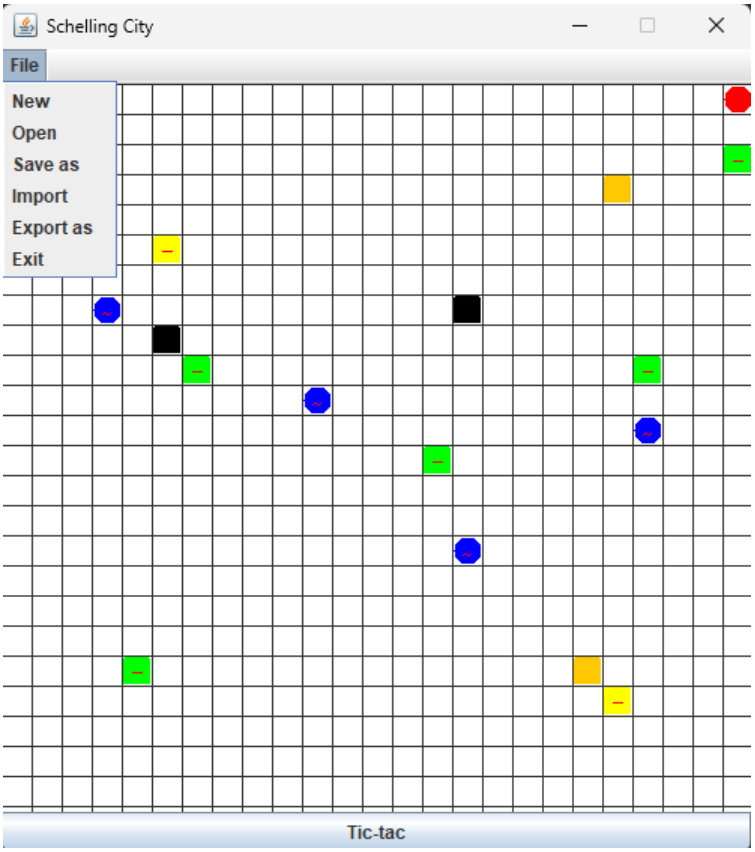
Creando la maqueta [En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

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

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

Revisar schelling/src/domain/City.java y schelling.astah

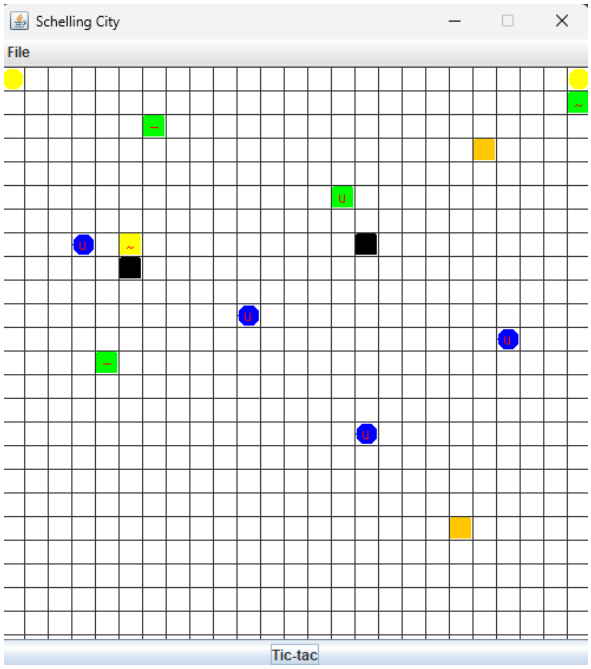
2. VISTA: Construyan un menú barra que ofrezca, además de las opciones básicas de entrada-salida, las opciones estándar de nuevo y salir (Nuevo, Abrir, Guardar como, Importar, Exportar como, Salir). No olviden incluir los separadores. Para esto creen el método prepareElementsMenu. Capturen la pantalla del menú.



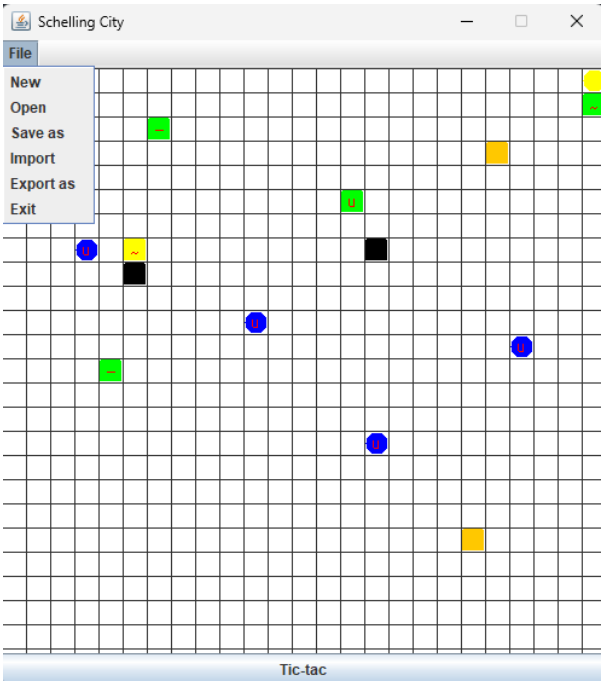
3. CONTROLADOR: Construyan los oyentes correspondientes a las seis opciones. Para esto creen el método prepareActionsMenu y los métodos base del controlador (optionOpen, optionSave, optionImport, optionExport, optionNew, optionExit), En las opciones que lo requieran usen un FileChooser y atiendan la excepción. Estos métodos llaman el método correspondiente de la capa de dominio que por ahora sólo lanza una excepción. Ejecuten las diferentes acciones del menú y para cada una de ellas capture una pantalla significativa.

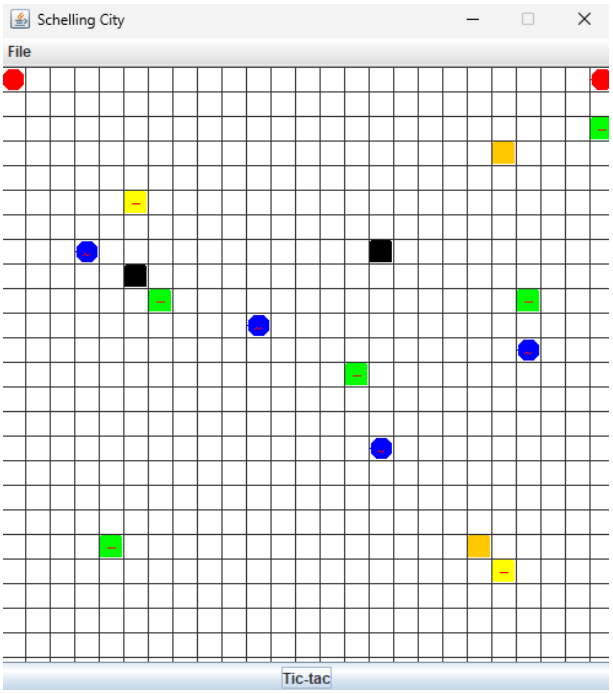
Para new:

Juego actual:



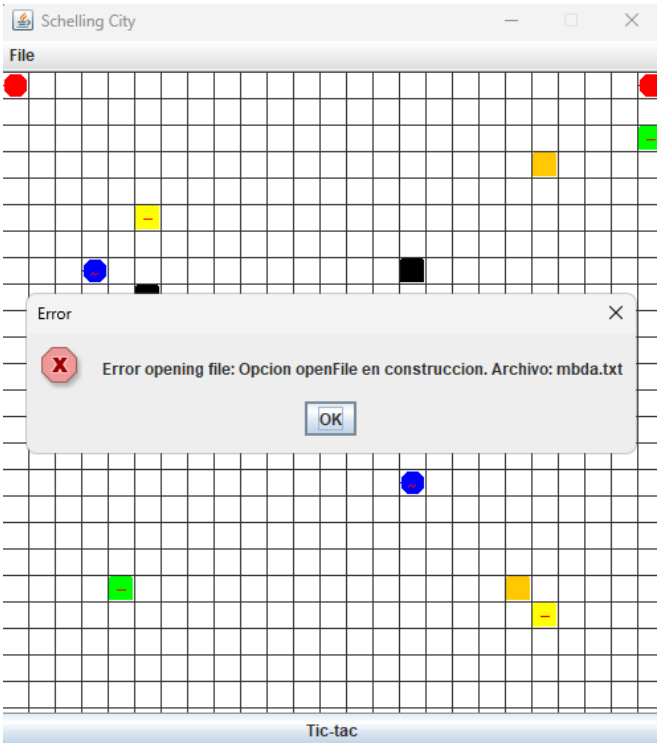
Si le damos a File y luego a new:



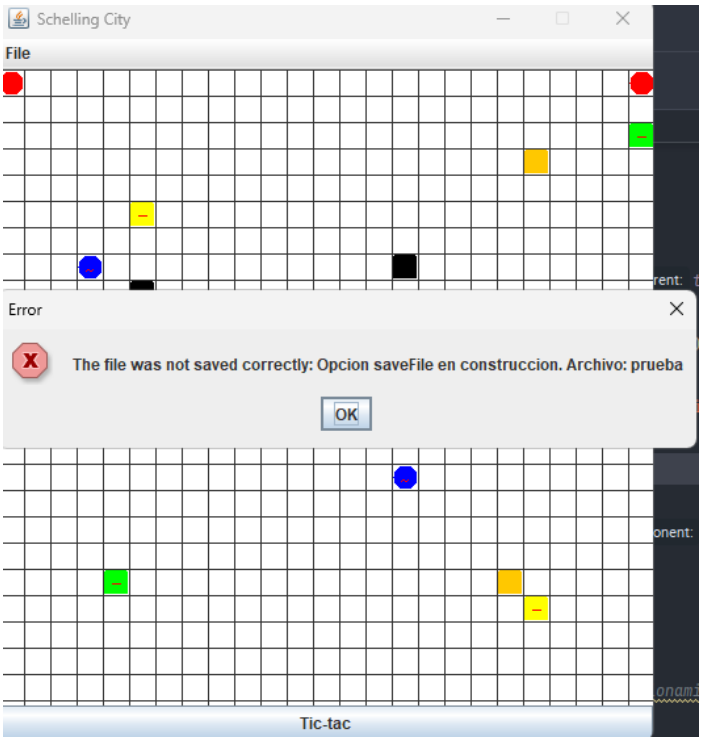


Como se puede ver, se crea un juego nuevo.

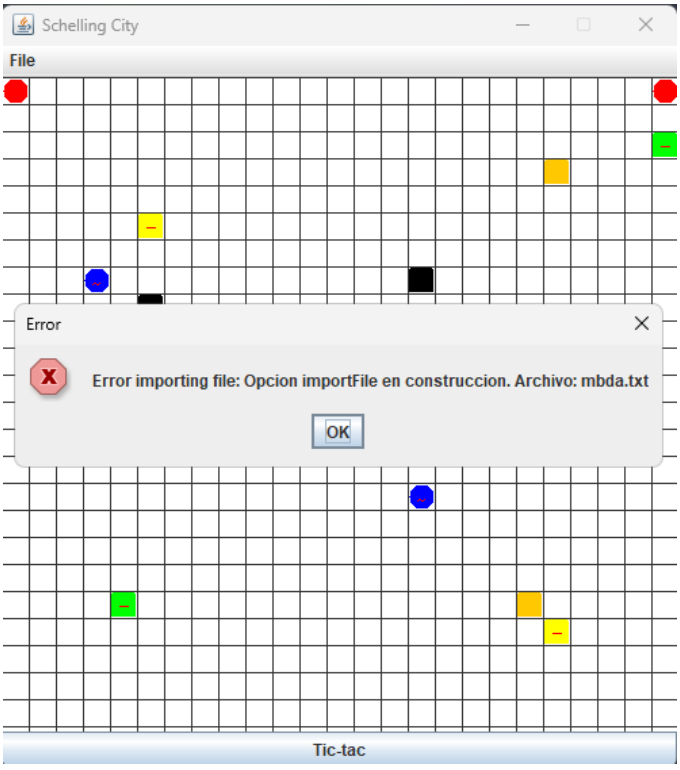
Para la opcion open:



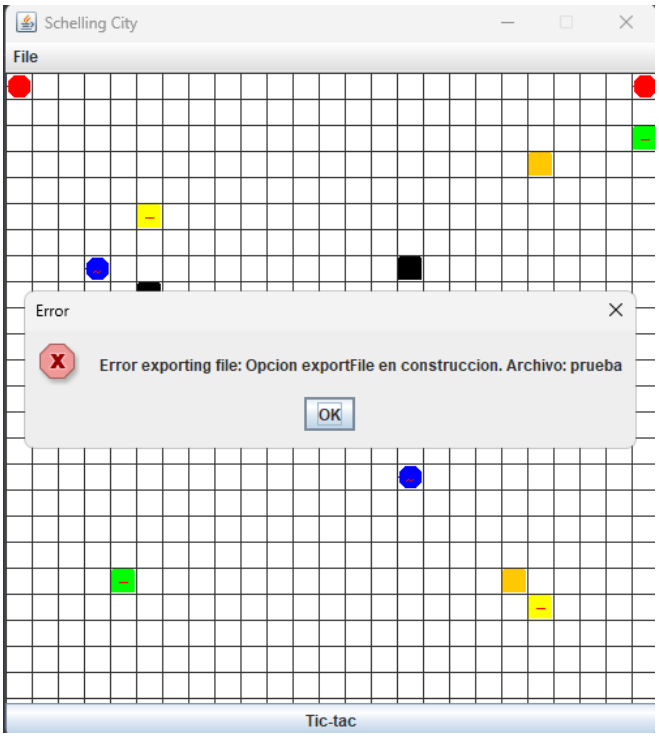
Para la opcion Save As:



Para Import:



Para Export As:



Implementando salir y nuevo

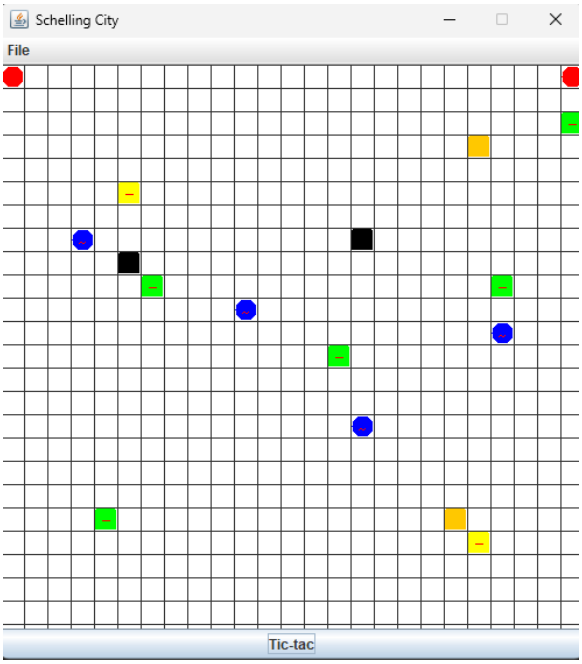
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD] Las opciones salir y nuevo van a ofrecer los dos servicios estándar de las aplicaciones. El primero no requiere ir a capa de dominio y el segundo sí.

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

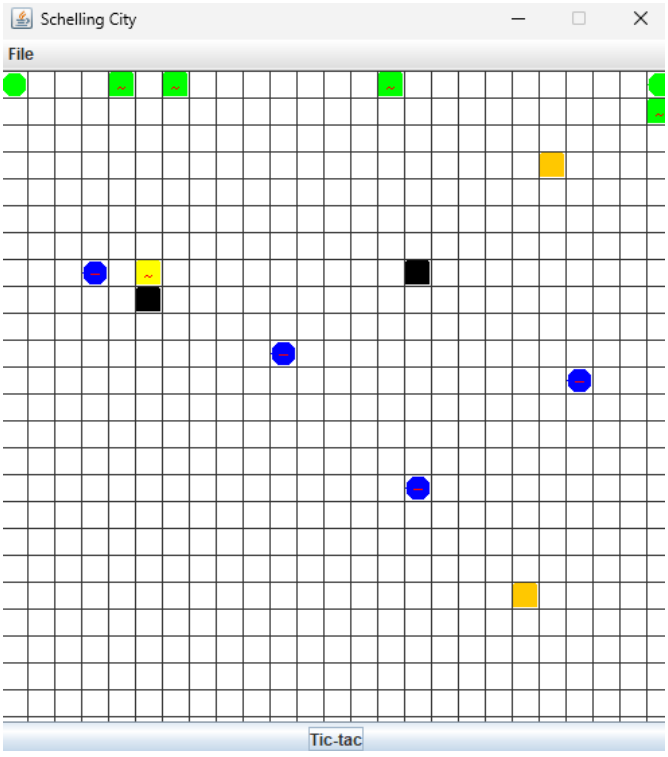
   Revisar schelling/src/CityGUI.java

2. Construyan el método optionNew que crea una nueva ciudad. Capturen una pantalla significativa.

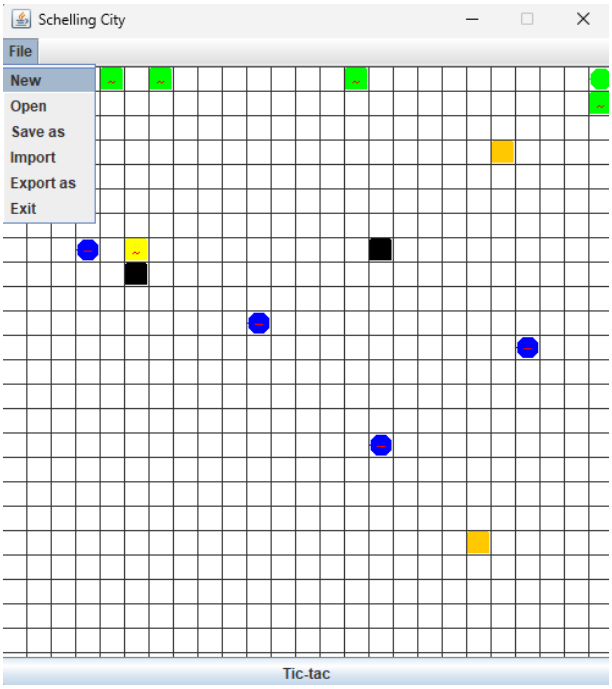
Ejecutamos la aplicacion:



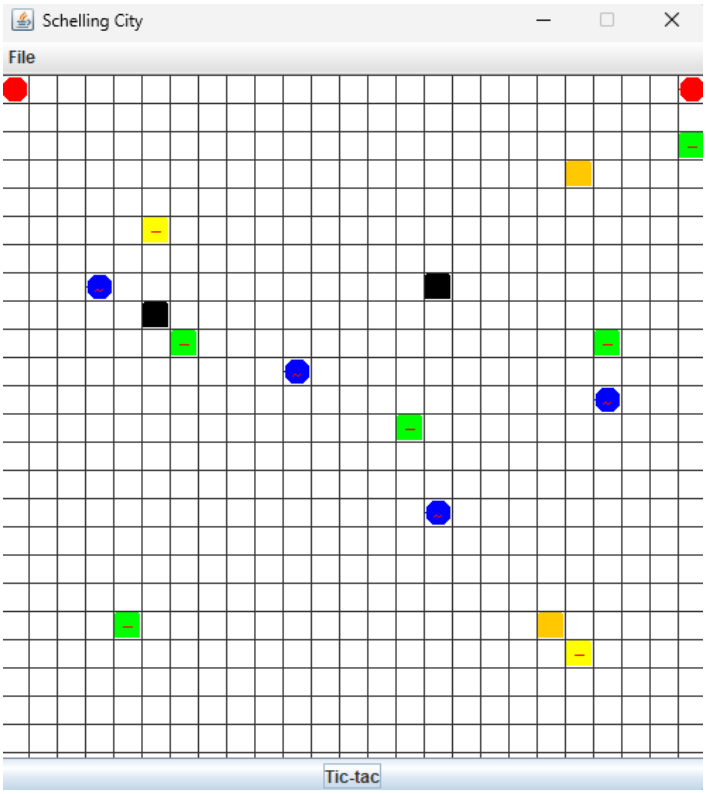
Le damos tic tac varias veces para que el tablero no sea el mismo:



A continuacion le damos en File y en new:



Al darle clic se generará la ciudad con los items como los teníamos cuando ejecutamos la aplicación ya que son items creados junto con el tablero:



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 la ciudad 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

   Revisar schelling/src/domain/City.java

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

Revisar schelling/src/domain/City.java

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

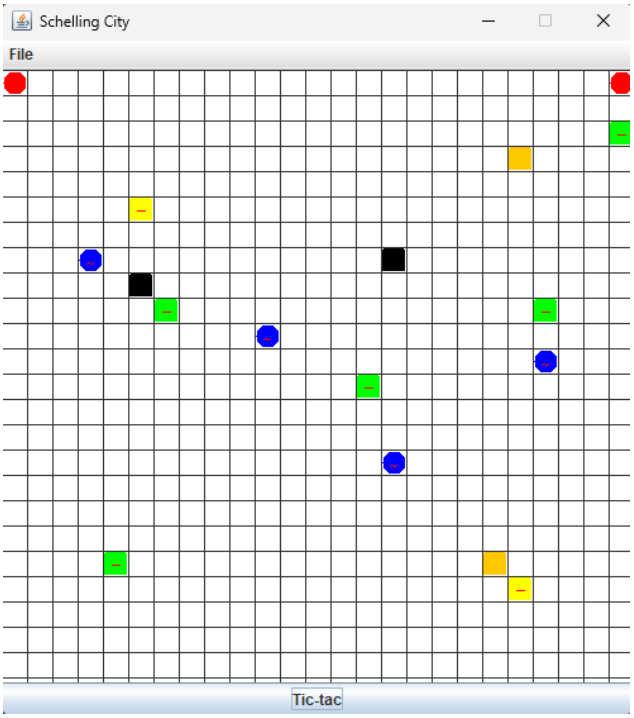
El archivo se creó en el disco y tiene un tamaño de 2KB

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

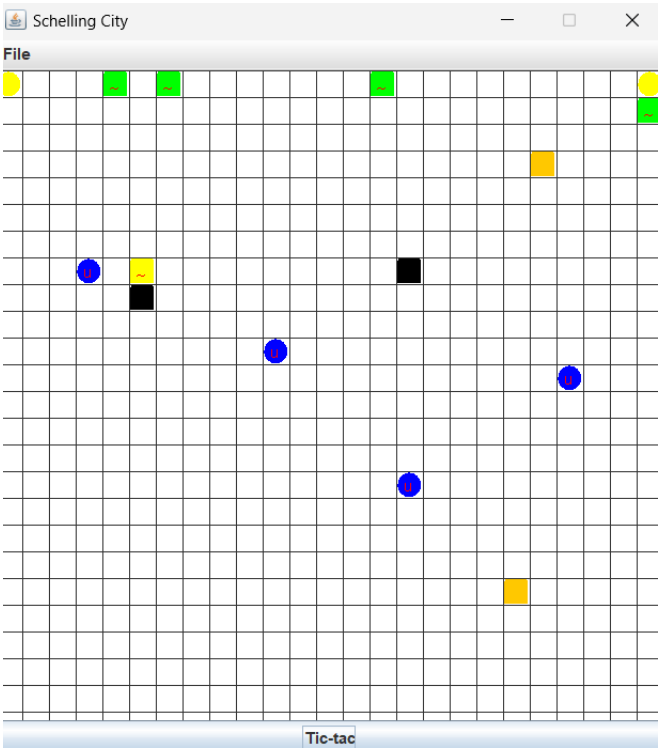
Revisar schelling/src/domain/City.java

5. Realicen una prueba de aceptación para este método iniciando la aplicación, creando un nuevo estado y abriendo el archivo oneCity.dat. Capturen imágenes significativas de estos resultados.

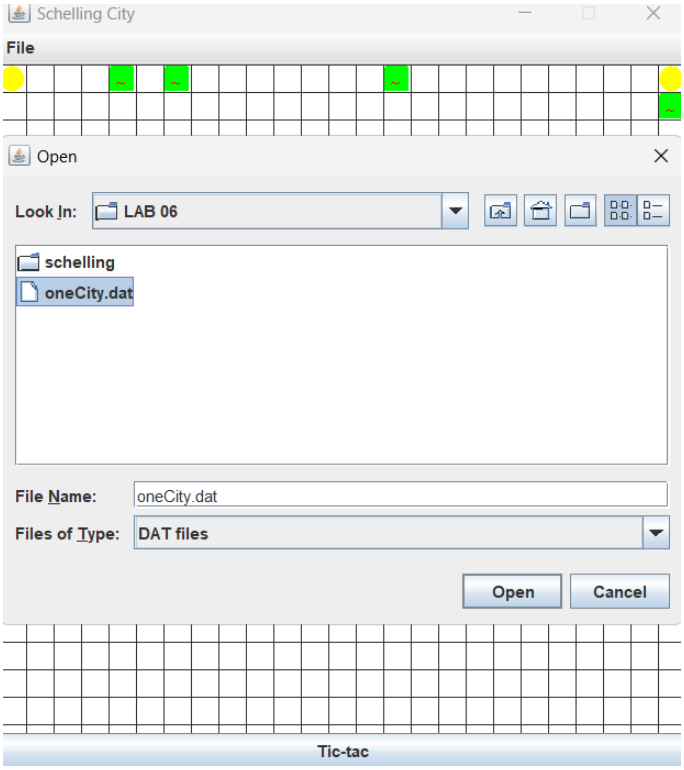
Ejecutamos la aplicacion:



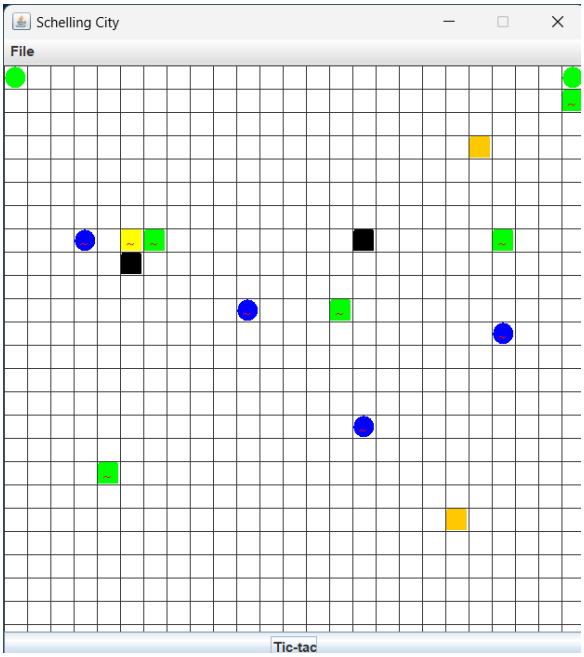
A continuación modificaremos dando un par de tic tacs para que no quede igual al default:



Ahora, abriremos el archivo .dat dándole a File y luego a open y buscamos el archivo:



Le damos en oneCity.dat y luego le damos en Open.



Como se puede ver en la imagen, el archivo se cargó correctamente.

Implementando importar y exportar

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

Estas operaciones nos van a permitir importar información de la ciudad desde un archivo de texto y exportarla. 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.

Person 10 10

Walker 20 20

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

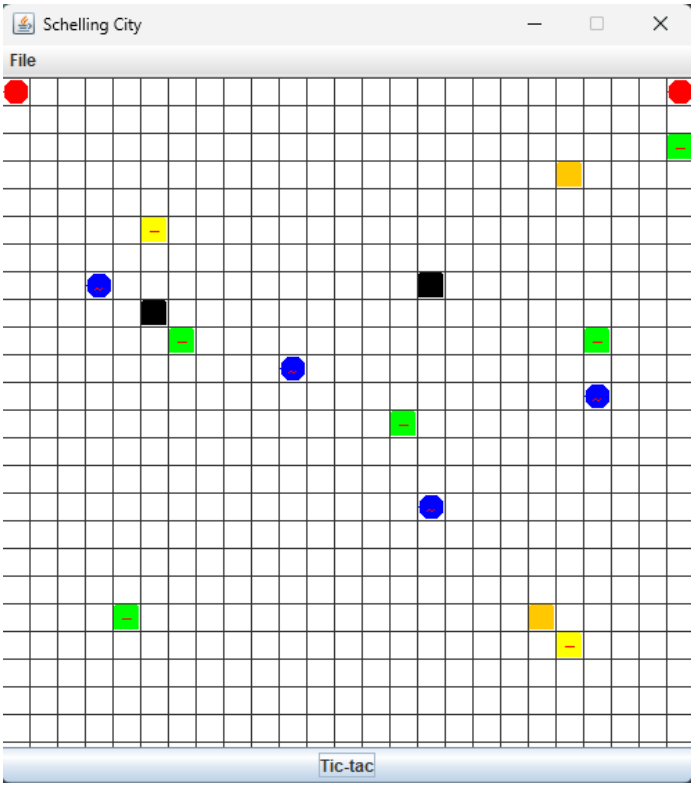
   Revisar schelling/src/domain/City.java

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.

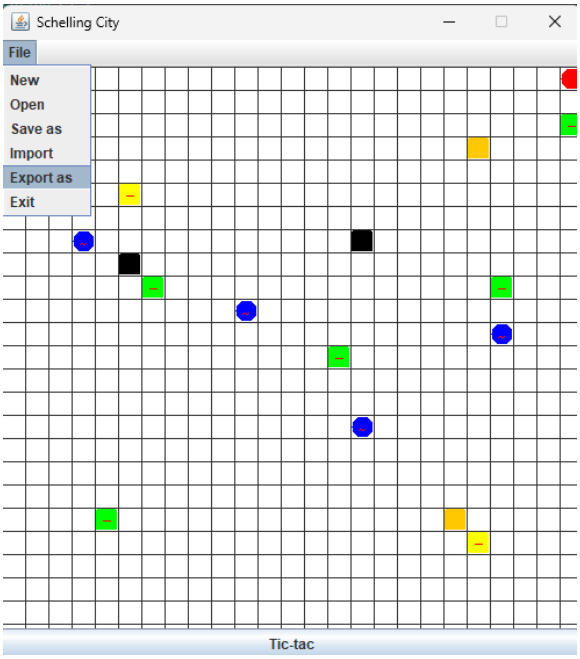
   Revisar schelling/src/domain/City.java

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

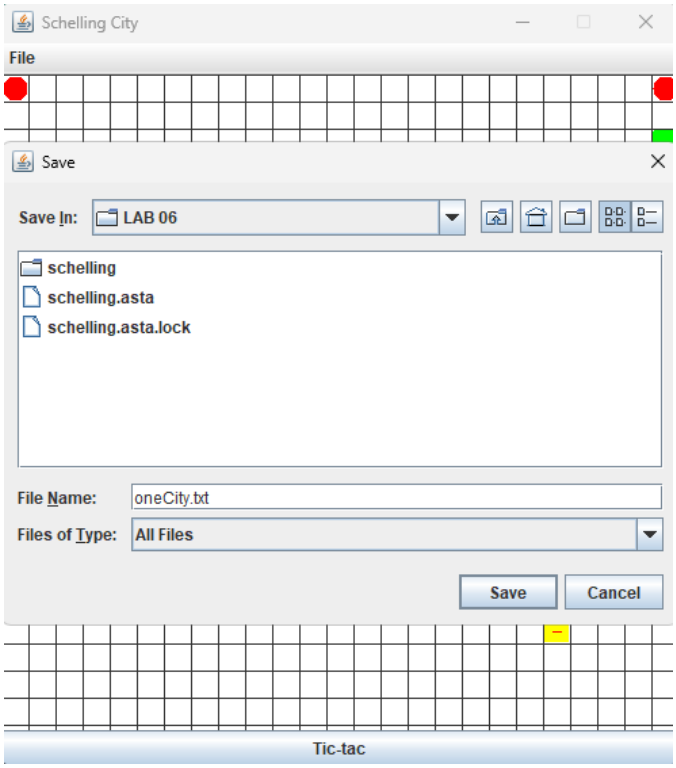
Iniciamos la aplicación:



Seguido le damos en file y export as:

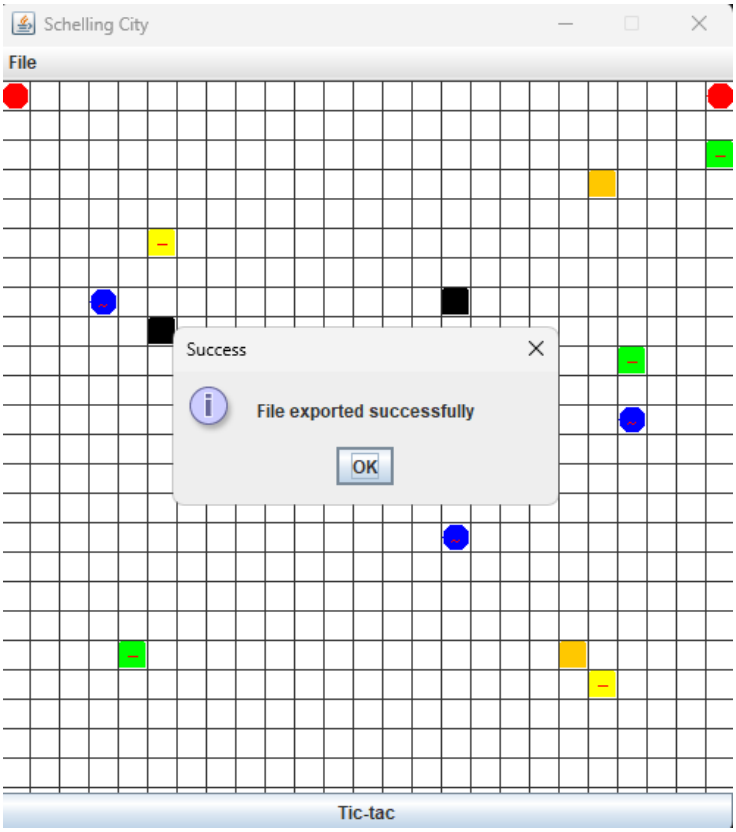


Se nos abrirá un JFileChooser donde elegiremos donde se guardará el archivo:

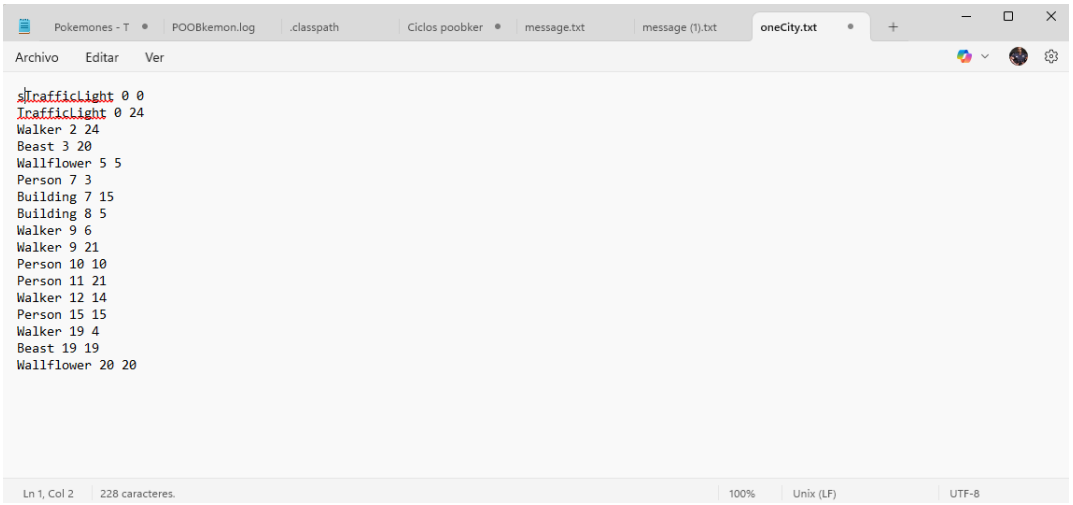


Guardamos el archivo como oneCity.txt y le damos en save.

Al guardarlo se nos abrirá un mensaje de que el archivo ha sido exportado exitosamente.



A continuacion ubicaremos el archivo y lo editaremos para ver su contenido:



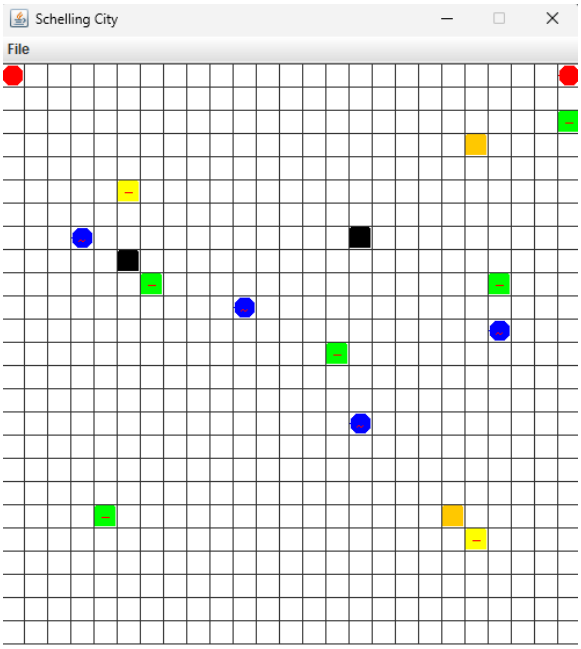
Como se puede ver en la imagen, se exportaron todos los Items que hay en la ciudad junto con su posición dentro de ella.

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)

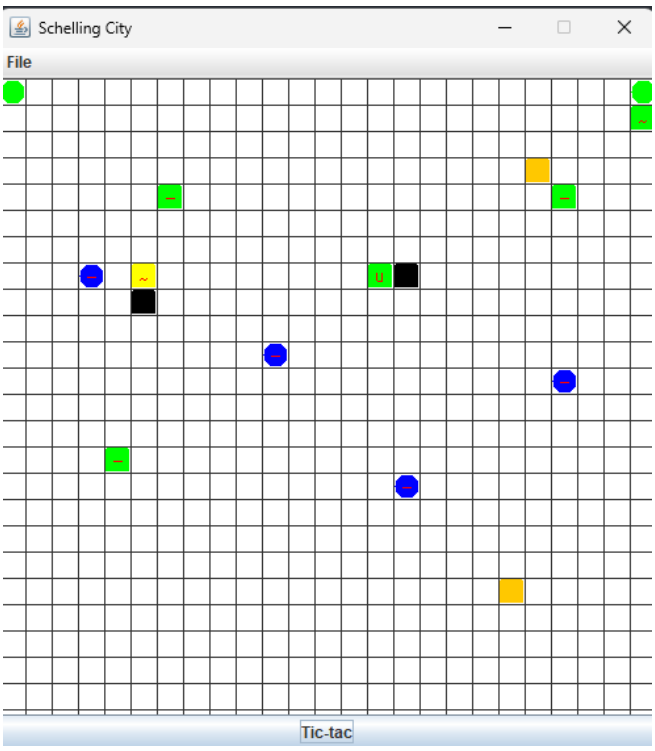
   Revisar schelling/src/domain/City.java

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

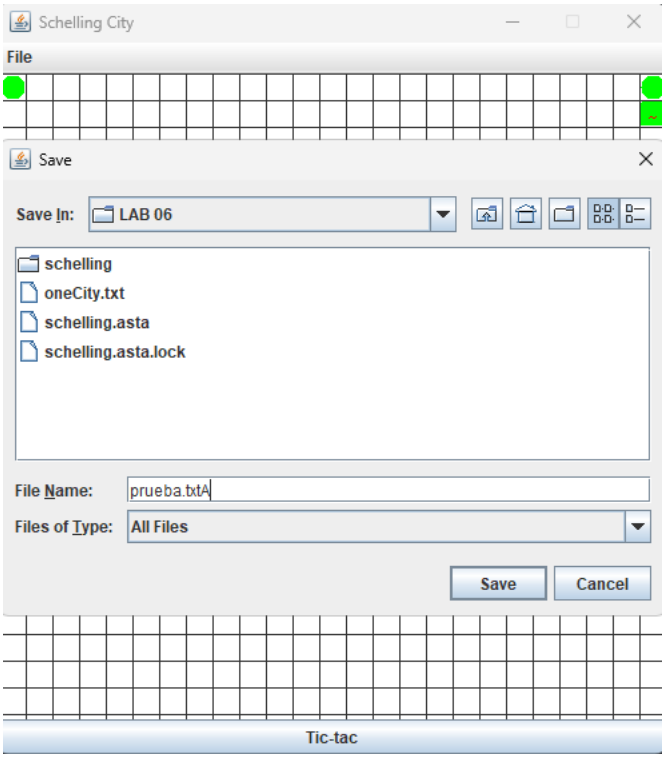
   Iniciamos la aplicación:



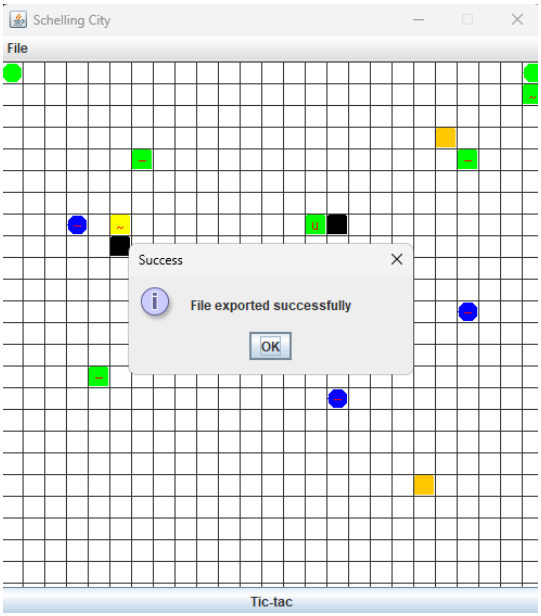
   Le damos en tic tac varias veces para modificar las posiciones de los objetos

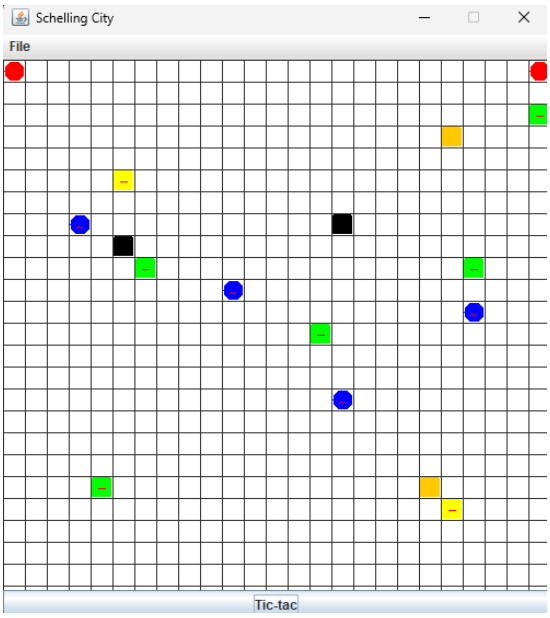


Al tenerlo modificado, le damos en file y lo exportamos a un archivo que se llamara prueba.txt

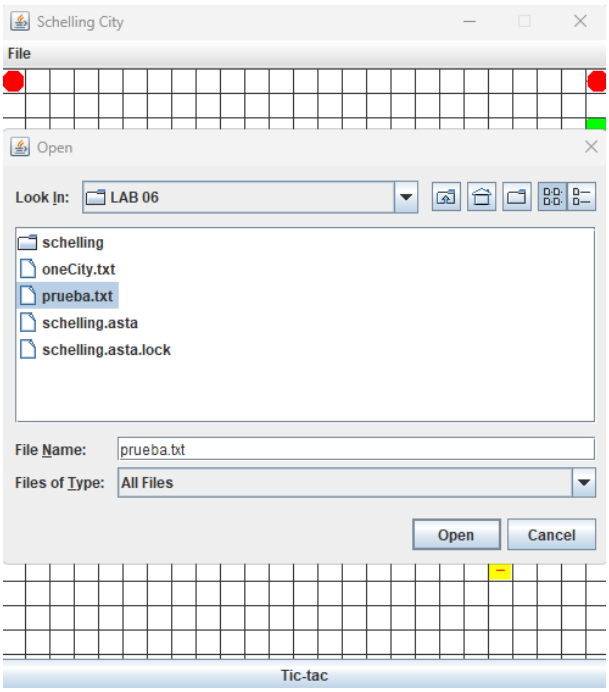


Le damos en save y creamos un nuevo juego:

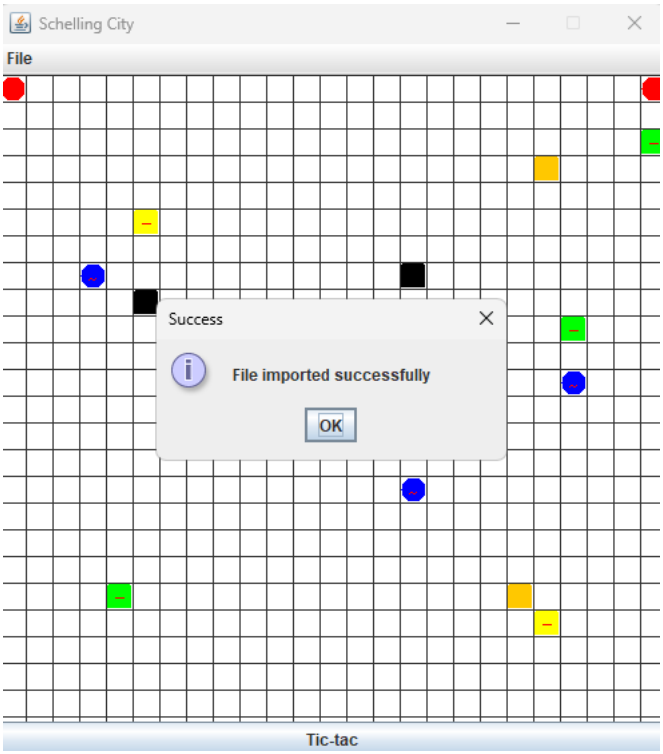




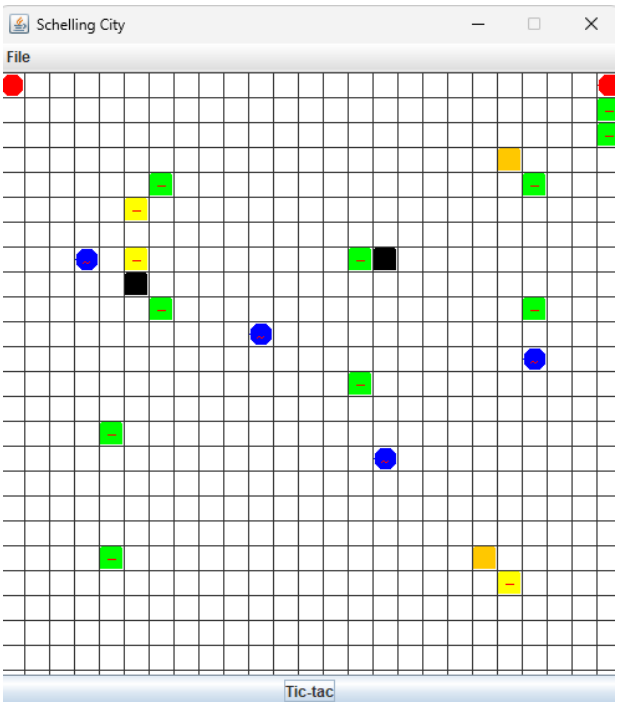
Ahora le damos en file e importamos el archivo prueba.txt



Le damos en abrir:

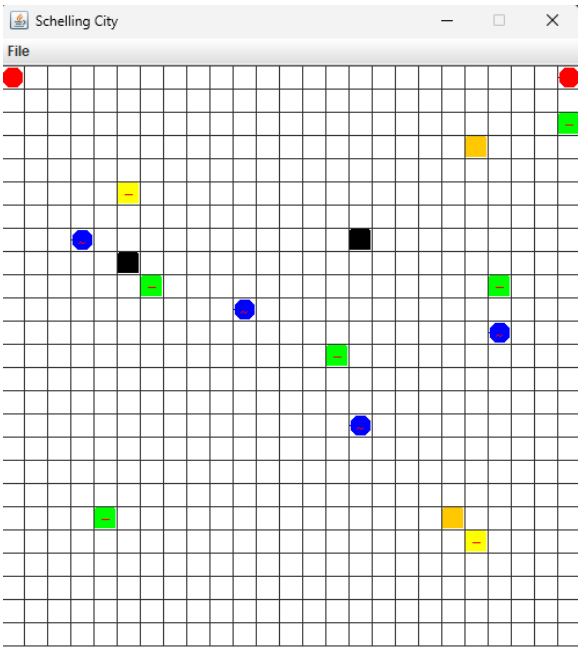


Y finalmente le damos en aceptar:

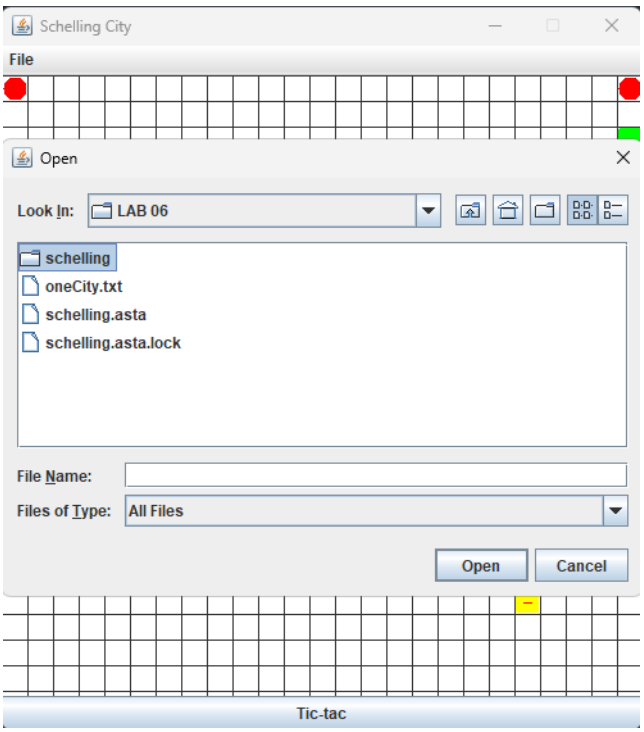


6. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto correcto en oneCity.txt. e importe este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.

   Iniciamos la aplicación:

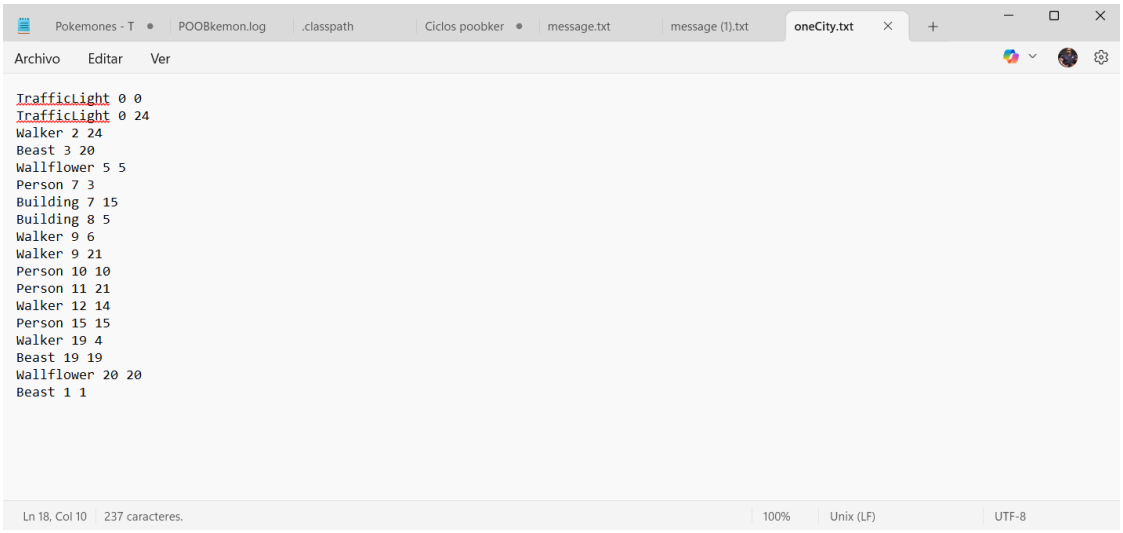


   Le damos en file y le damos en importar:

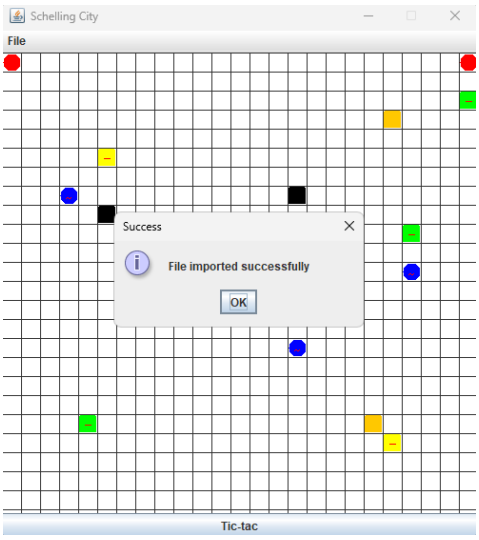


Aqui elegiremos el archivo que queremos importar que en este caso es oneCity.txt

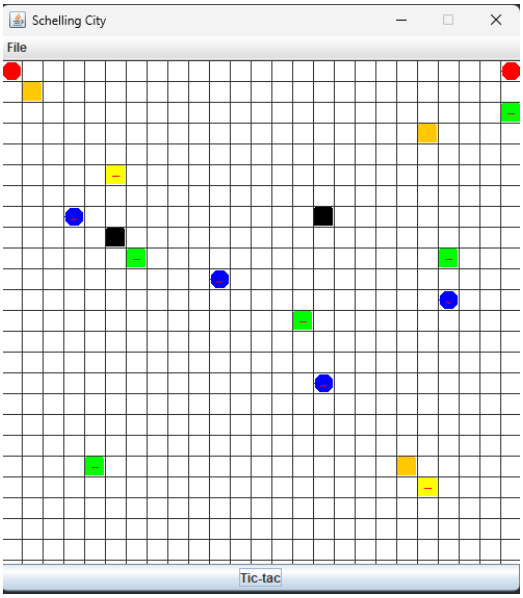
Previamente modificamos el archivo para que se creen nuevos elementos, en este caso se agregó un item de tipo beast en la posicion [1,1]



Le damos en abrir y nos aparece lo siguiente:



Nos aparece que el archivo ha sido importado correctamente, por lo quea al darle en ok se generara el nuevo tablero con los nuevos items



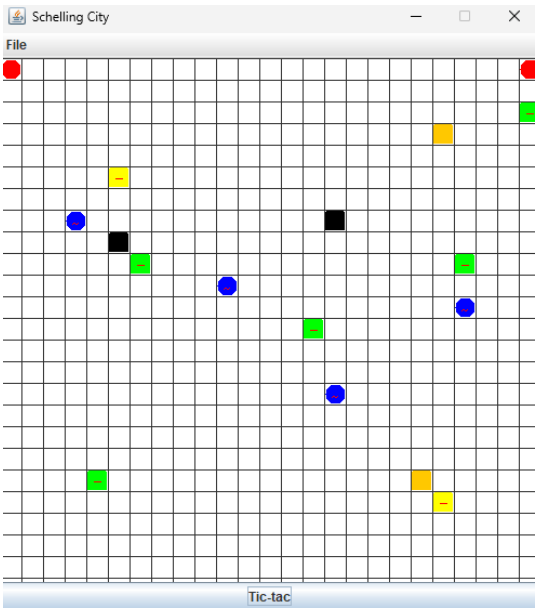
Como se evidencia se ha creado un item de tipo beast en la posicion [1,1].

Analizando comportamiento

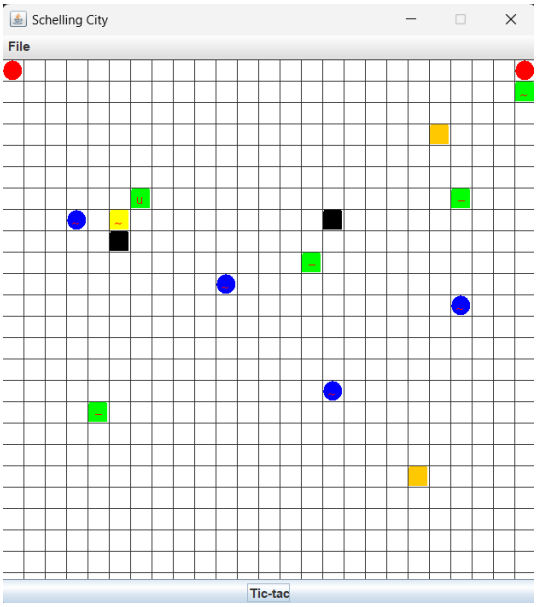
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

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

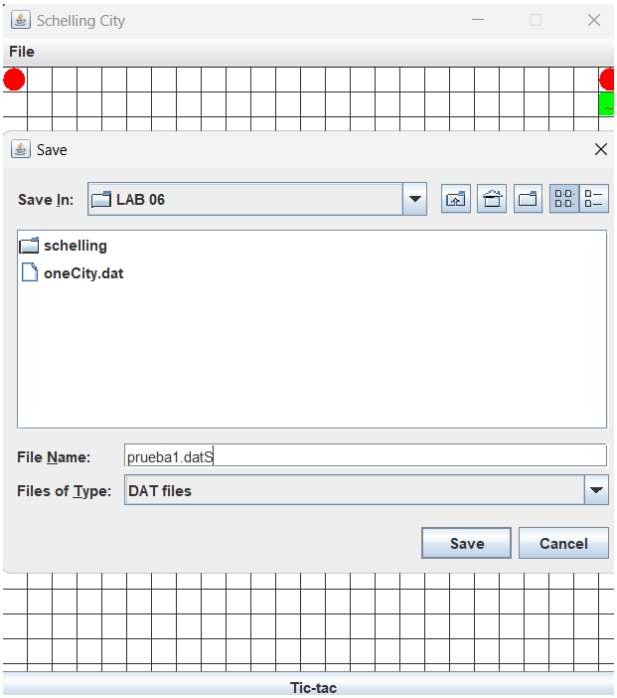
   Ejecutamos la aplicación:



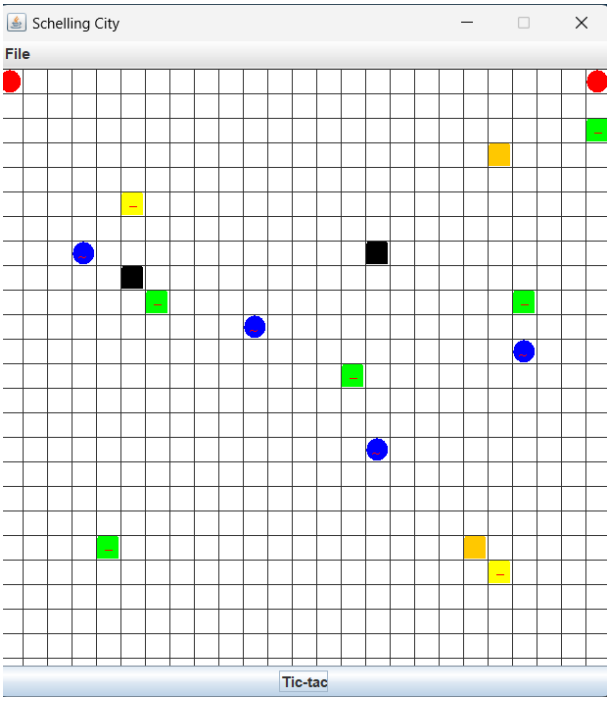
   Le damos 3 veces tic tac:

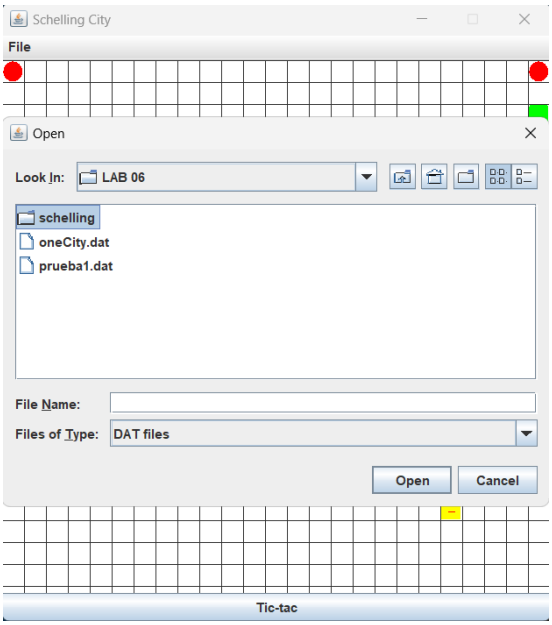


Le damos en File, Save As y lo guardamos como prueba1.dat:

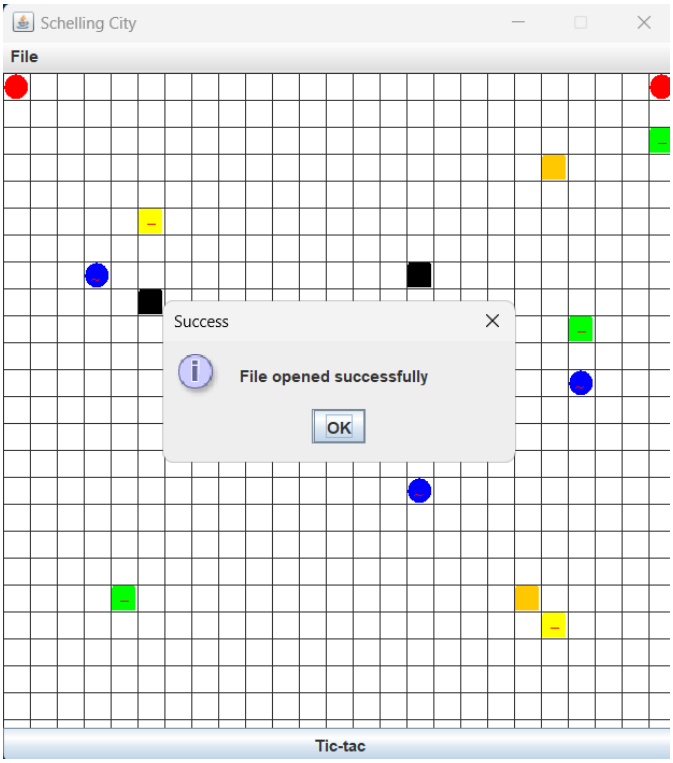


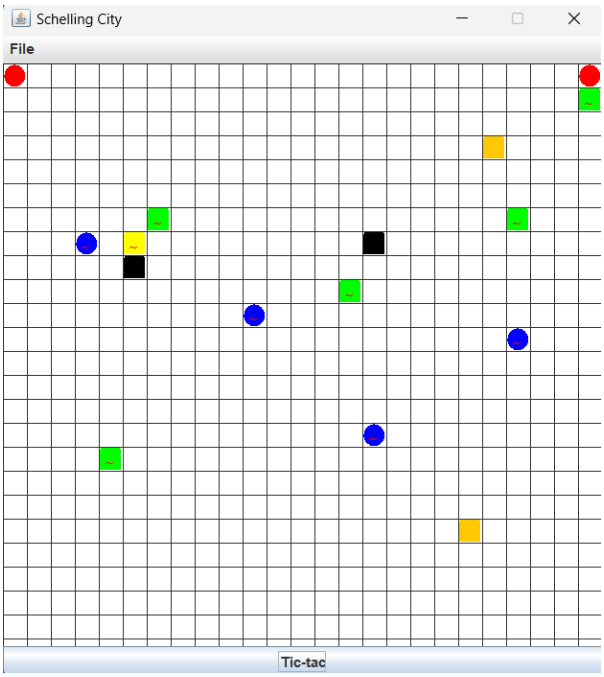
Generamos uno nuevo tablero y abrimos el archivo:





Vemos que el archivo se cargó correctamente y le damos en aceptar:

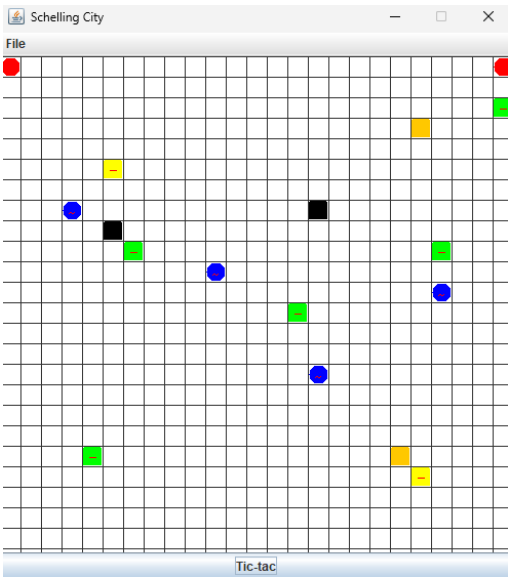




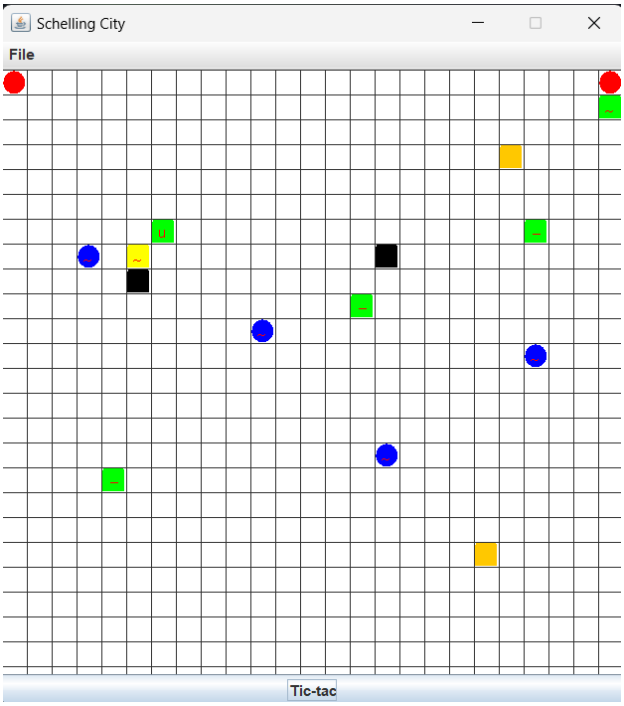
Como se puede evidenciar, lo que hace el save as y el open es guardar la información completa del juego en un archivo y cuando se ejecuta la función open, está por decirlo así borrando el juego actual y creando uno nuevo con la información del archivo que se abrió.

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

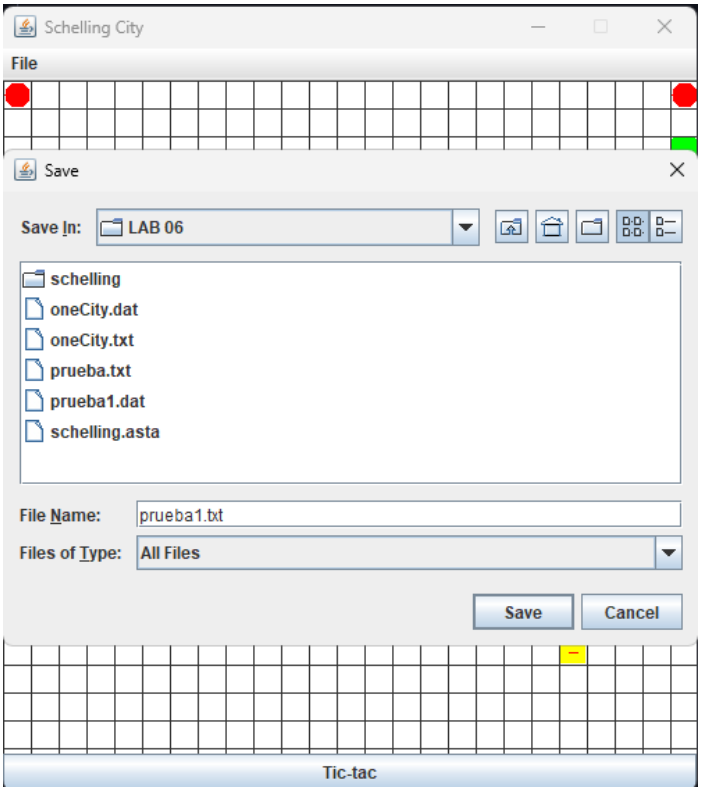
   Ejecutamos la aplicación:

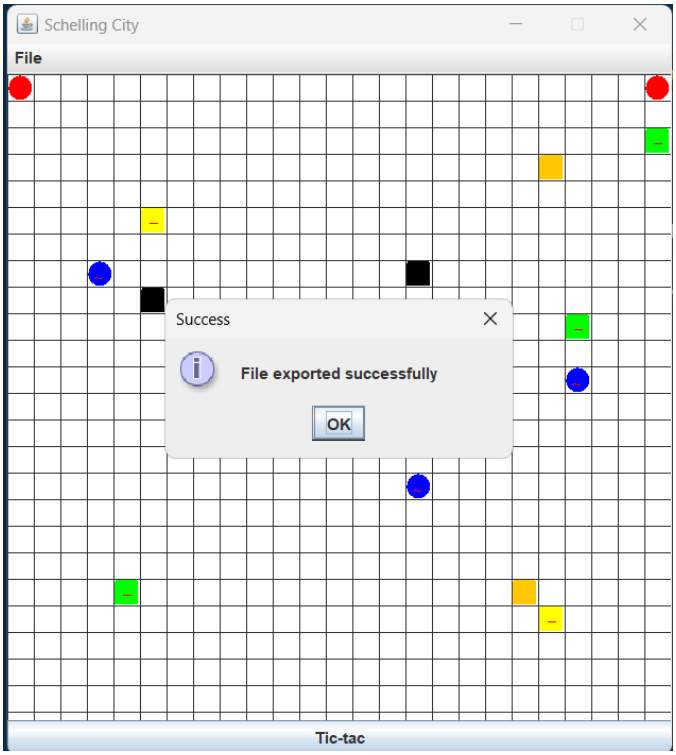


Le damos 3 veces tic tac:

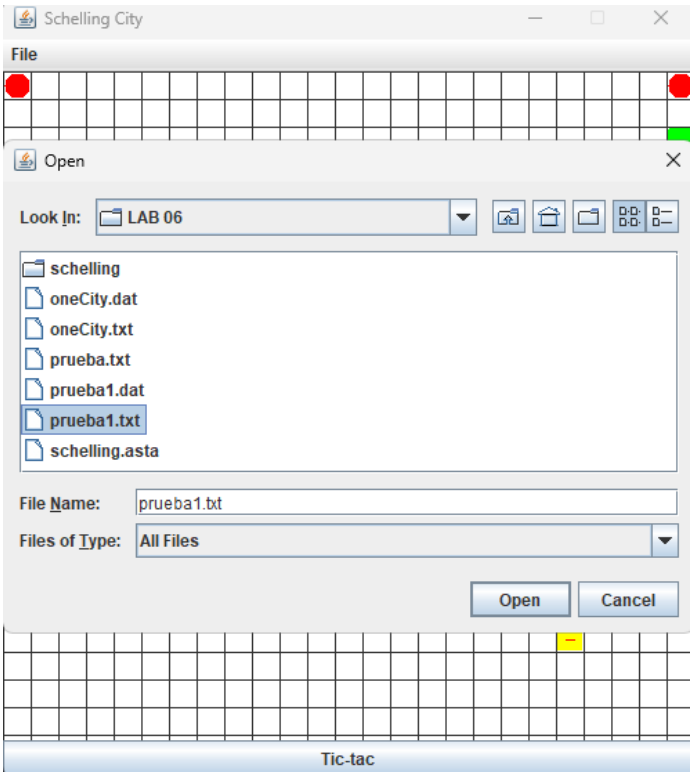


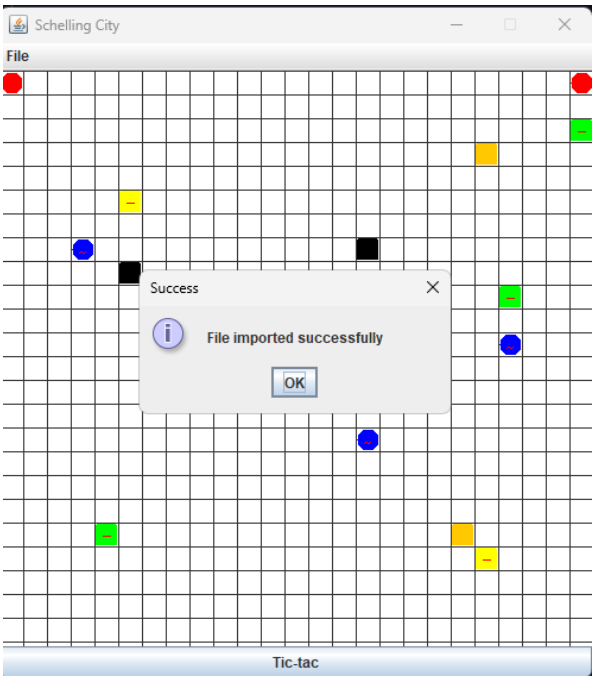
Le damos en file y luego en export as y guardamos un archivo .txt

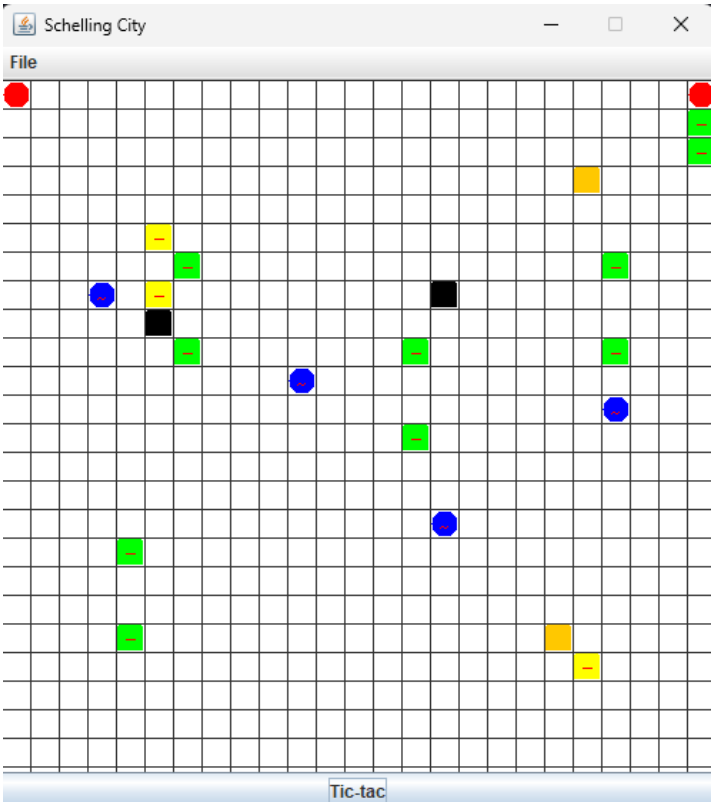




Creamos un nuevo juego en file y luego new e importamos el archivo que acabamos de exportar


Schelling City
File
Open
Look In:
LAB 06
schelling
oneCity.dat
oneCity.txt
prueba.txt
prueba1.dat
prueba1.txt
schelling.asta
File Name:
prueba1.txt
Files of Type:
All Files
Open
Cancel
Tic-tac


Schelling City
File
Success
File imported successfully
OK
Tic-tac



Lo que hace el export as es generar un archivo donde se guardan los items y sus posiciones en el tablero, al importar este archivo a un tablero lo que se esta haciendo es añadir los items en las posiciones establecidas en el archivo junto con los items que esten en el tablero antes de importarlo.

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

A principal diferencia de comportamientos entre el save as, open y export as e import es que el save as lo que hace es guardar un juego por decirlo así, independientemente si hay items en el tablero actual, si abrimos un archivo, se reiniciara y se colocaran el tablero con los items del archivo, en cambio en el import lo que se hace es que se añaden los objetos al tablero actual sin borrar lo que haya actualmente en el tablero.

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

Revisar schelling/src/domain/City.java

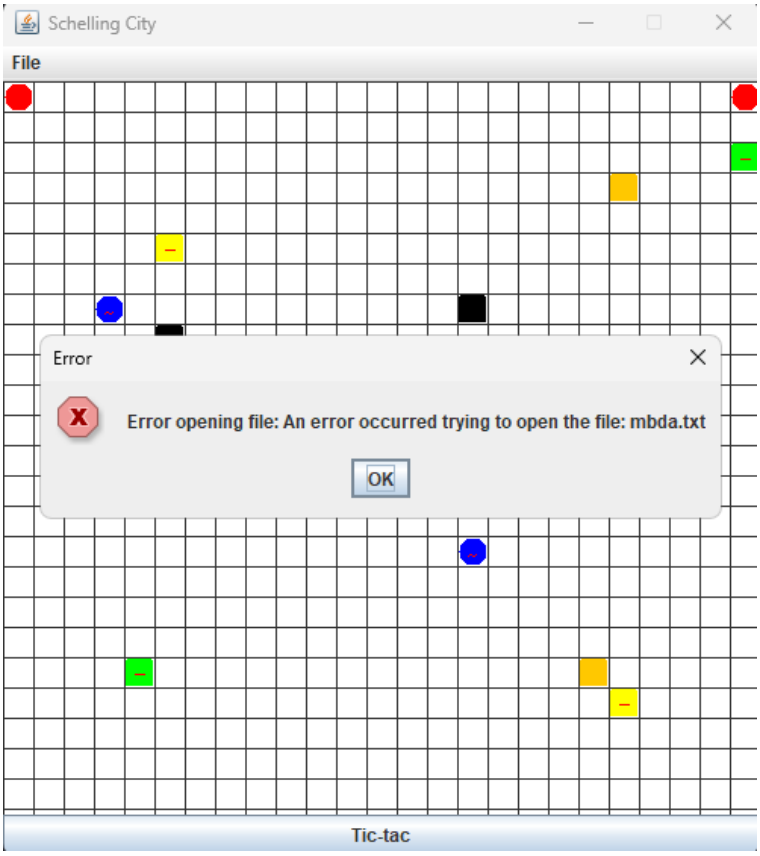
2. Perfeccionen el manejo de excepciones de los métodos open y save detallando los errores. No olviden pruebas de unidad.

   Revisar schelling/src/domain/City.java
   Revisar schelling/src/presentation/CityGUI.java

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

Para esta prueba intentaremos abrir un archivo con un formato distinto al .dat:

Primero ejecutamos el programa y luego intentamos abrir un archivo con un formato distinto como se verá a continuación:



Perfeccionando importar y exportar.

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

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

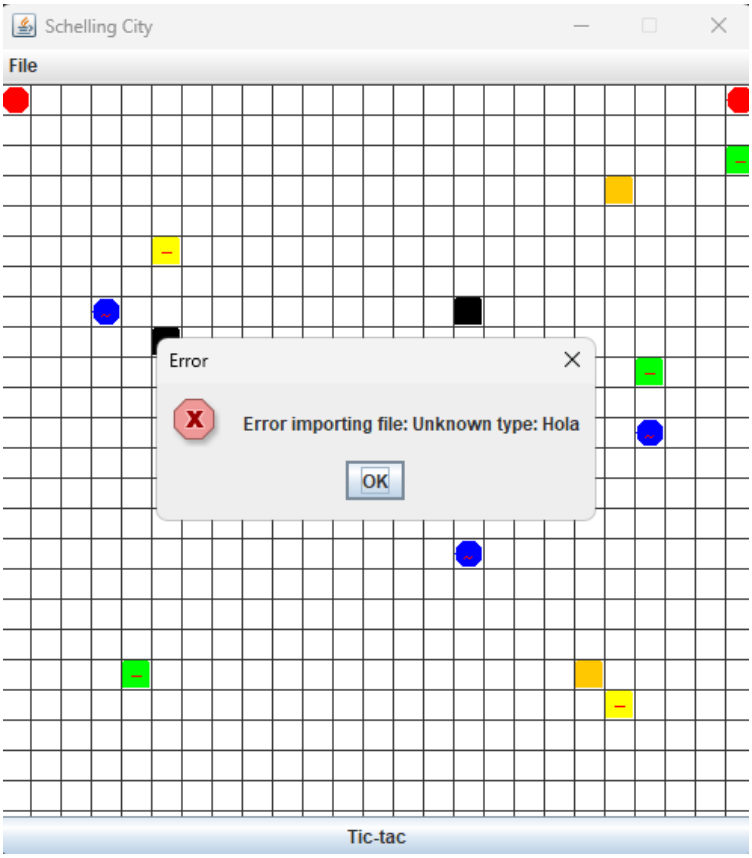
   Revisar schelling/src/domain/City.java

2. Perfeccionen el manejo de excepciones de los métodos import y export detallando los errores. No olviden pruebas de unidad.

   Revisar schelling/src/domain/City.java
   Revisar schelling/src/presentation/CityGUI.java

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

Para esta prueba de aceptación se importa un archivo .txt con la información de los items, pero se agrega un item Hola que no existe:



Perfeccionando importar. Hacia un minicompilador.

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

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

   Revisar schelling/src/domain/City.java

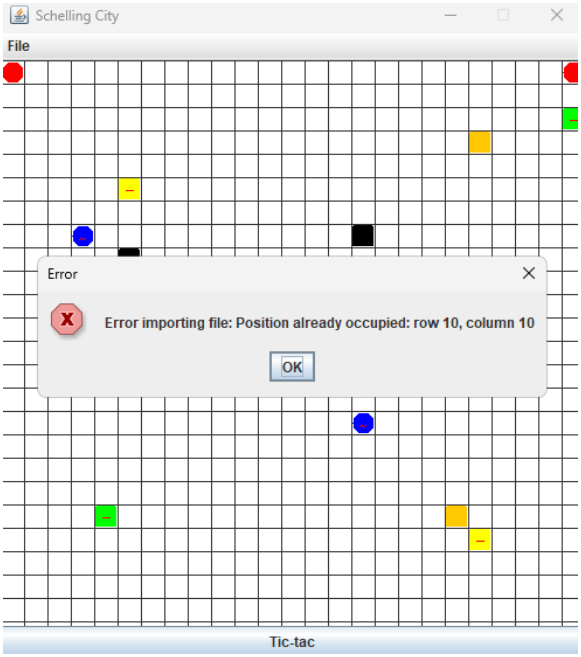
2. Perfeccionen el método import para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palabra que tiene el error y causa de error.

   Revisar schelling/src/domain/City.java

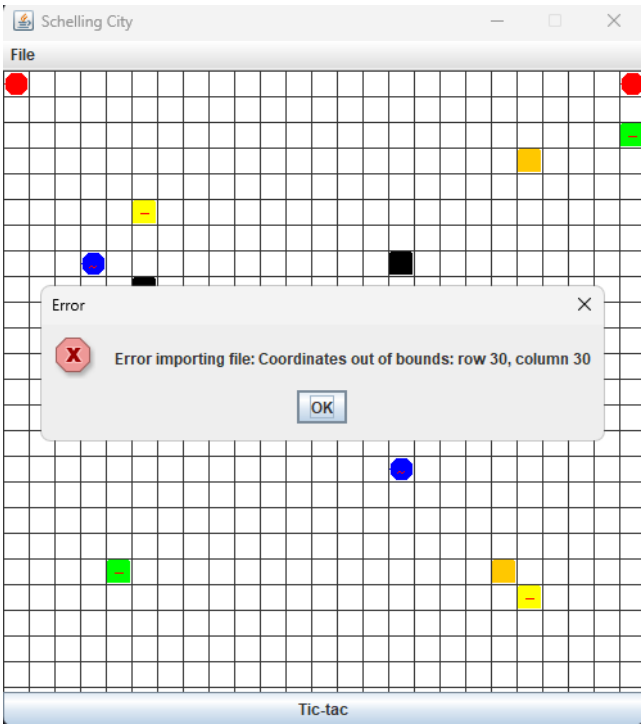
3. Escriban otro archivo con errores, llámelo cityErr.txt, para ir arreglándolo con ayuda de su “importador”. Presente las pantallas que contengan los errores.

Se creo un archivo cityErr.txt el cual la primera vez que lo importamos aparecera el siguiente error:

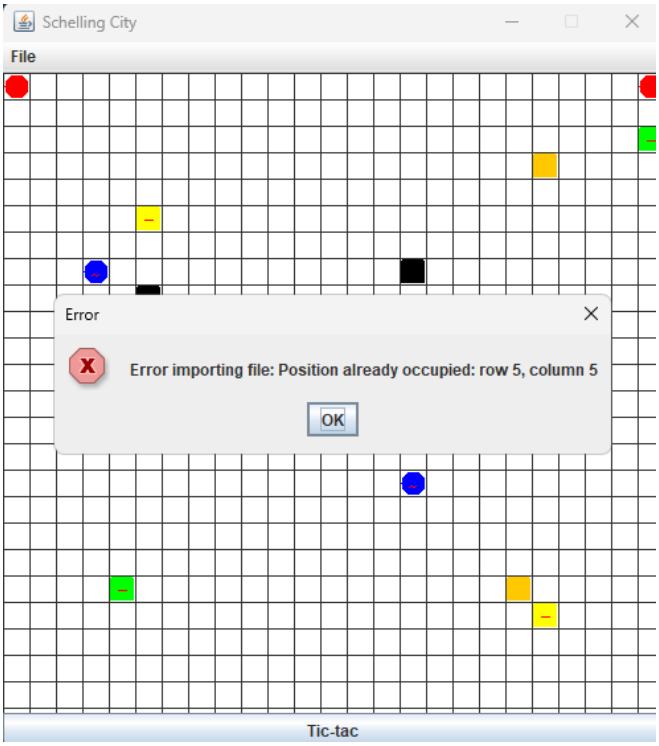
Lo ubicamos en otro lado y lo volvemos a importar:



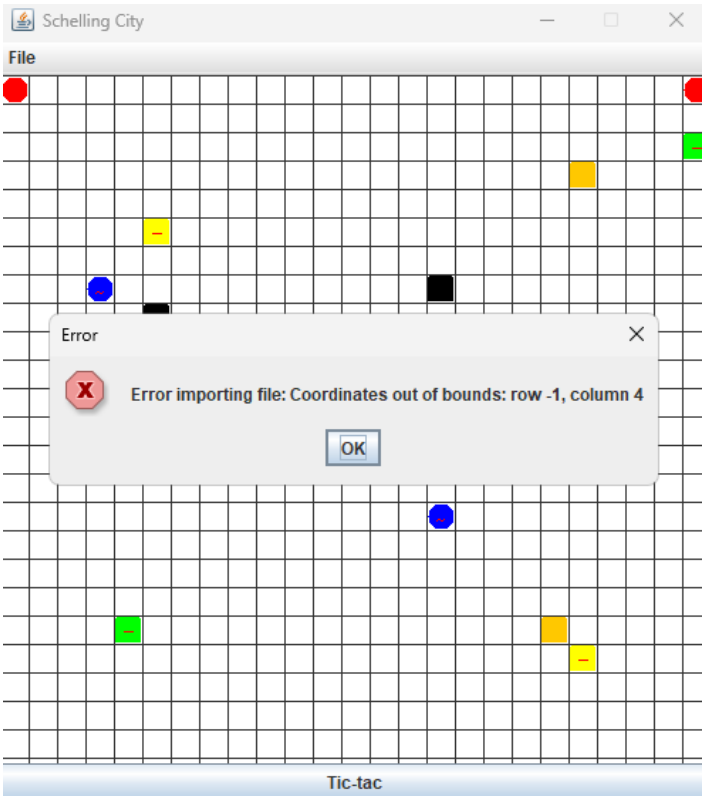
Aqui el error se da porque se sale de los limites:



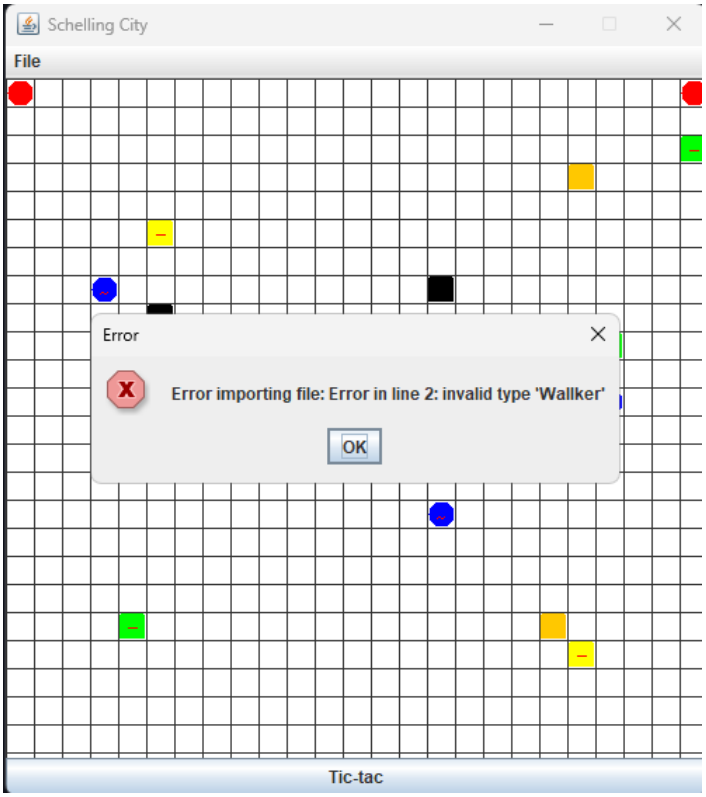
Aqui el error da porque ya existe un item ocupando esta posicion:



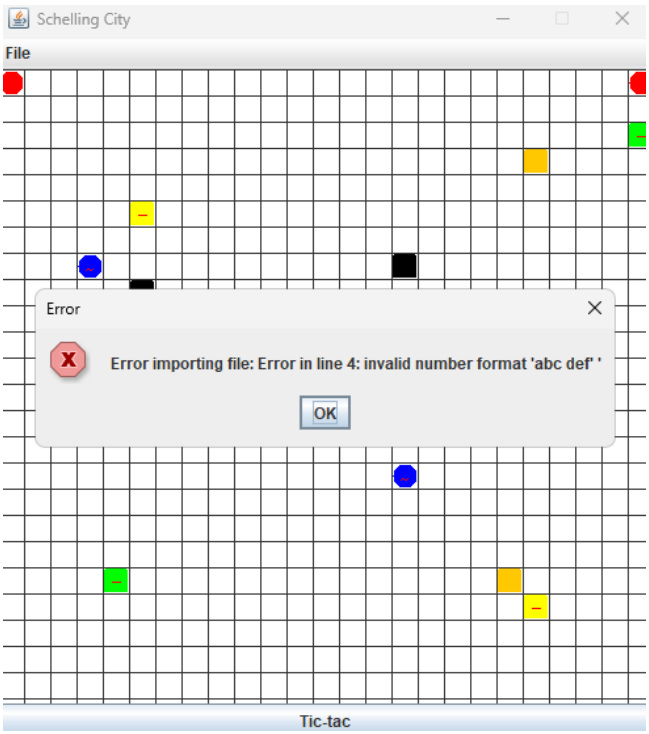
Coordenadas fuera de los limites:



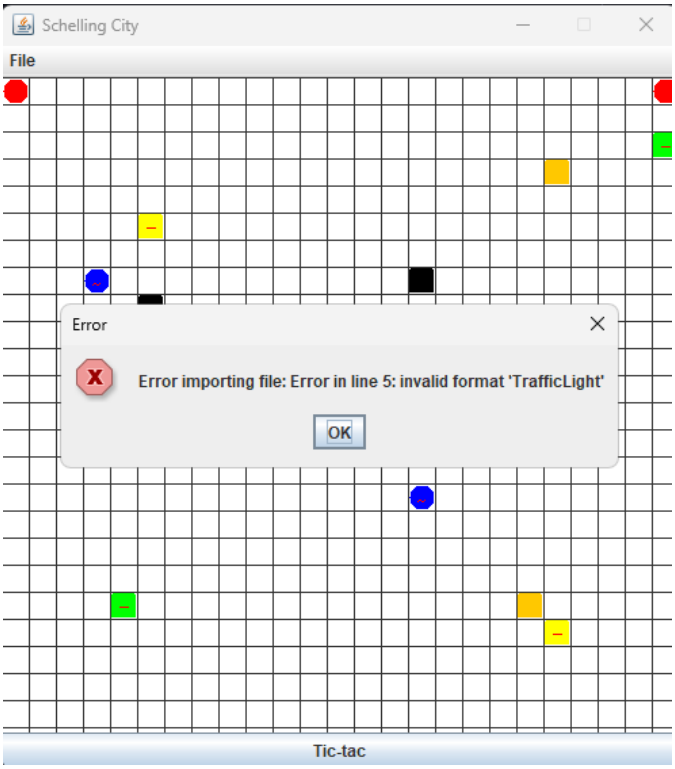
Tipo invalido:



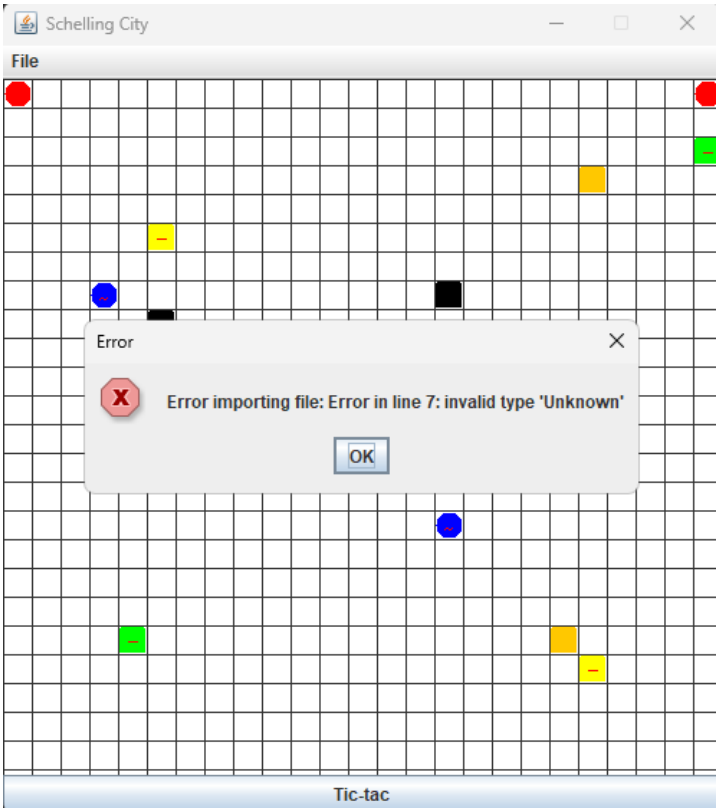
Formato invalido:



Formato invalido ya que no hay coordenadas:



Tipo invalido, no existe “Unknown”:



BONO. Perfeccionando importar. Hacia un minicompilador flexible.

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

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

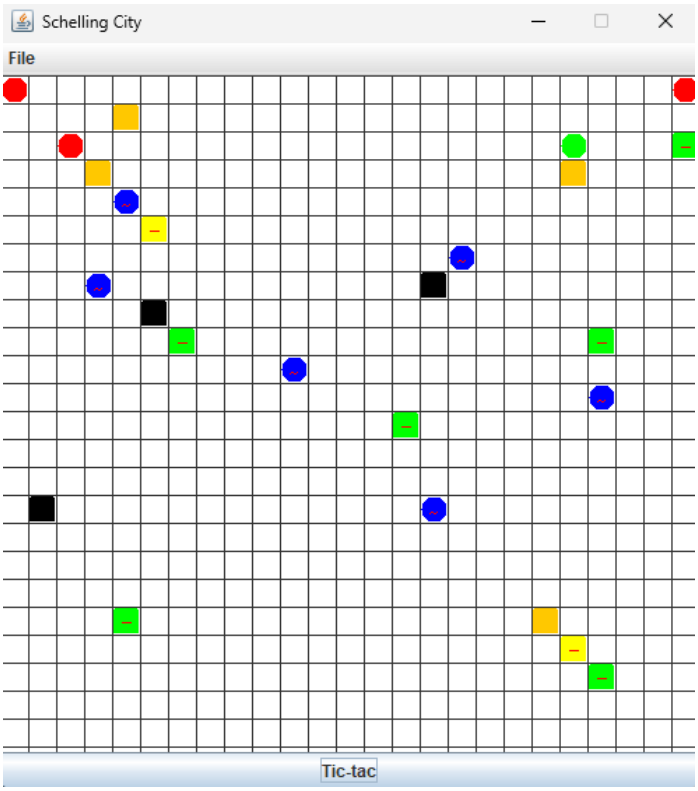
Revisar schelling/src/domain/City.java

2. Perfeccionen los métodos import y export para que pueda servir para cualquier tipo de elementos creados en el futuro. No olviden pruebas de unidad. (Investiguen cómo crear un objeto de una clase dado su nombre)

Revisar schelling/src/domain/City.java

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

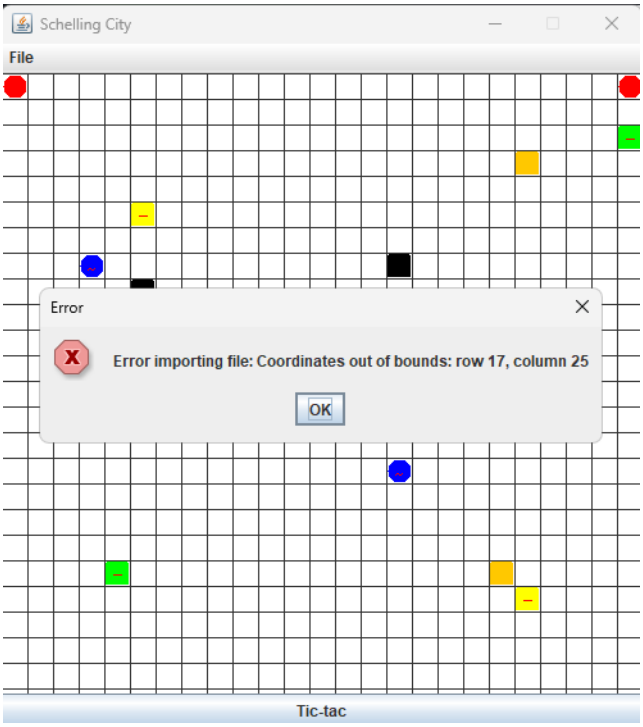
Para probar su flexibilidad crearemos un nuevo item que se llamara tree, no tiene función por lo que solo será para probar la flexibilidad, a continuación, agregaremos al archivo de pruebas el tree:



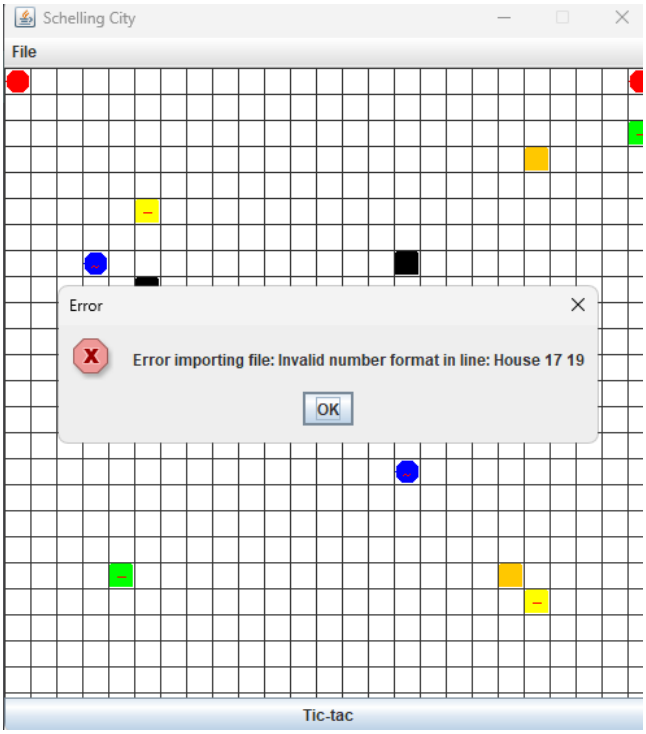
Como se puede ver, se agregó la clase Tree y en el txt también se agregó, cuando lo importamos, se agrega exitosamente.

Pruebas significativas:

Coordenadas fuera de los limites:



Formato invalido:



RETROSPECTIVA

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

?(Horas/Hombre)

El tiempo invertido por hombre fue de 20 horas aproximadamente

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

El estado actual del laboratorio es completado ya que se realizó cada uno de los puntos asignados en su totalidad y consideramos que no quedo faltando nada en ninguno.

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

La más útil fue la de property ownership, nos ayudó a cambiar la mentalidad de como llevábamos manejando los laboratorios y empezamos a preocuparnos más por la calidad de entrega del trabajo que por la parte que nos tocaba a cada uno

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

El mayor logro de este laboratorio fue aprender el manejo de archivos ya que a nuestro parecer es bastante importante ya que esto lo vemos implementando en muchos lugares y nos parece bastante útil haberlo aprendido.

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

El mayor problema técnico quizás fue la construccion de los metodos importar y exportar que en un inicio se nos complicó bastante debido a que no la conociamos pero pues investigando e intentando pudimos perfeccionarlo.

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

Resultados?

Como equipo, siempre tratamos de dividirnos el trabajo de manera equitativa donde podamos aprender los dos de manera igualada, también solemos trabajar en pair programming lo que nos ayuda a tener un mejor rendimiento a la hora de hacer el código, nos comprometemos a seguir mejorando la calidad de los laboratorios para que cada vez sea mejor la entrega de estos.

7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares

Adecuados.

La referencia más útil quizás fue la IA ya que esta nos ayudó con algunos errores y con el manejo de algunos errores.

Referencias:

- GitHub & OpenAI. (2024). GitHub Copilot [Servicio de inteligencia artificial]. https://github.com/features/copilot
- Medina, D. (2020, 26 de mayo). Using split() and trim() for data cleaning in JavaScript. Medium. https://medium.com/@davidmedina0907/using-split-and-trim-for-data-cleaning-in-javascript-1167ceb1d4d6
- Andujar. (2015, 18 de septiembre). Crear objeto a partir del nombre de clase con parámetros usando reflexión en Qt. Stack Overflow en español.

https://es.stackoverflow.com/questions/7889/crear-objeto-a-partir-del-nombre-de-clase-con-parámetros-usando-reflexión-en-qt