**PROGRAMACIÓN ORIENTADA A OBJETOS**

**PERSISTENCIA**

**2024-01**

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

# **OBJETIVOS**

1. Completar el código de un proyecto considerando requisitos funcionales.
2. Diseñar y construir los métodos básicos de manejo de archivos: abrir, guardar, importar y exportar.
3. Controlar las excepciones generadas al trabajar con archivos.
4. Experimentar las prácticas XP : Only one pair [integrates code at a time](http://www.extremeprogramming.org/rules/sequential.html).

Use [collective ownership](http://www.extremeprogramming.org/rules/collective.html).

**ENTREGA**

* 1. Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
  2. Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada

en los espacios preparados para tal fin.

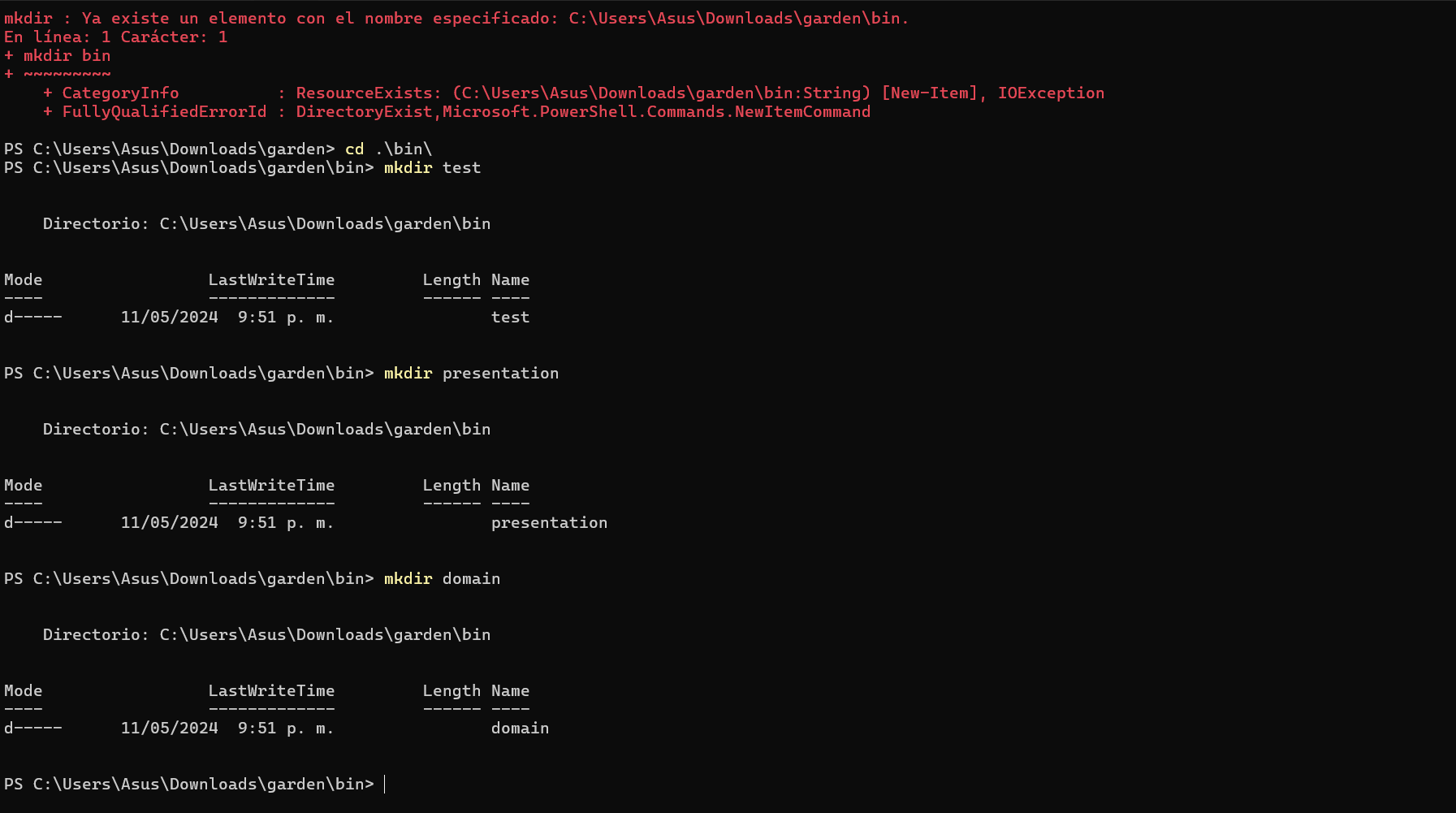
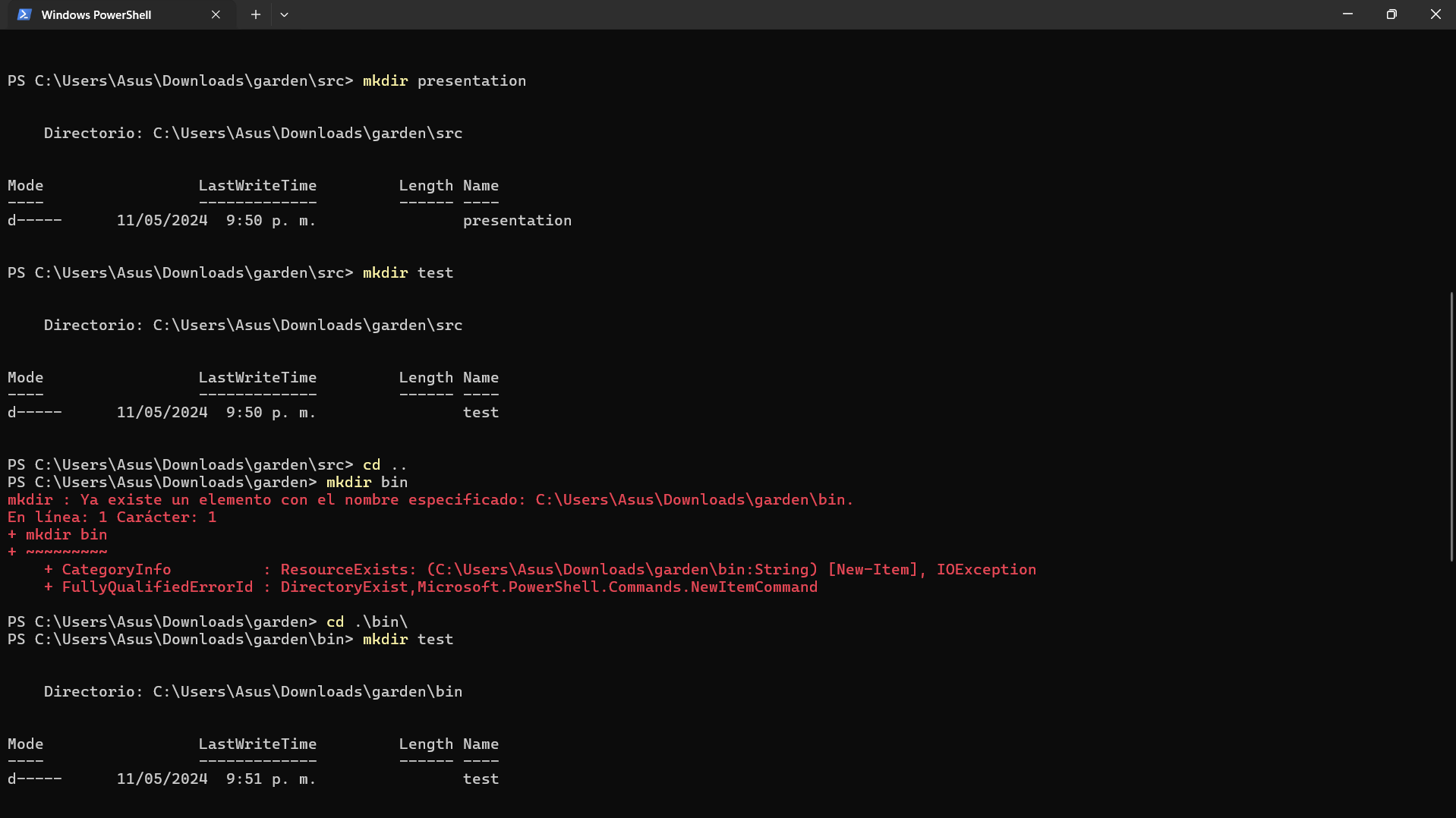
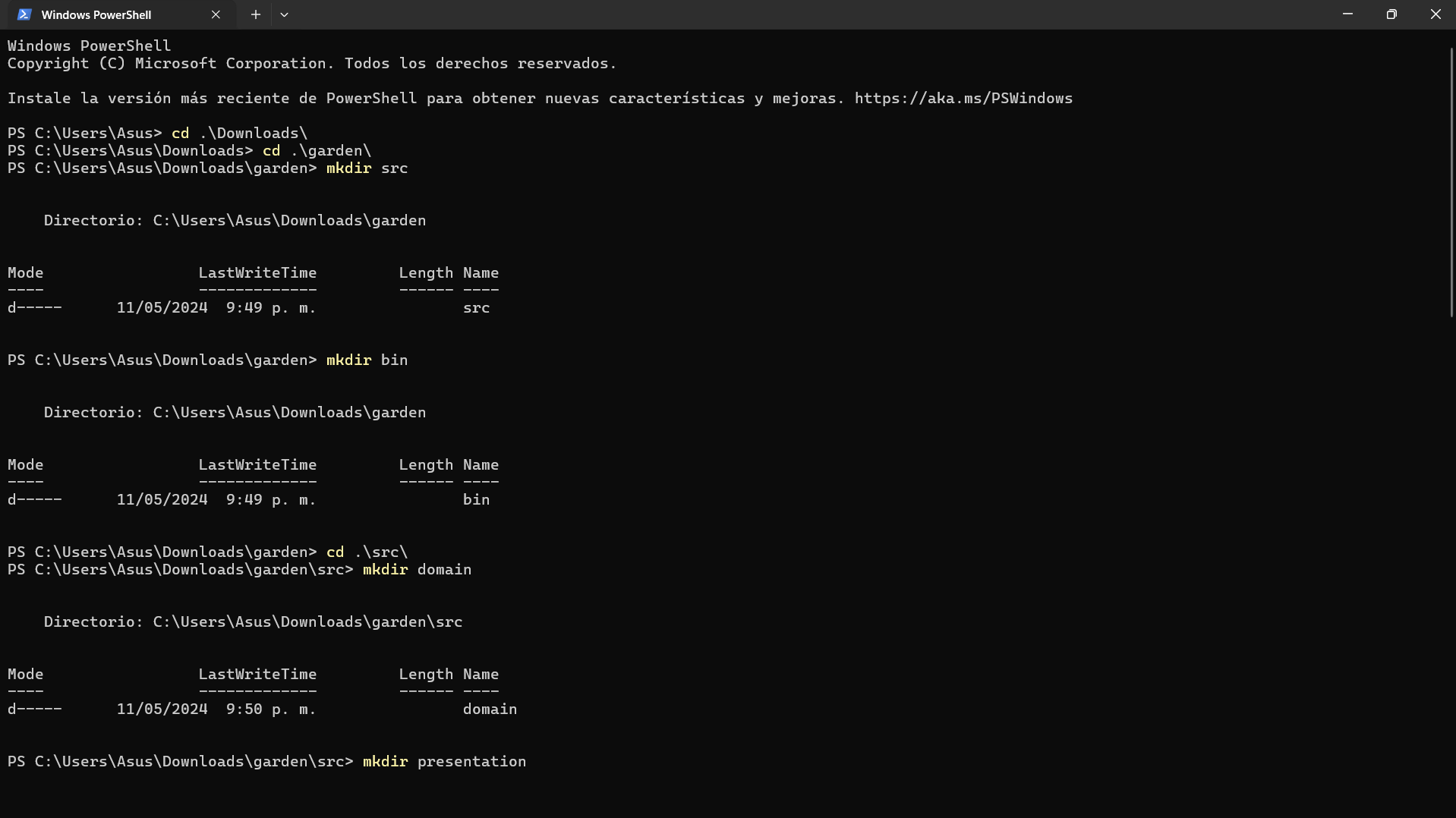
* 1. En el espacio de entrega de avance deben indicar los logros y los problemas pendientes por resolver.

# **DESARROLLO**

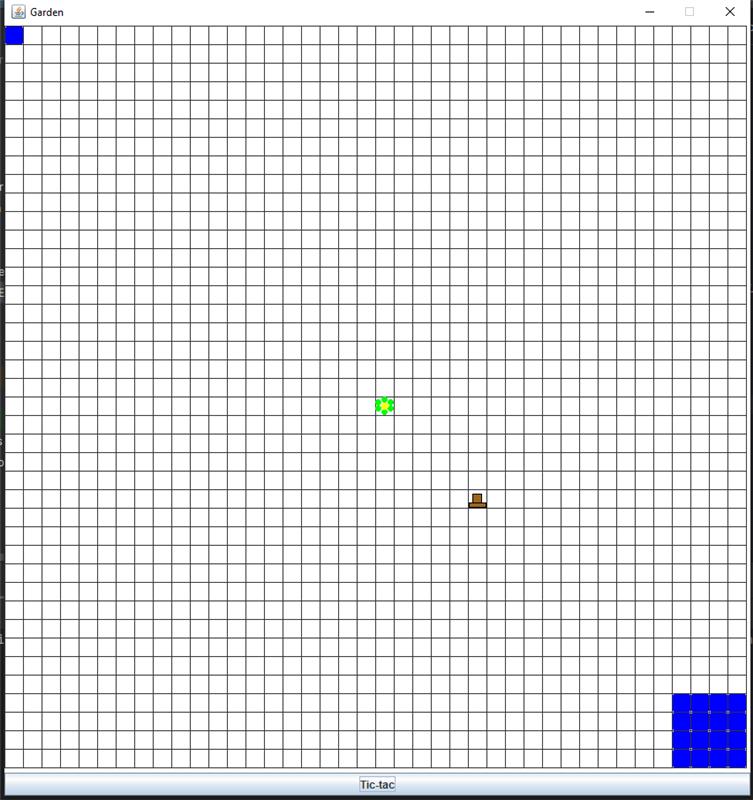
**Preparando**

En este laboratorio vamos a extender el proyecto garden adicionando un menú barra con las optiones básicas de entrada-salida y las optiones 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**



1. Ejecuten el programa, revisen la funcionalidad.

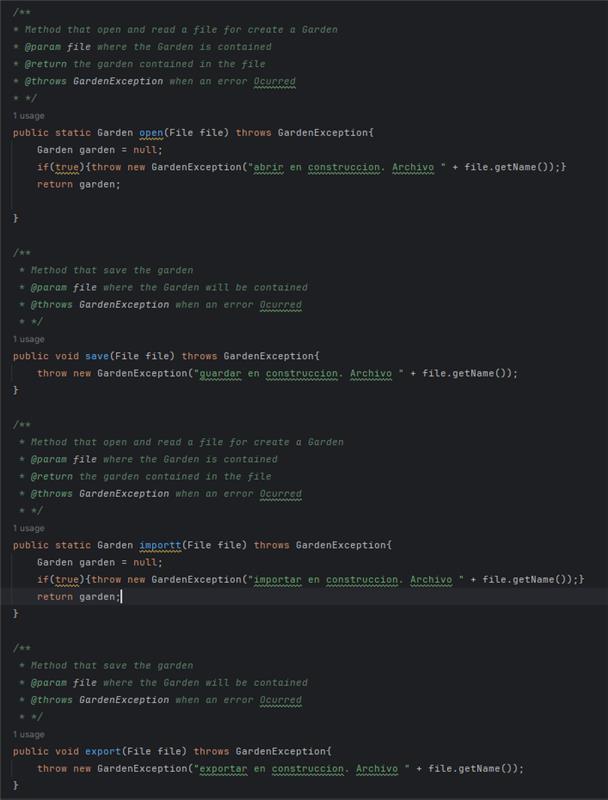


# **Creando la maqueta**

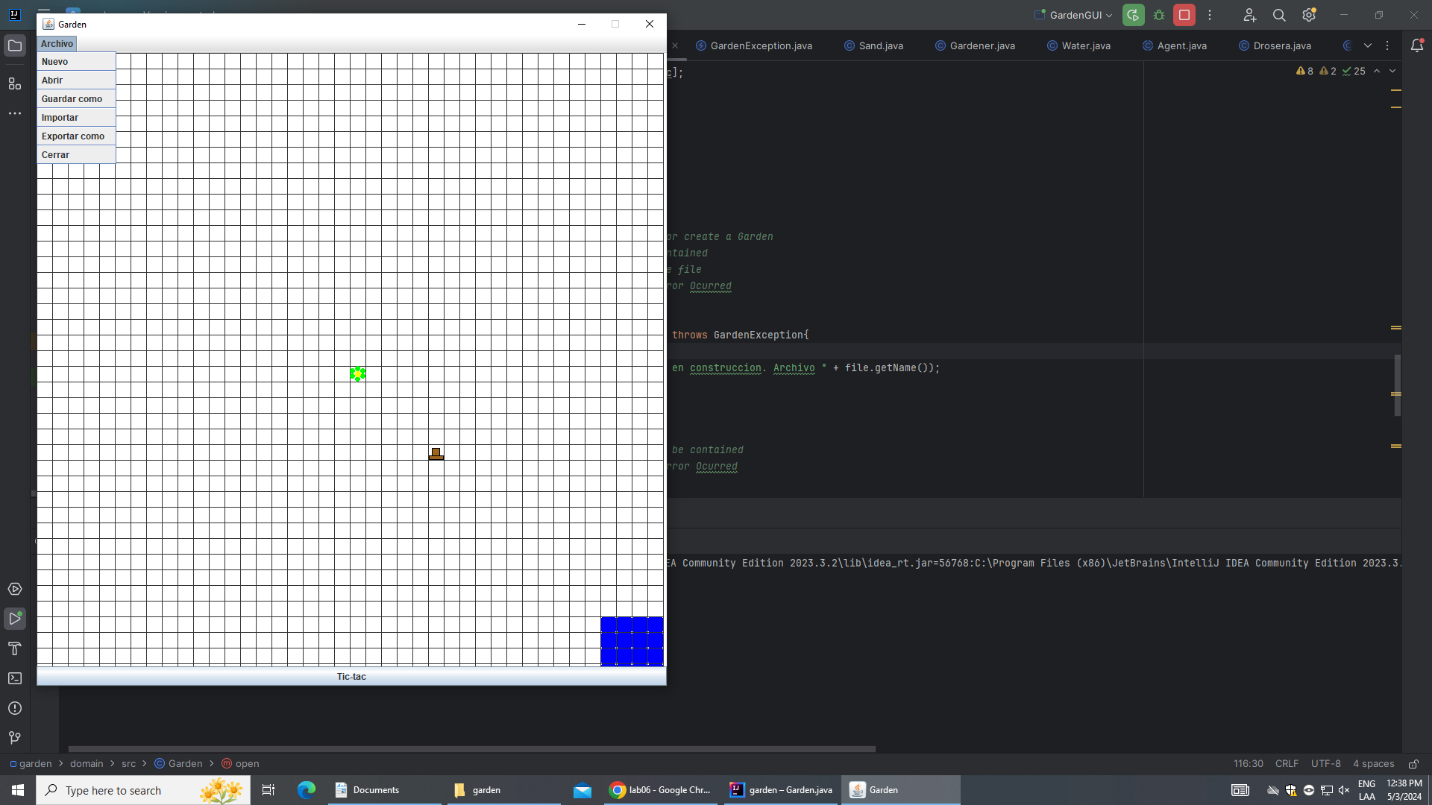
**[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]**

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 optiones básicas de entrada-salida (open, save, import y export). Los métodos deben simplemente propagar una gardenException con el mensaje de “Opción nombreOpción en construcción. Archivo nombreArchivo”. Los métodos deben tener un parámetro File.

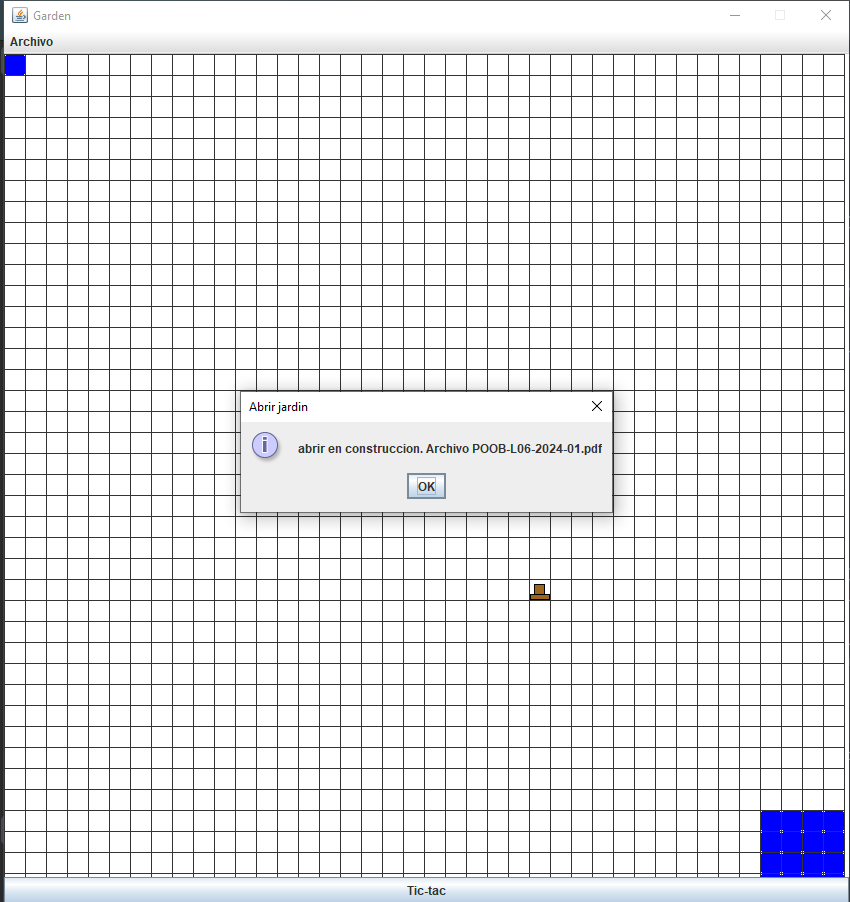


1. **VISTA :** Construyan un menú barra que ofrezca, además de las optiones básicas de entrada-salida, las optiones estándar de nuevo y salir (Nuevo, Abrir, Guardar como, Importar, Exportar como, Salir). No olviden incluir los separadores. Para esto creen el método prepareElementsMenu. Capturen la pantalla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

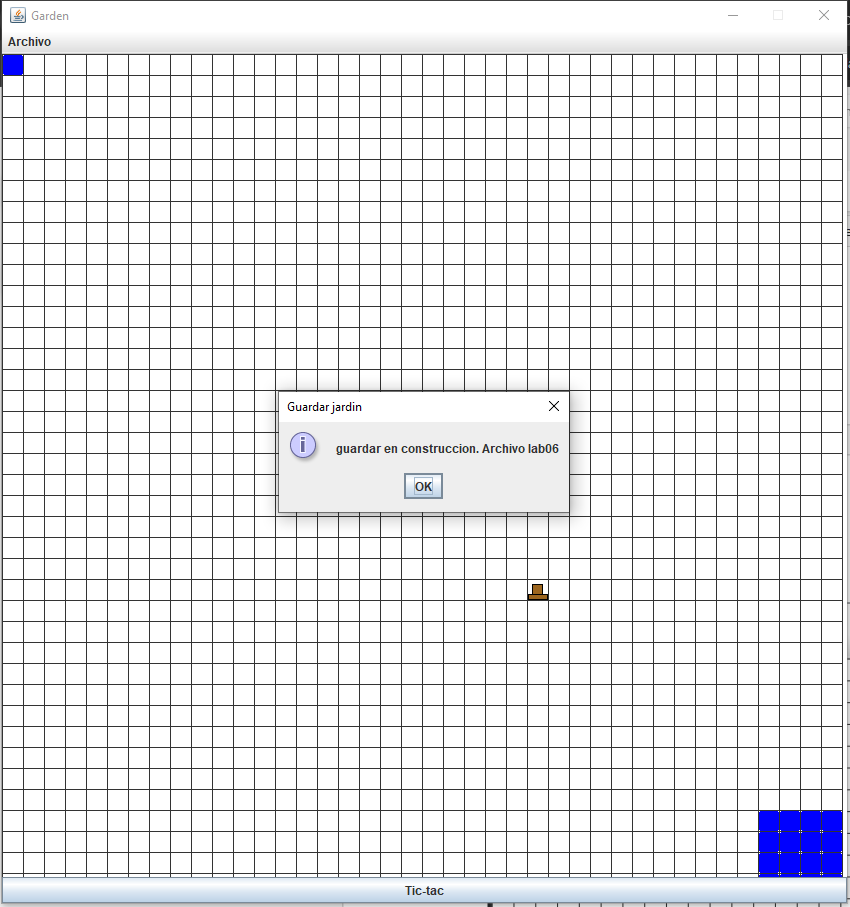


1. **CONTROLADOR:** Construyan los oyentes orrespondientes a las seis optiones. Para esto creen el método prepareActionsMenu y los métodos base del controlador (optionOpen, optionSave, optionImport, optionExport, optionNew, optionExit), En las optiones que lo requieran usen un FileChooser y atiendan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

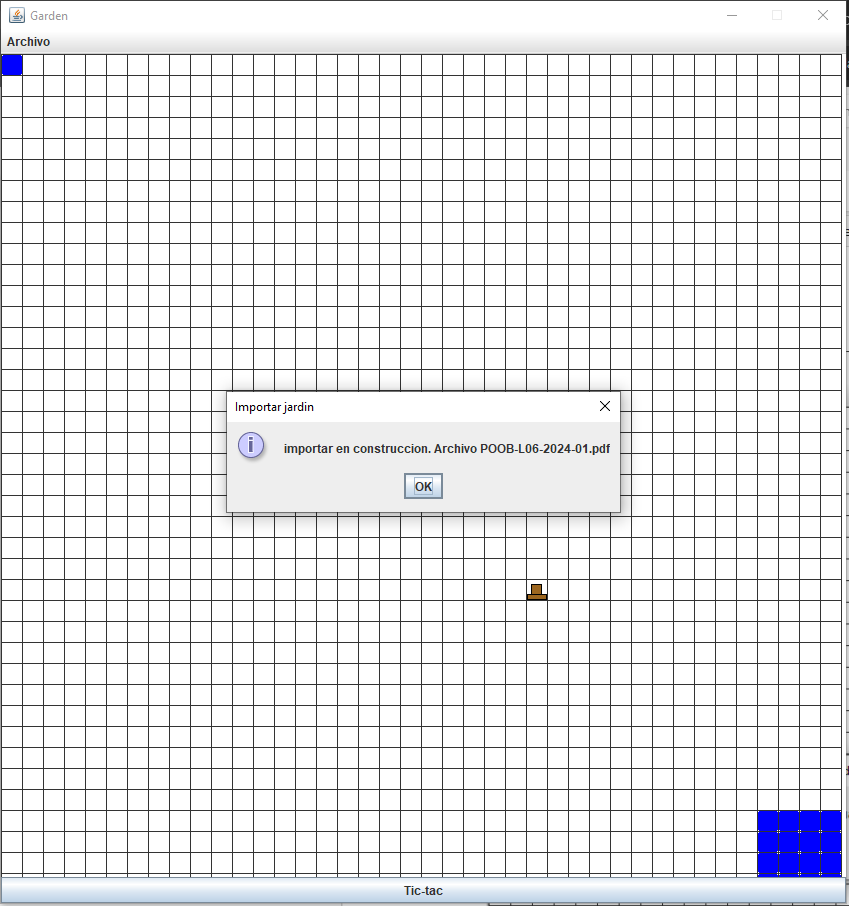
Abrir:



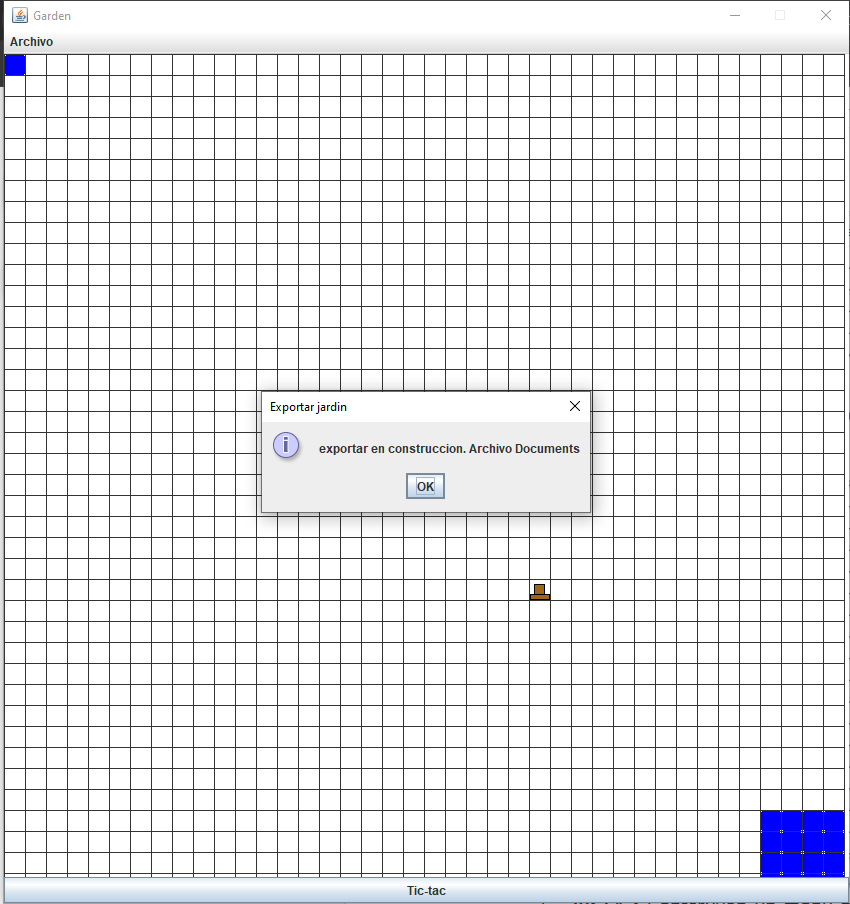
Guardar:



Importar:



Exportar:

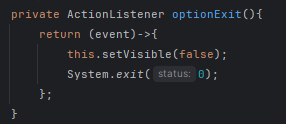


# **Implementando salir y nuevo**

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

Las optiones 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.



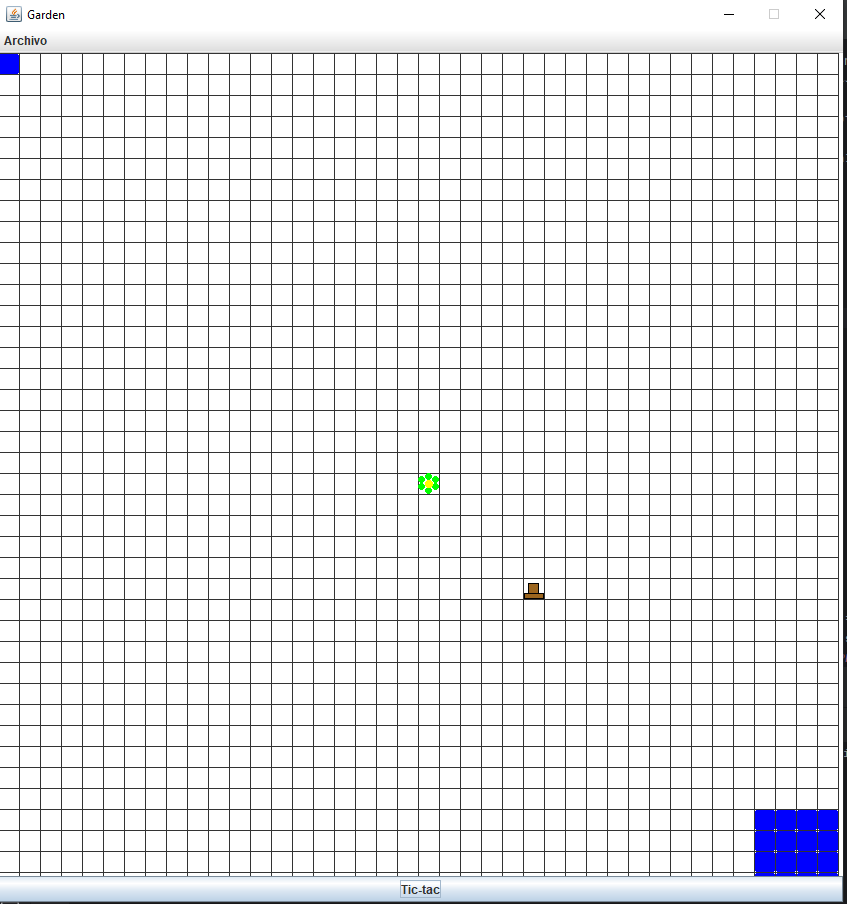


1. Construyan el método optionNew que crea un nuevo garden. Capturen una pantalla significativa.

Nuevo (antes):



(después)

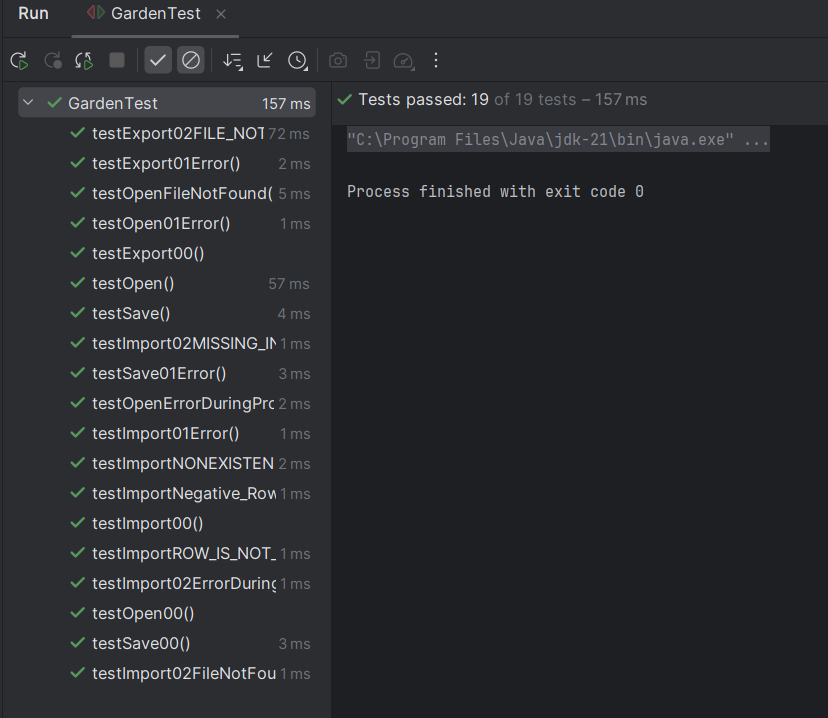


# **Implementando salvar y abrir**

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

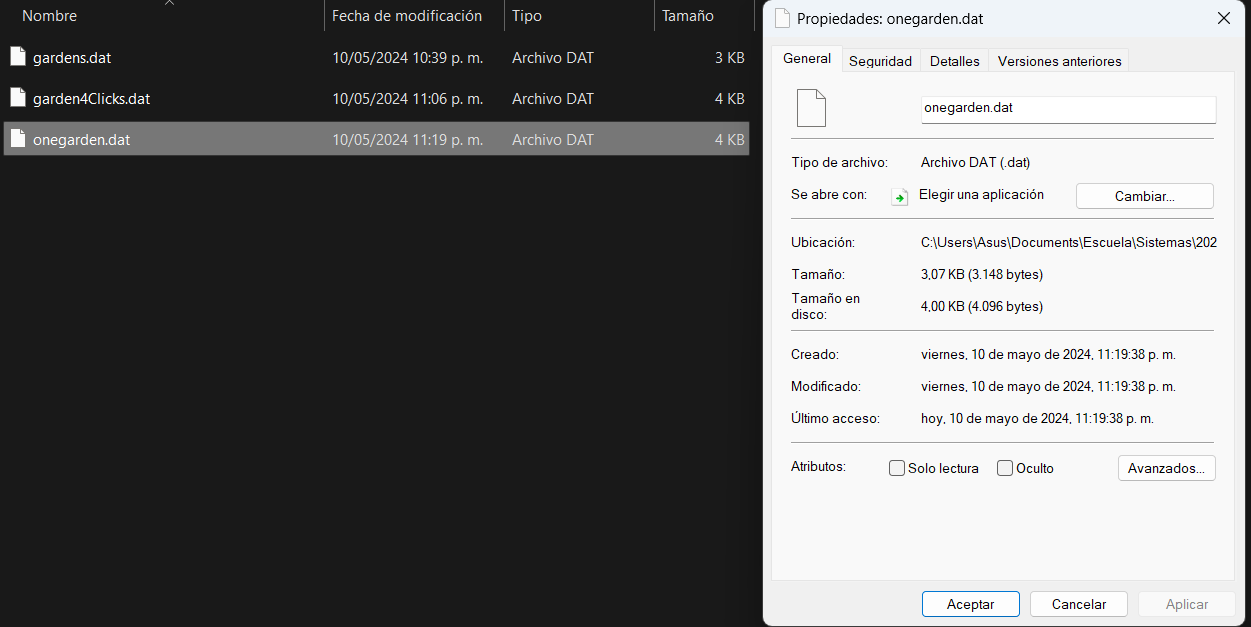
Las optiones salvar y abrir van a ofrecer servicios de persistencia del garden como objeto. Los nombres de los archivos deben tener como extensión .dat.

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 garden. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.

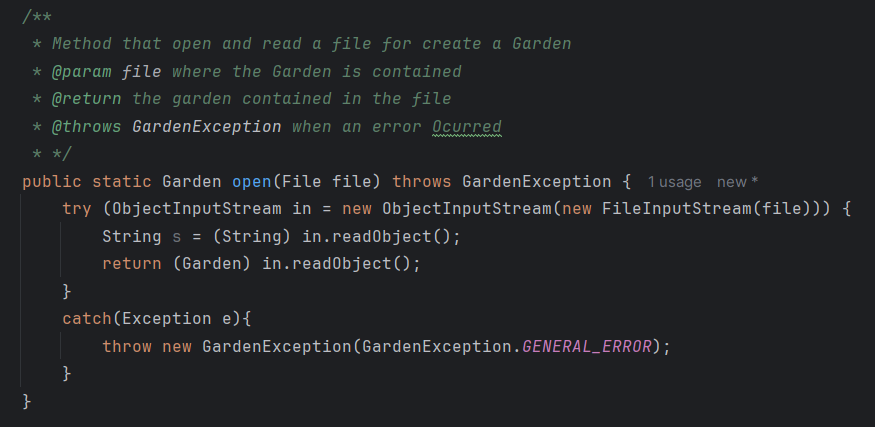


1. Validen este método guardando el estado obtenido después de dos clics como

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

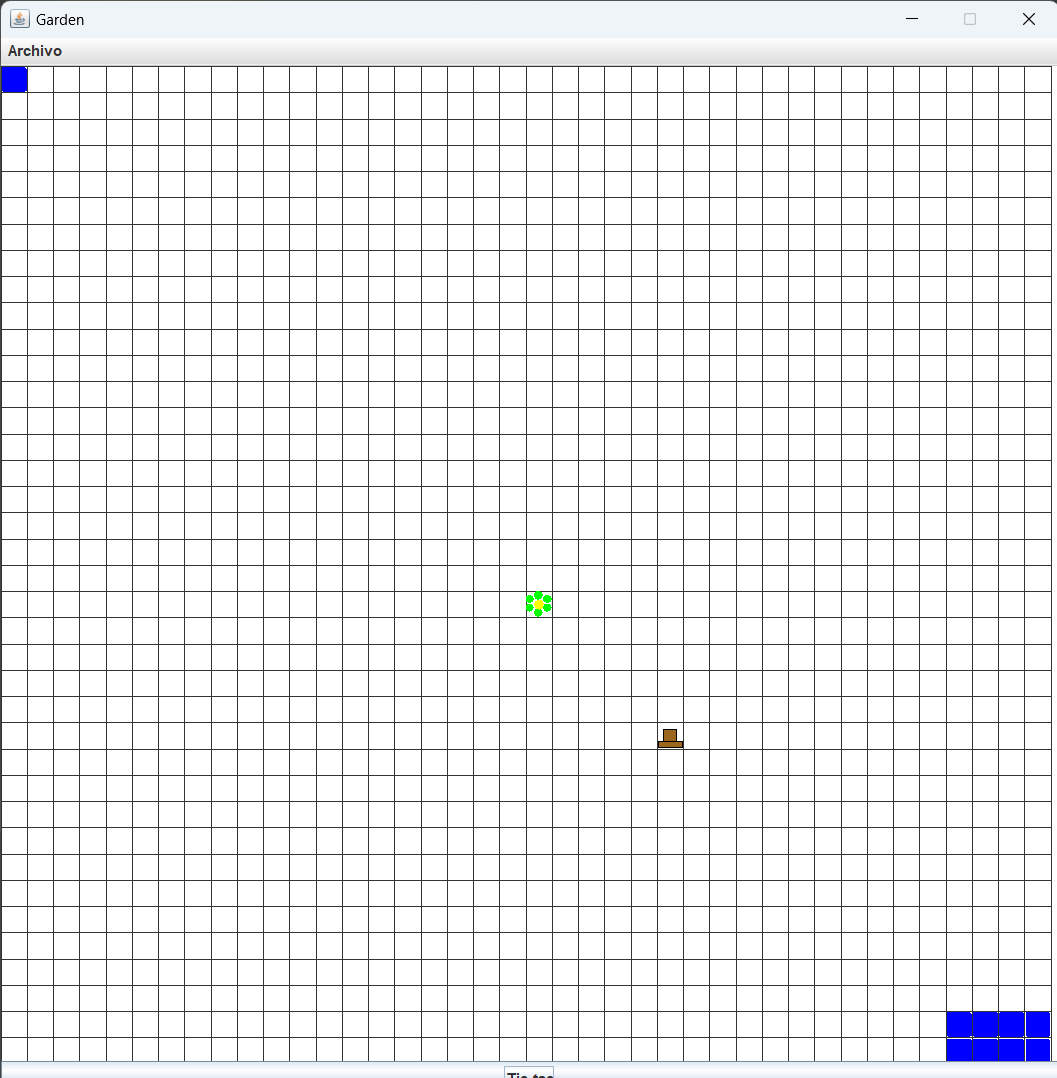
Sí se creó en el disco. El archivo pesa 3KB.

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

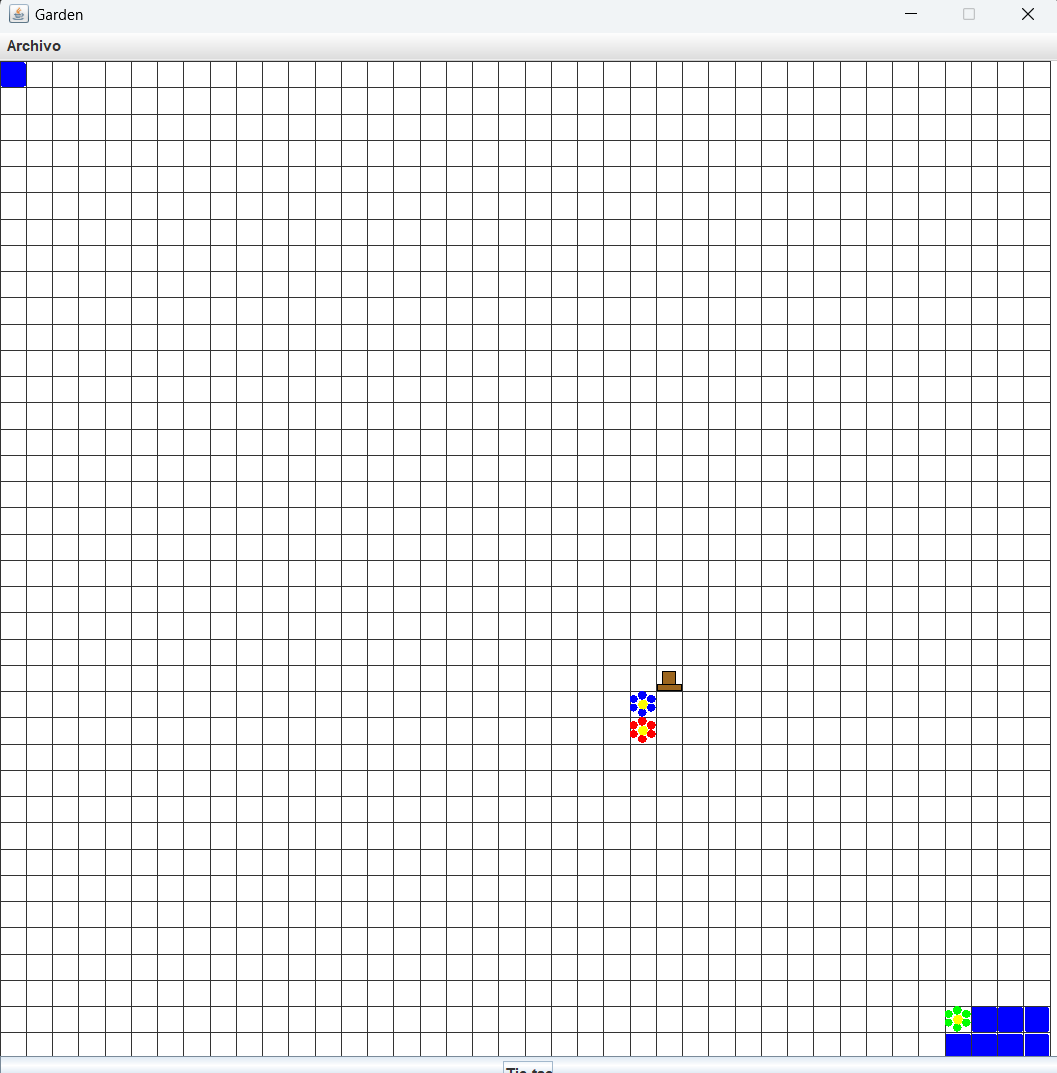
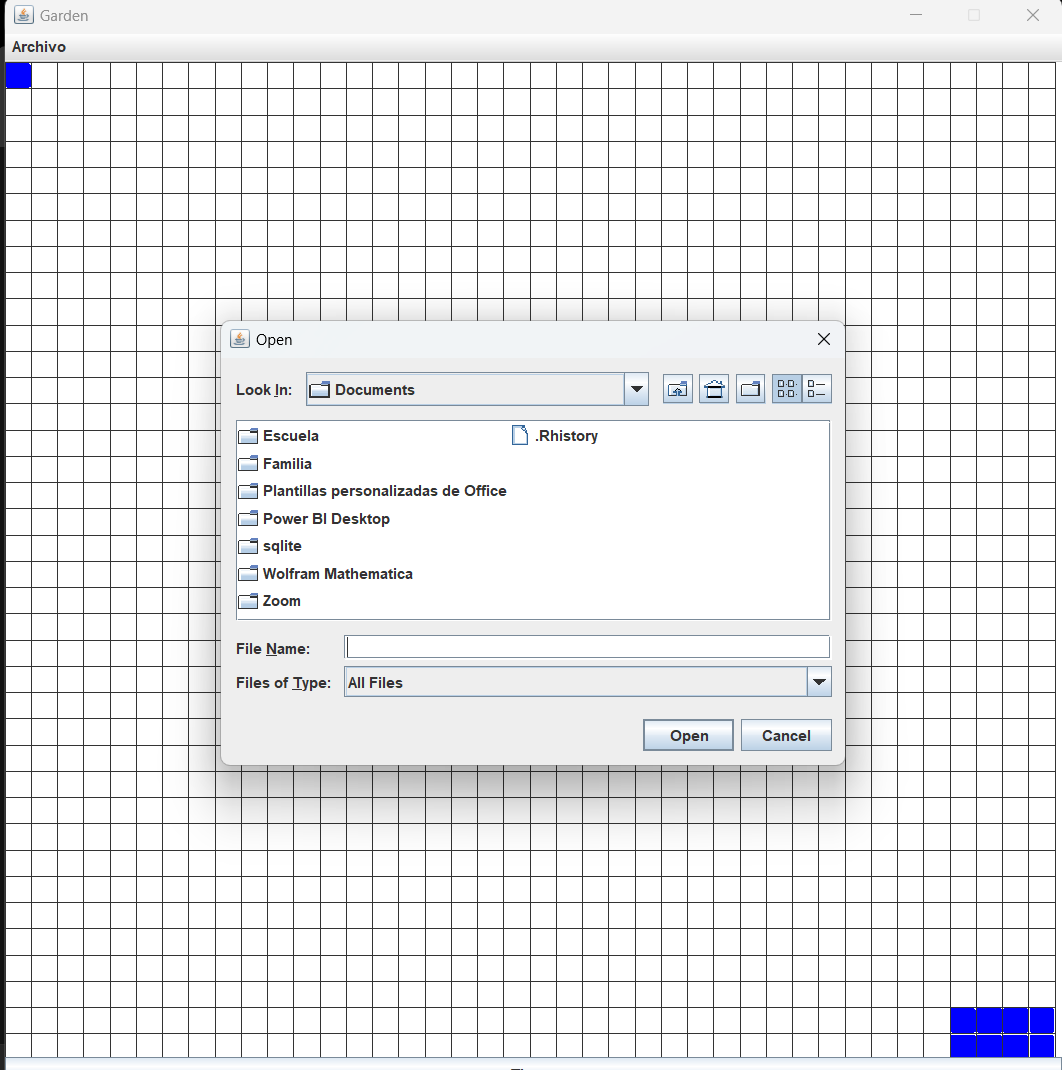


1. Realicen una prueba de aceptación para este método iniciando la aplicación, creando un nuevo estado y abriendo el archivo onegarden.dat. Capturen imágenes significativas de estos resultados.

El usuario abre la aplicación garden



Se dirige al apartado de archivo y selecciona abrir



Busca el archivo en donde guardo su jardin de 2 clicks y lo abre

Una vez da click en abrir, se actualiza su pantalla principal, mostrandole el jardin seleccionado

# **Implementando importar y exportar**

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

Estas operaciones nos van a permitir importar información del garden 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

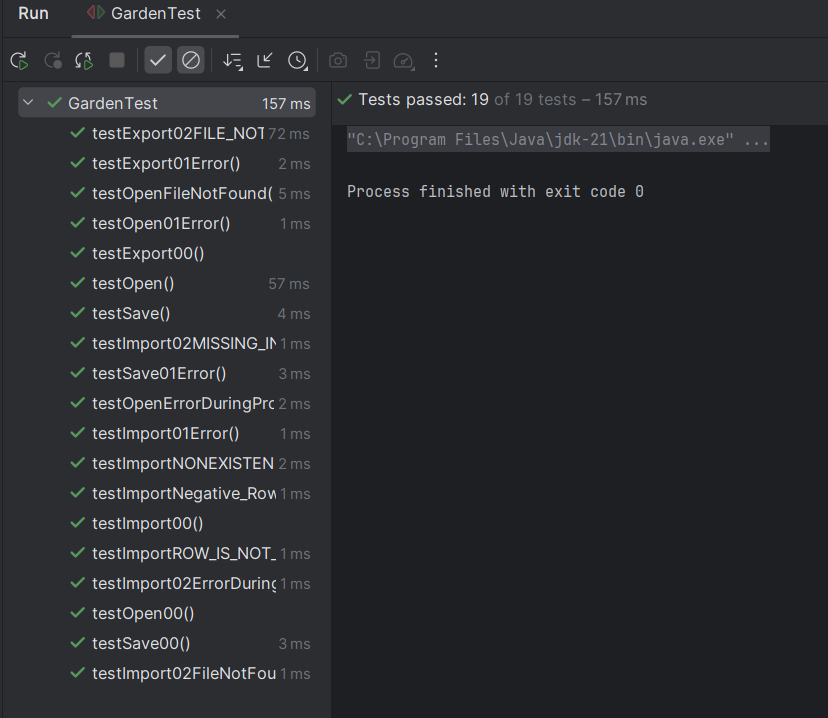
En cada línea asociada un elemento se especifica el tipo y la posición. Flower 10 10

Sand 20 20

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

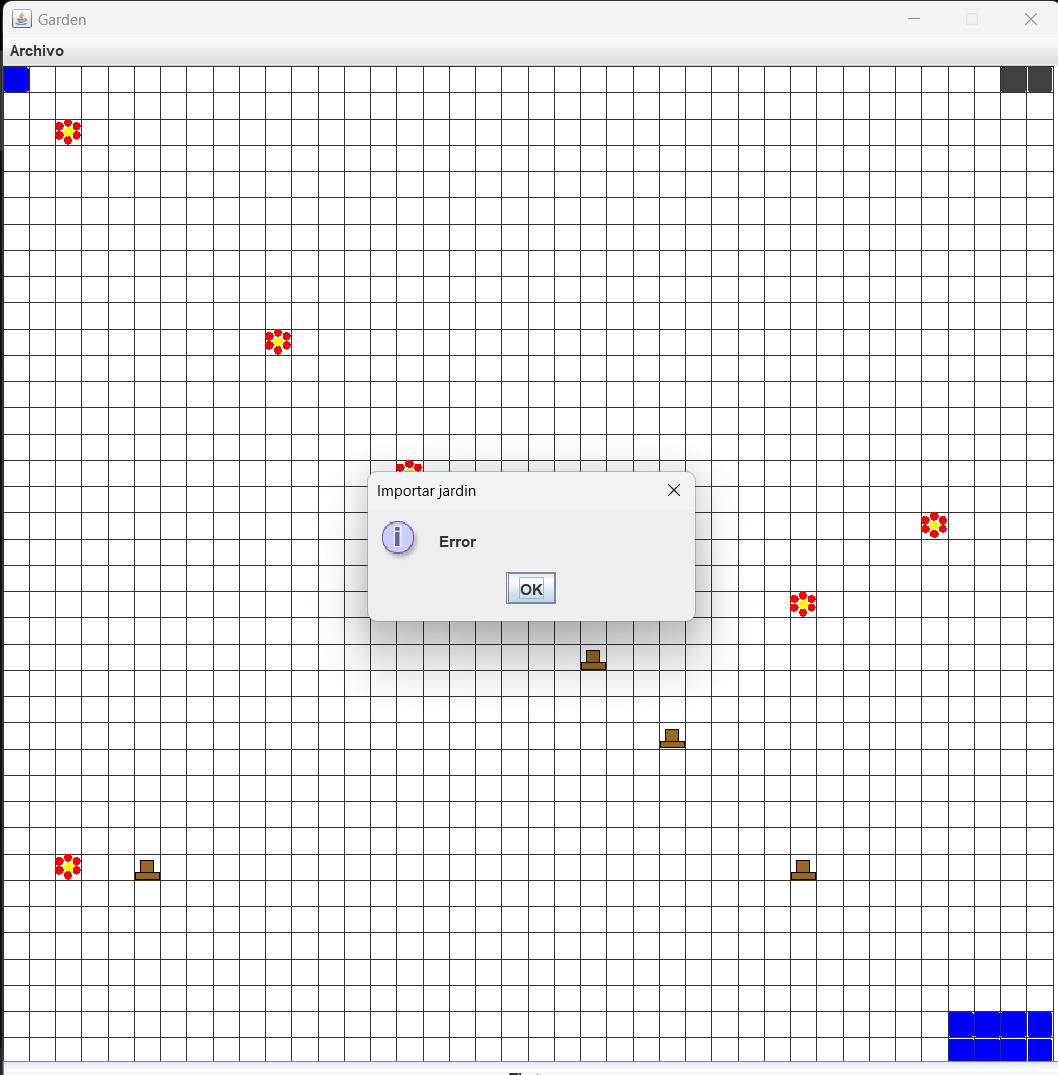
export00

1. 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.

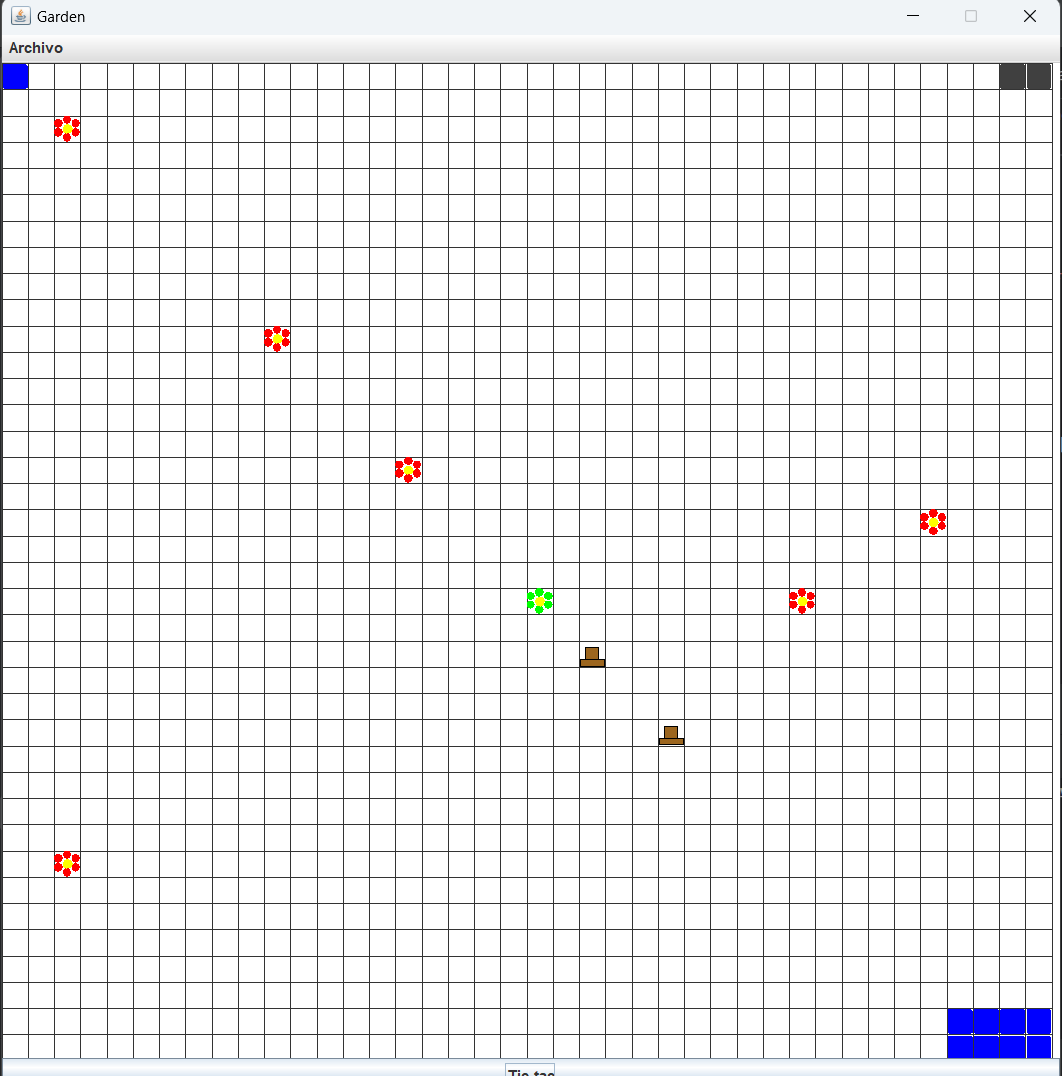


1. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como onegarden.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?

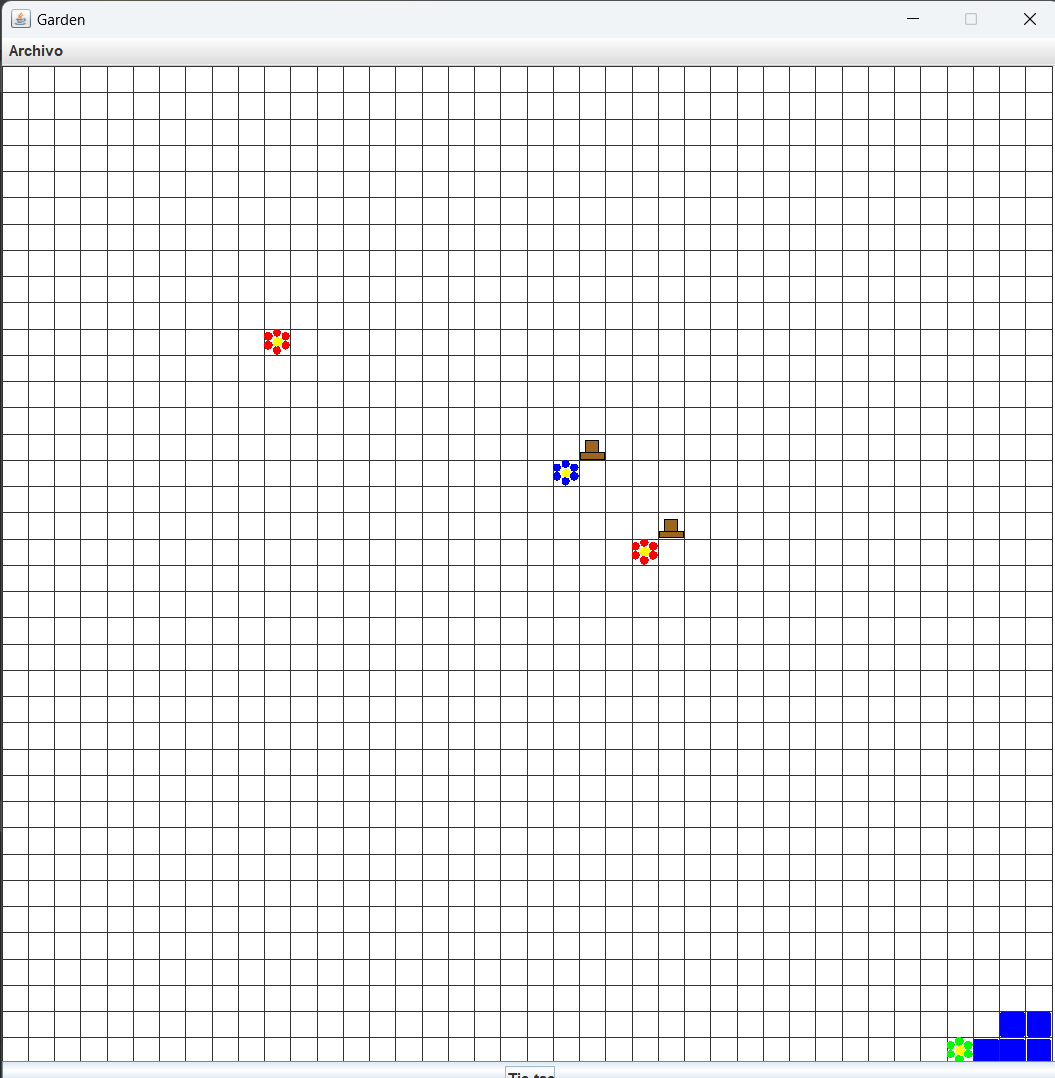
Cuando el usuario trata de acceder a un archivo que ha sido editado, ahora arroja error



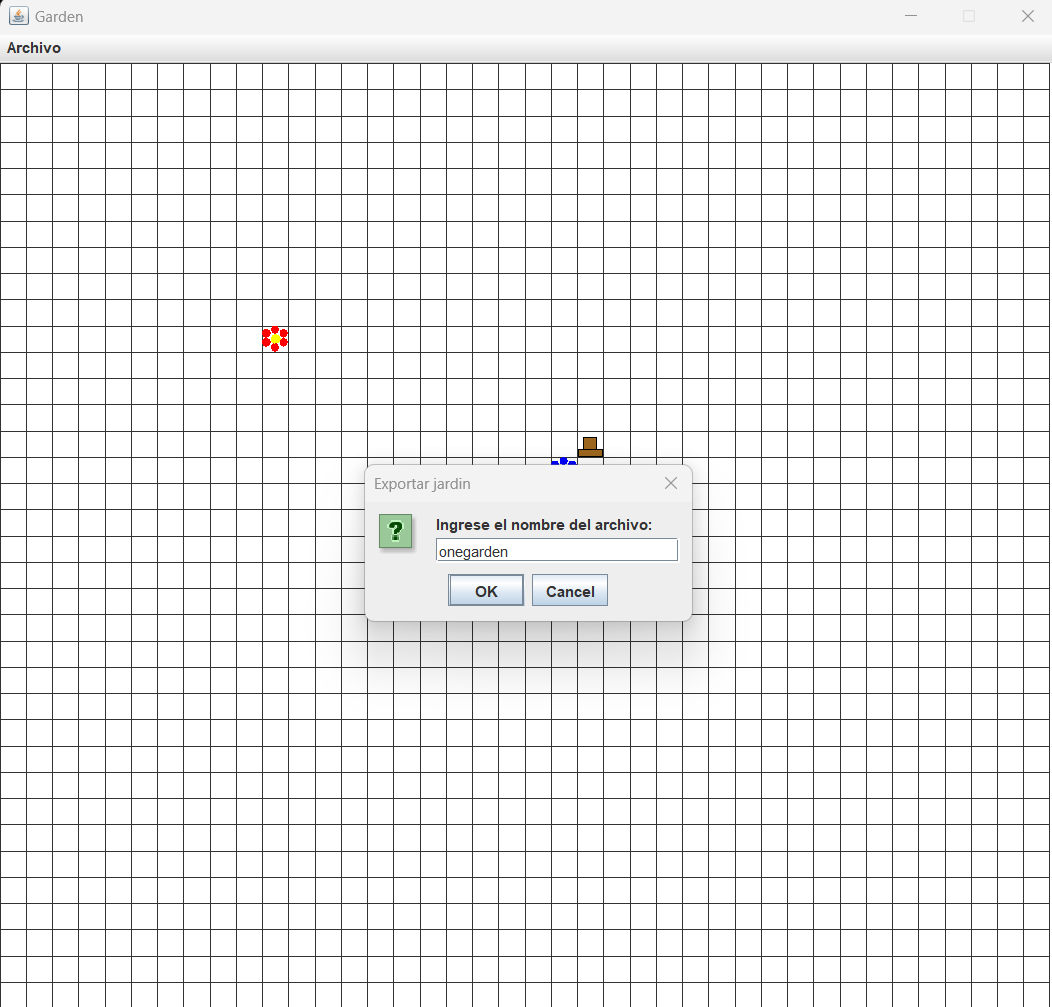
El usuario ingresa a la aplicación



El usuario realiza 8 clicks



El usuario decide exportar su jardin, dandole el nombre de onegarden

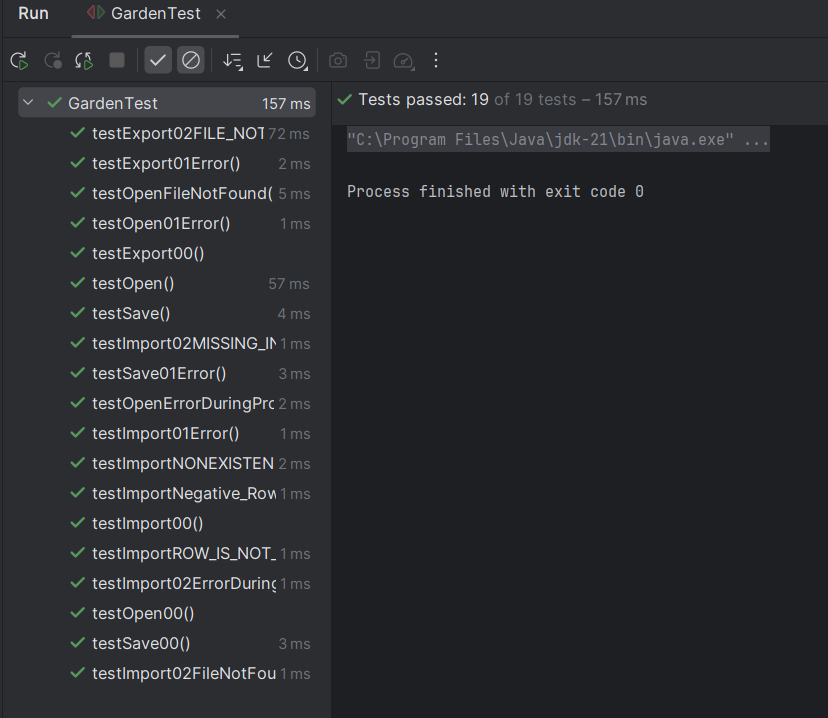


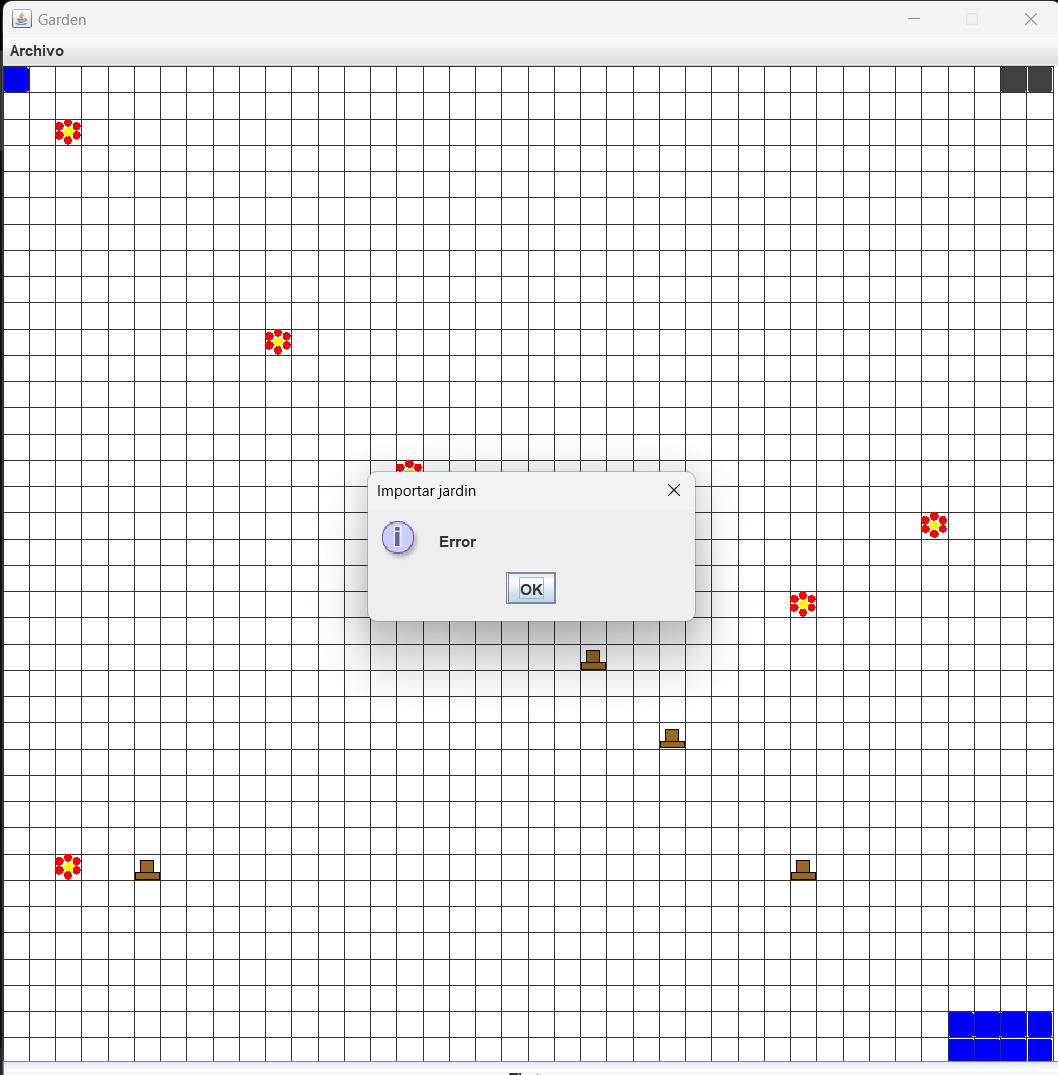
Cierra la aplicación

1. 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)





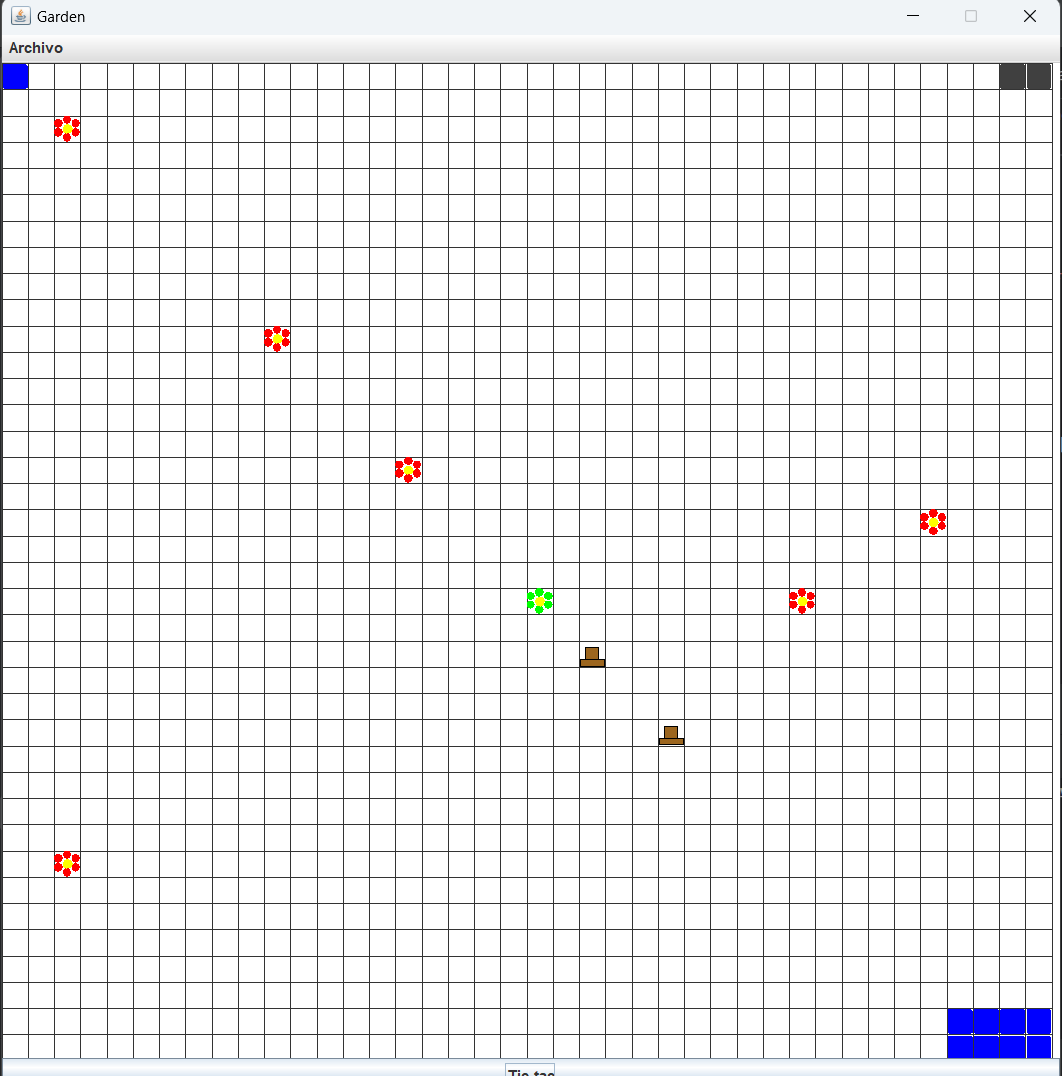
1. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a onegarden.txt. saliendo, entrando, creando una nueva e importando el archivo othergarden.txt. ¿Qué resultado obtuvieron? Capturen la pantalla final

Al tratar de abrir othergarden, arroja el error. Que fue el que se estableció por defecto

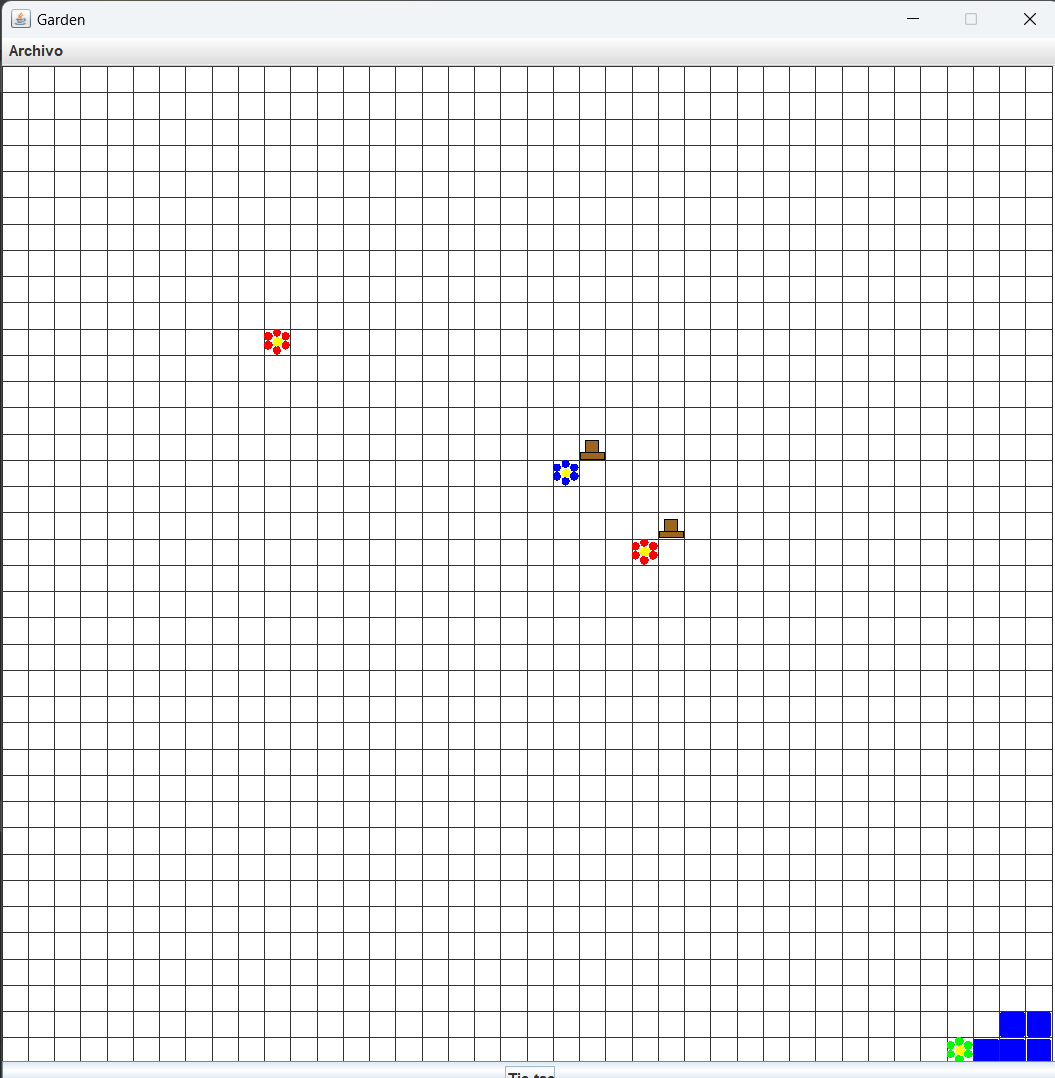
1. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto

correcto en onegarden.txt. e importe este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.

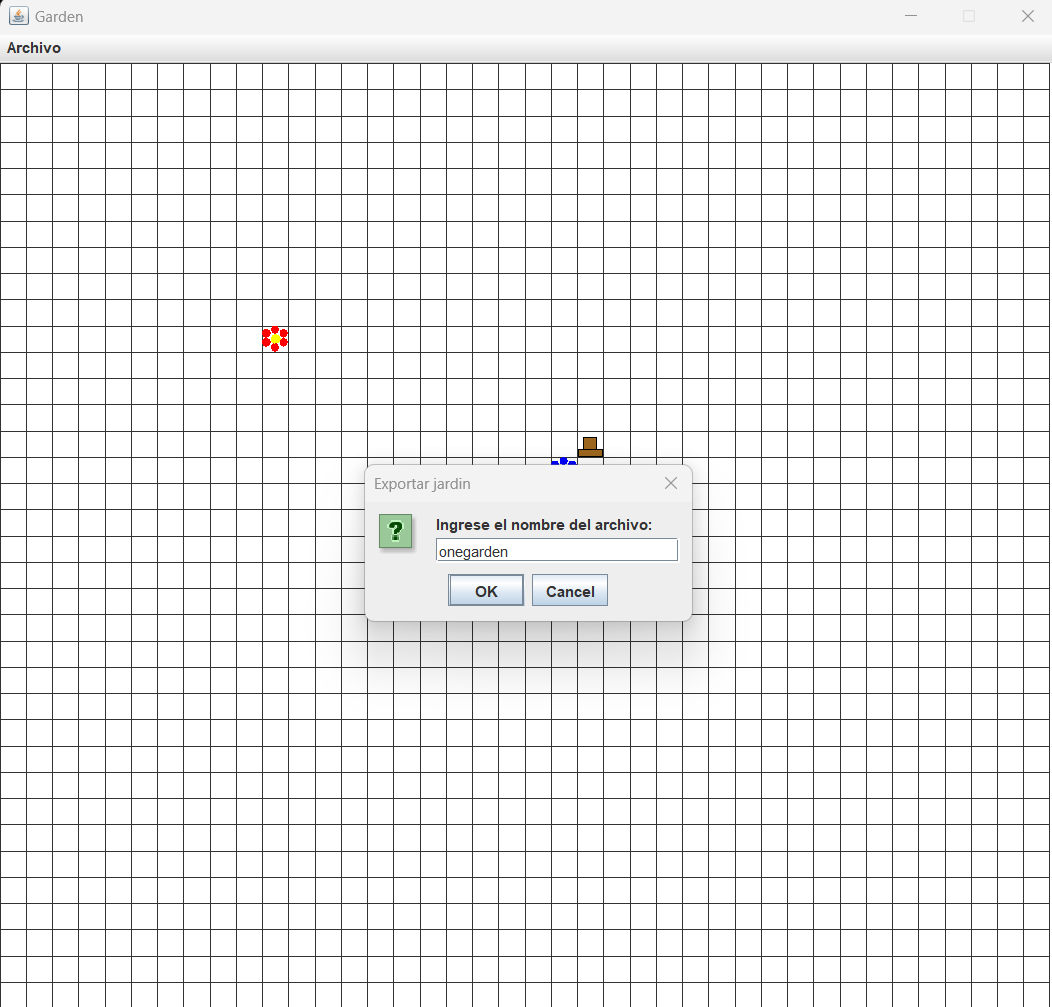
El usuario ingresa a la aplicación



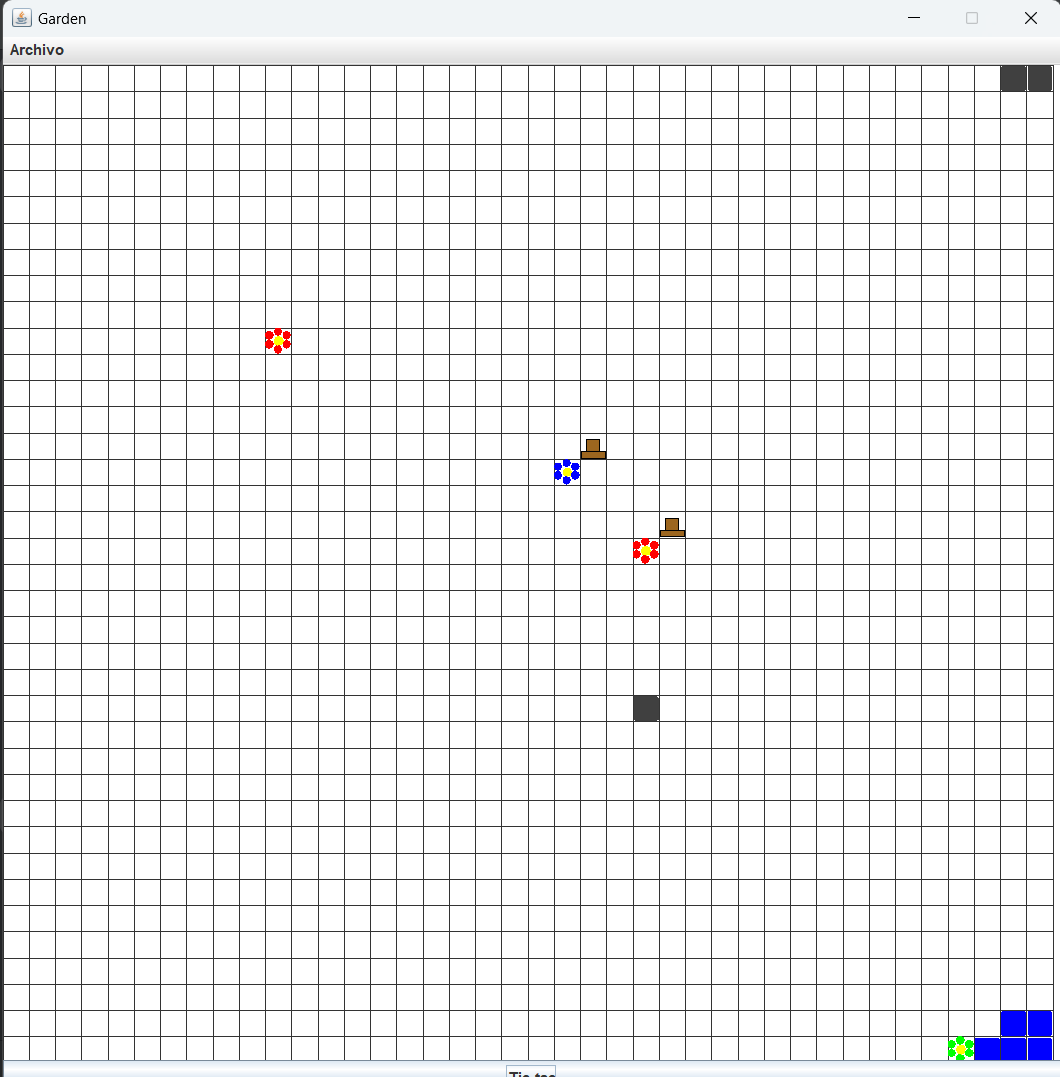
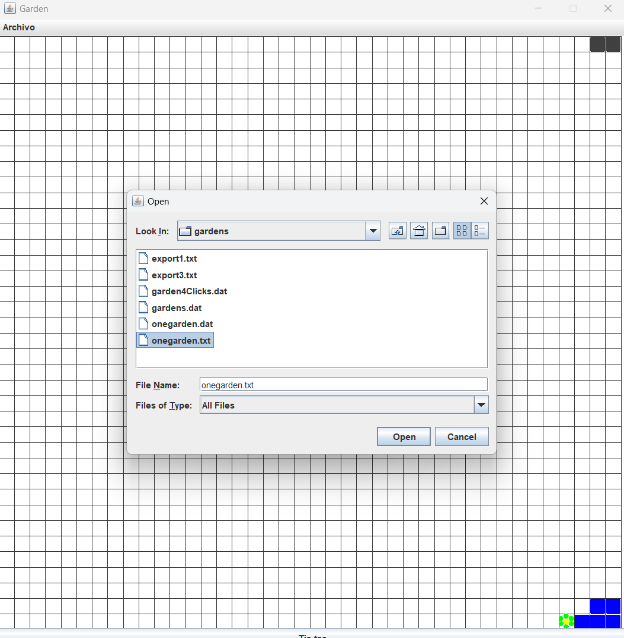
El usuario realiza 8 clicks



El usuario decide exportar su jardin, dandole el nombre de onegarden



Cierra la aplicación y cuando la vuelve abrir, decide abrir su ultimo jardin (onegarden)



Como resultado final, el usuario pudo abrir nuevamente su último jardín.

# **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 abrir el archivo, dar tres clicks y salvar el archivo, se crea un archivo .dat, con el nombre que el usuario le ha asignado. Como todas las clases implementan la interfaz Serializable, Java por debajo almacena y codifica toda la información de la clase en cadenas de bytes, que posteriormente cuando se abre el archivo, vuelve a traducirlas a objetos, reestableciendo todo “automaticamente”.

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

Al exportar un archivo, lo que hacemos es recorrer toda la matriz, dejando por escrito en un archivo .txt todas las entidades que se encontraban en el jardín en ese moemnto. Posteriormente, cuando se abre, se lee todo el archivo, si en las lineas encuentra los nombres de las clases, junto con sus dos posiciones, crea el objeto y lo añade al jardin en las posiciones dadas.

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

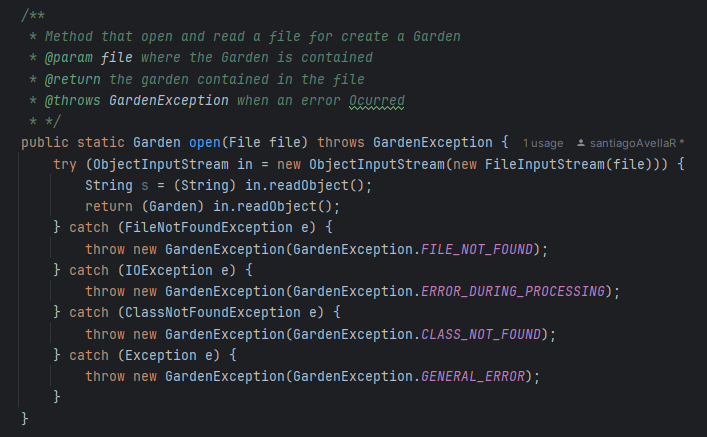
Por un lado, en 1, java se encarga de guardar absolutamente toda la información de los objetos de manera automática. El texto no es legible para el ser humano. Si se altera en un carácter el texto, en ambos casos, va a haber un error.

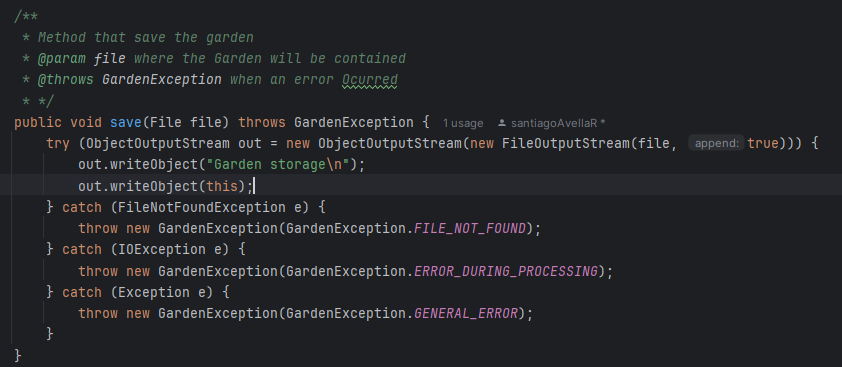
Por el lado de 2, si cada objeto tuviera más atributos por dentro, sería un proceso más tedioso, ya que tendríamos que estructurar una organización para acomodar la información y además dejar por escrito todos los atributos que componen al objeto. En este caso, la información puede llegar a ser entendida por una persona.

# **Perfeccionando salvar y abrir**

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

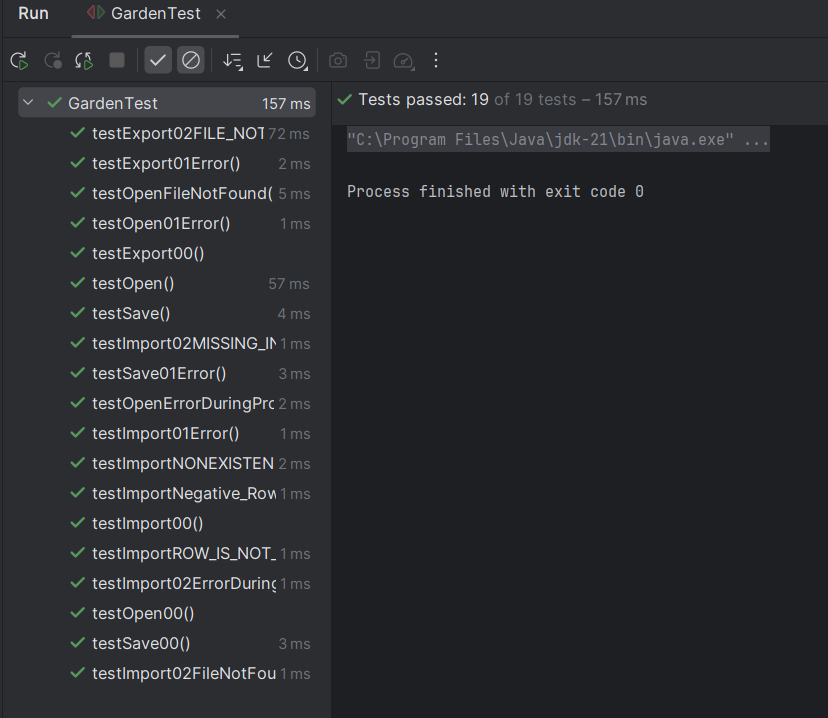
1. Copien las versiones actuales de open y save y renómbrenlos como open01 y save01
2. Perfeccionen el manejo de excepciones de los métodos open y save detallando los errores. No olviden pruebas de unidad.





Puede ocurrir:

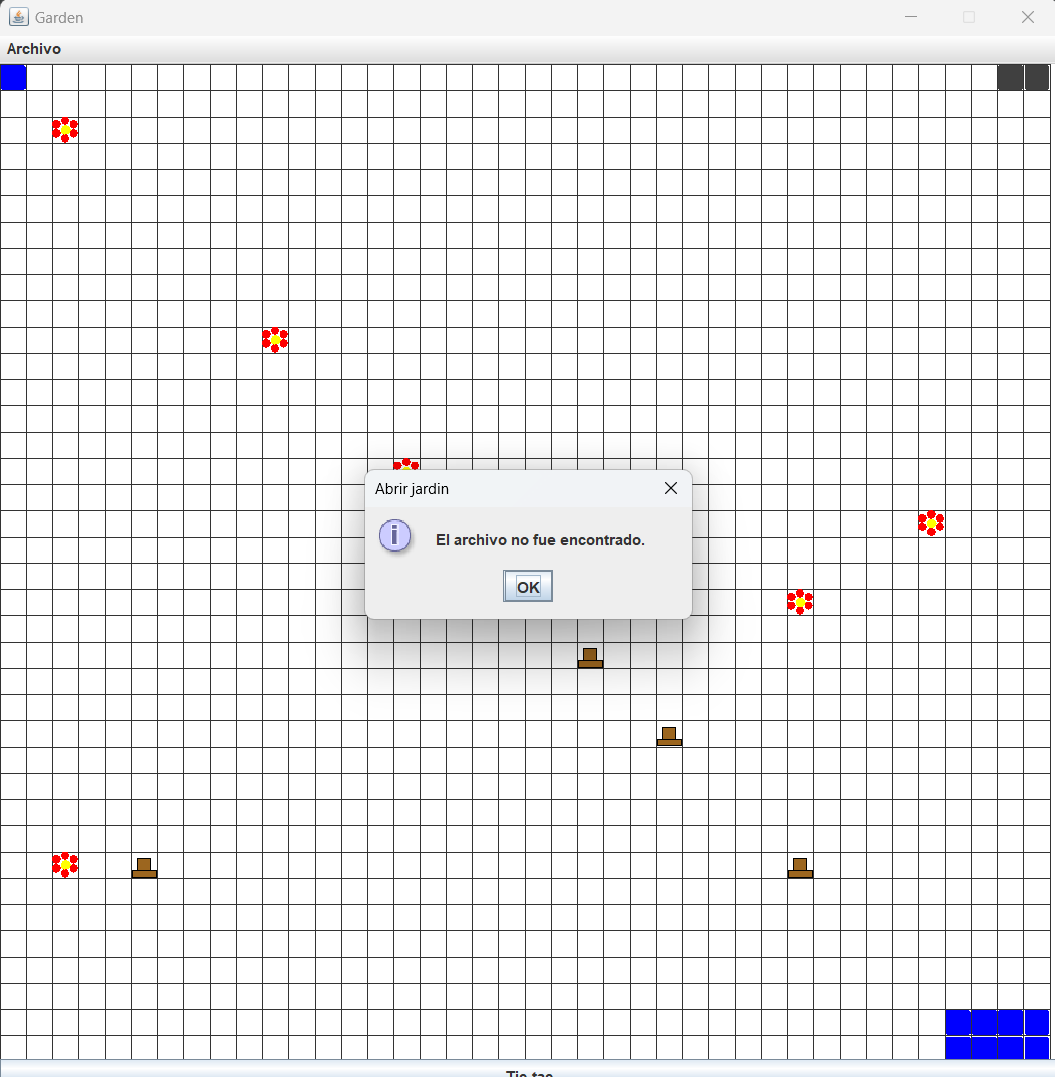
* FileNotFoundException, en caso de que el archivo no se haya encontrado
* IOException, por algún error durante el proceso de abrir, cerrar o durante la modificación del archivo.
* ClassNotFoundException, en caso de que una clase no se encuentre durante el proceso de decodificar los bytes.
* Exception para el error general



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

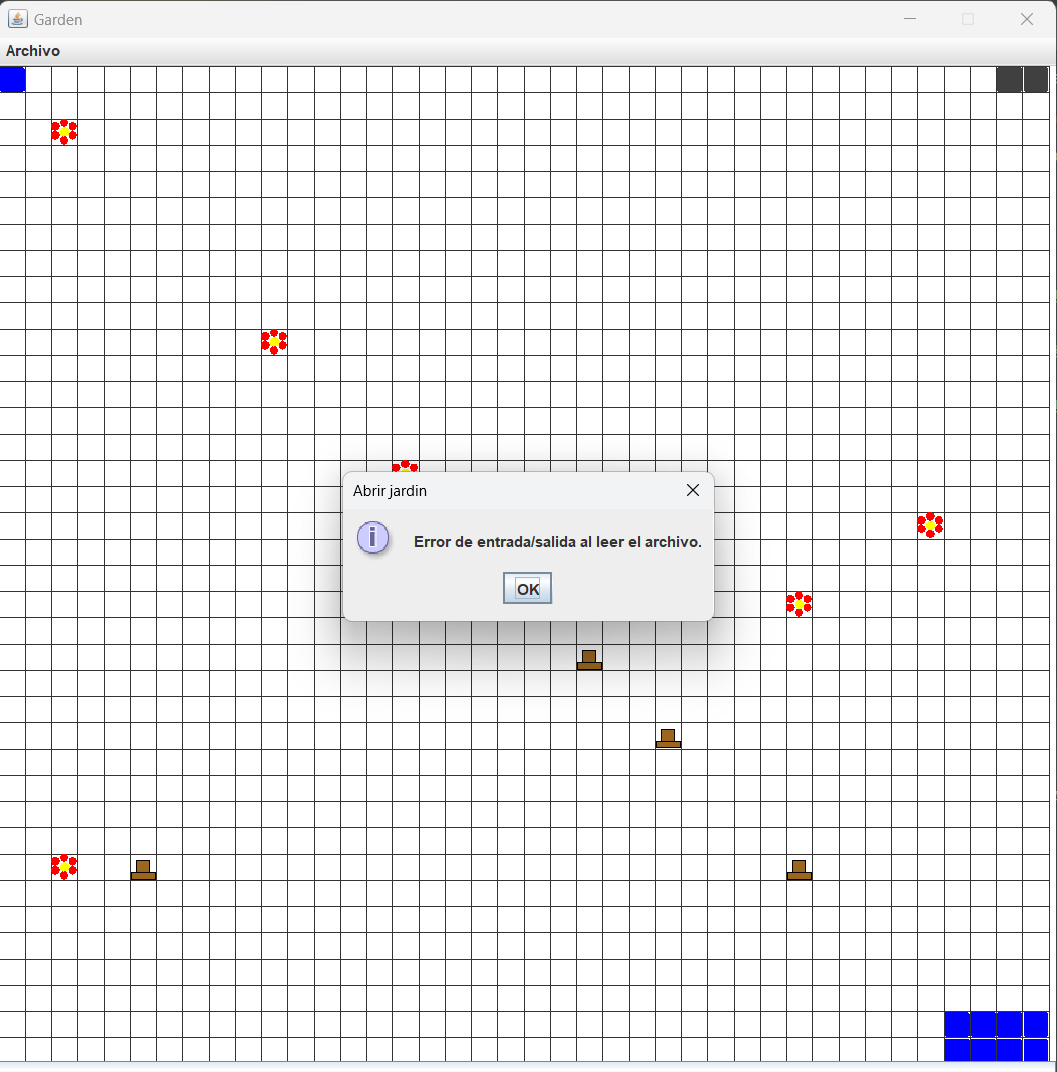
Para archivo no encontrado:

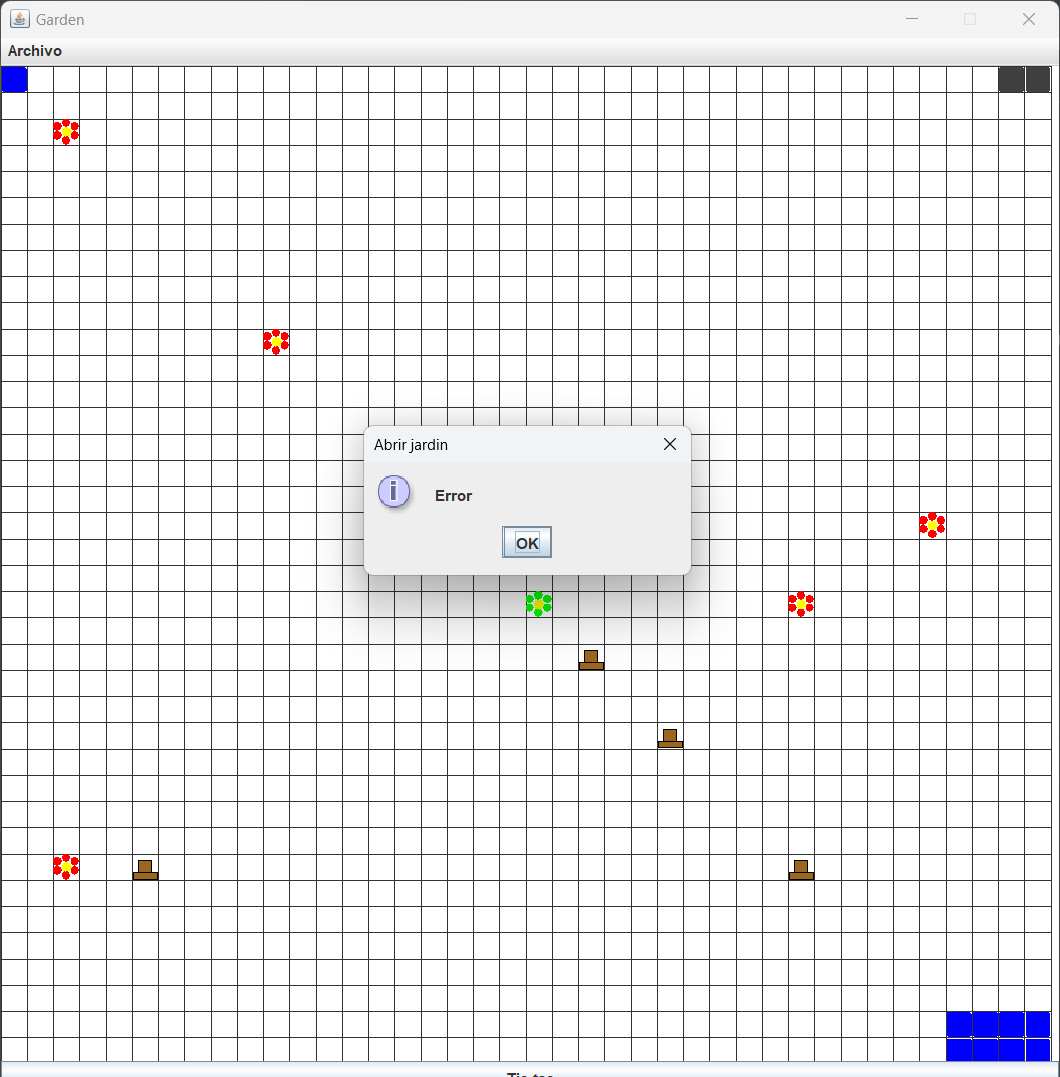
El usuario trata de abrir el archivo con un nombre no existente



Error al manipular, leer o escribir en el archivo:

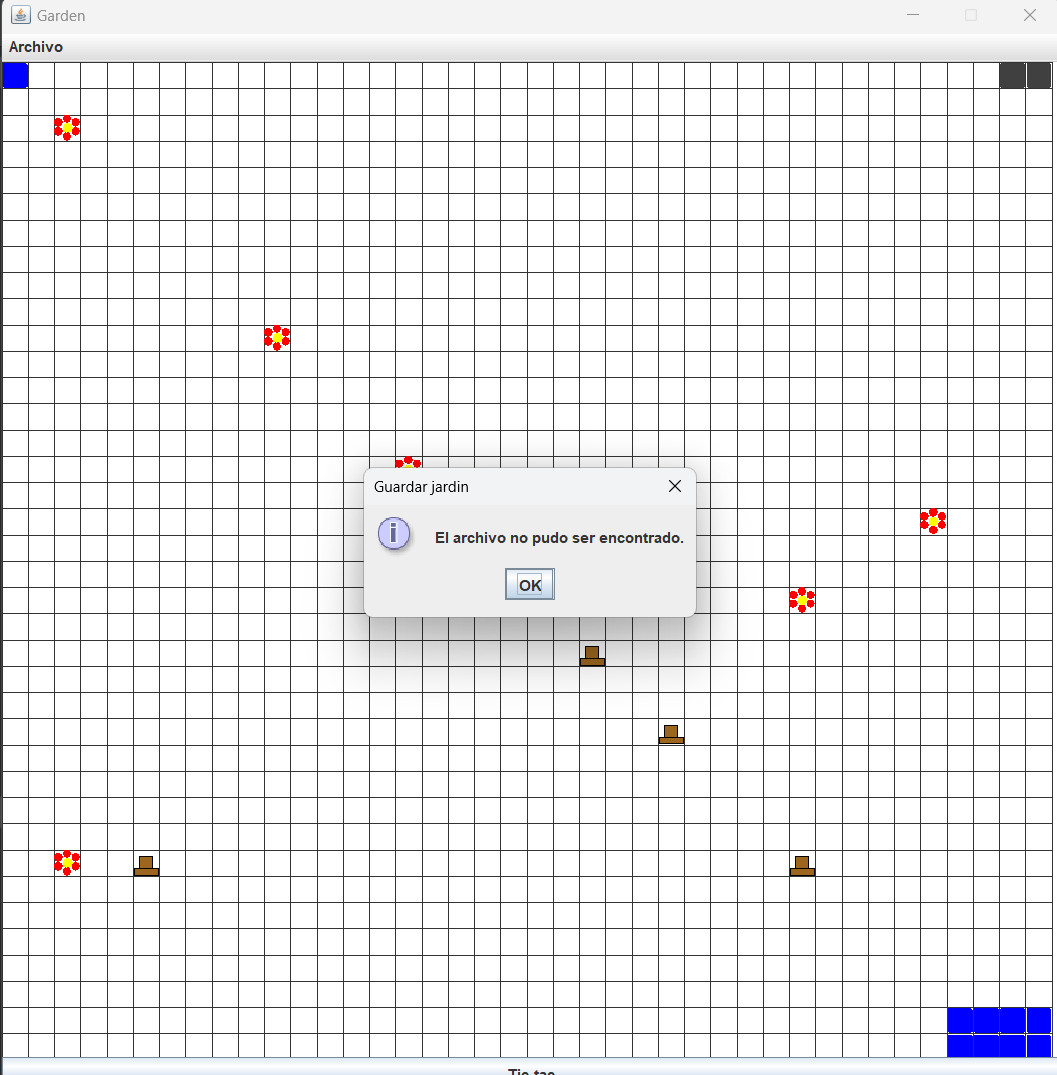
Por accidente, el usuario altera algo del código del archivo .dat, cuando lo abre, le aparece el error

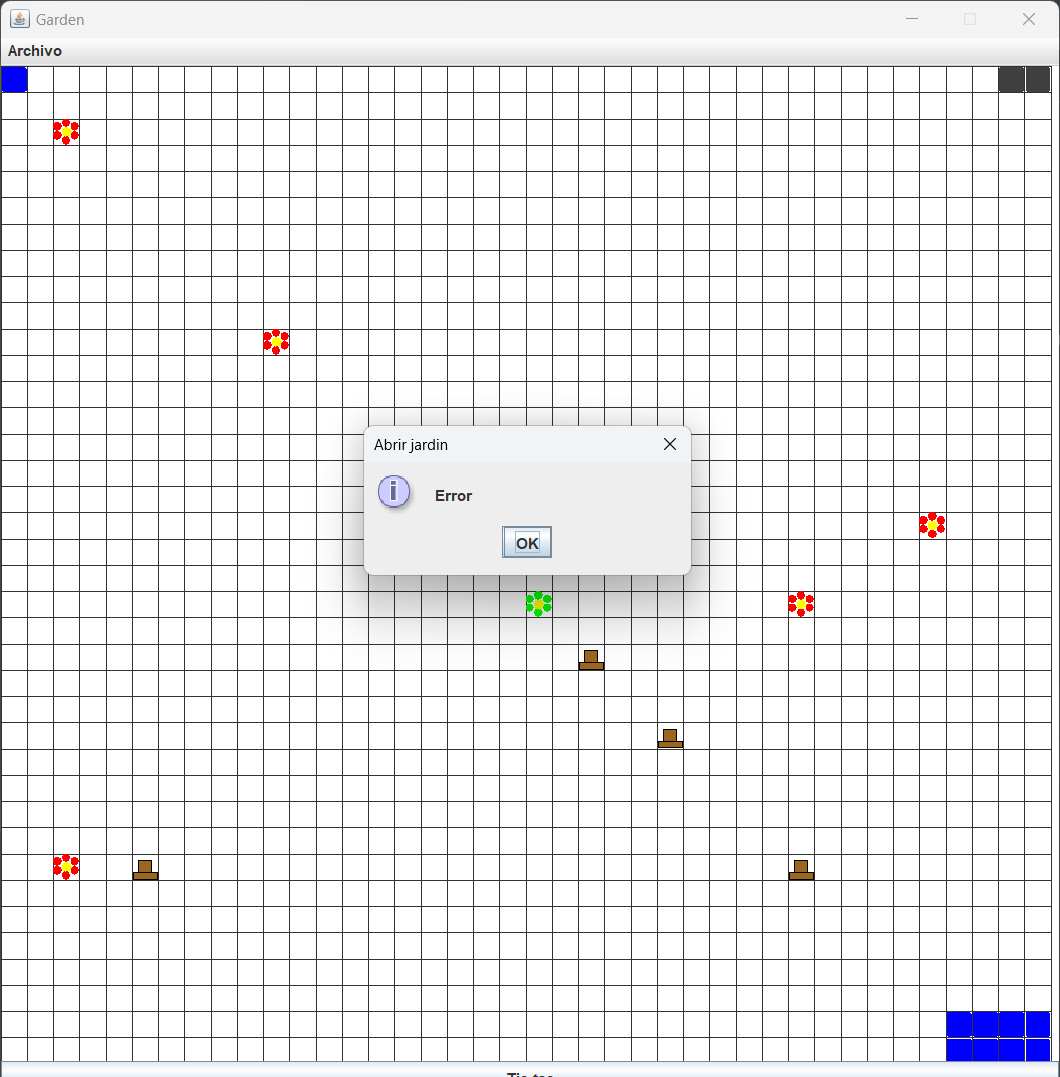


Error general  


Si implementaramos una nueva “cosa” (Thing) en el jardín y traramos de codificarlo, arrojaría el Class not found.

En el caso de save, cuando le pasamos un folder inexistente



Error general  


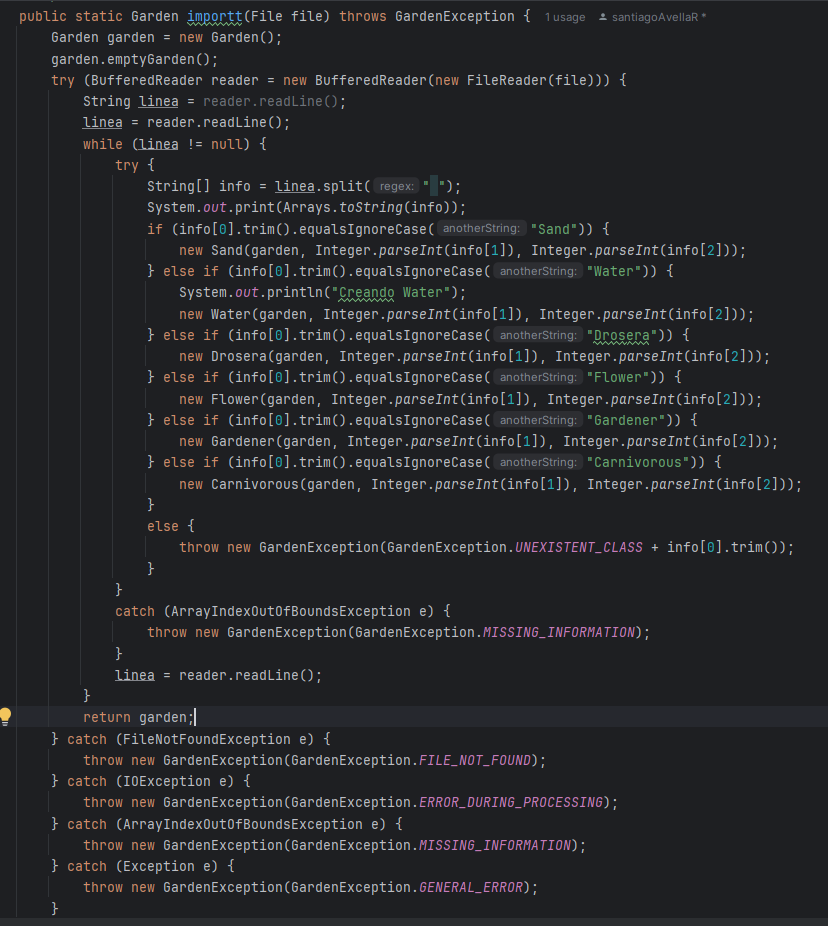
# **Perfeccionando importar y exportar.**

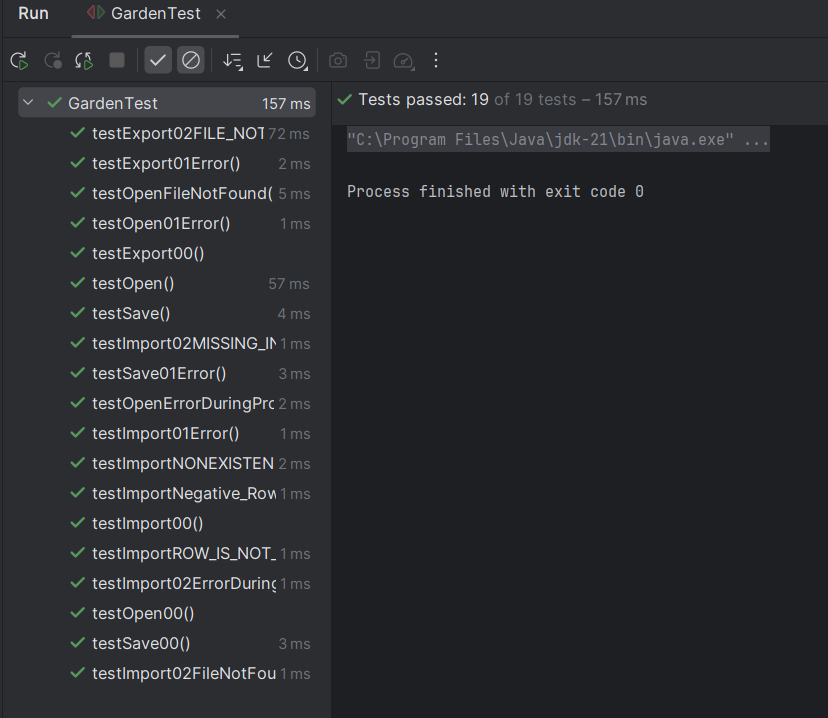
**[En lab06.doc, \*.asta , gardenErr.txt \*.java] [NO OLVIDEN BDD y MDD]**

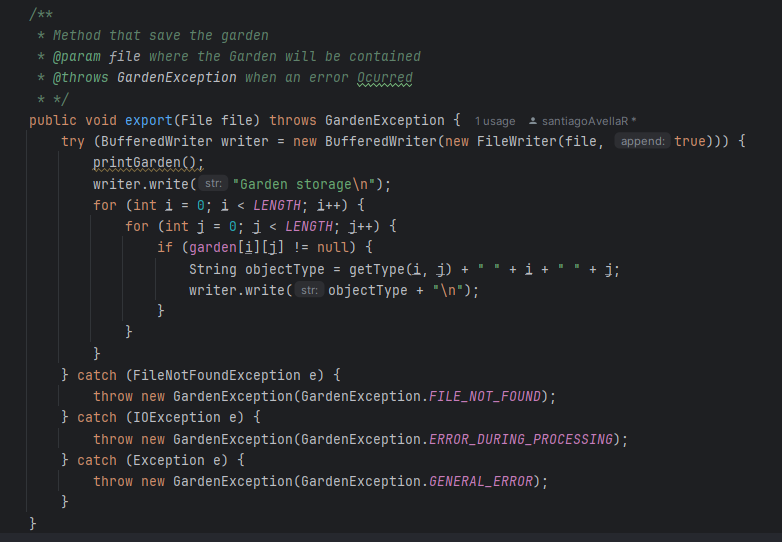
1. Copien las versiones actuales de import y export y renómbrenlos como import01 y

export01

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







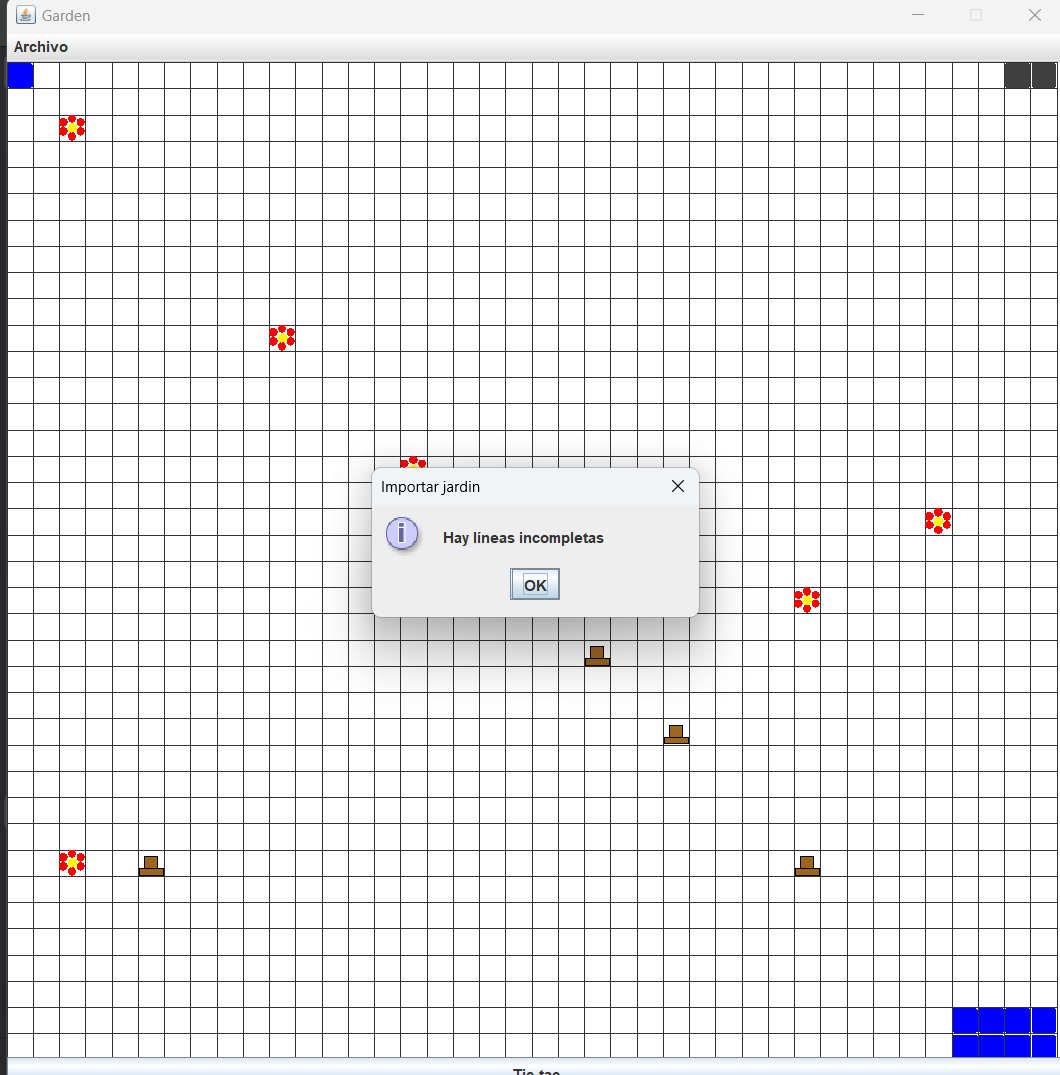
1. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

El usuario trata de importar un archivo que empieza de la siguiente manera

Garden storage

Water 0

Gardener 23 25



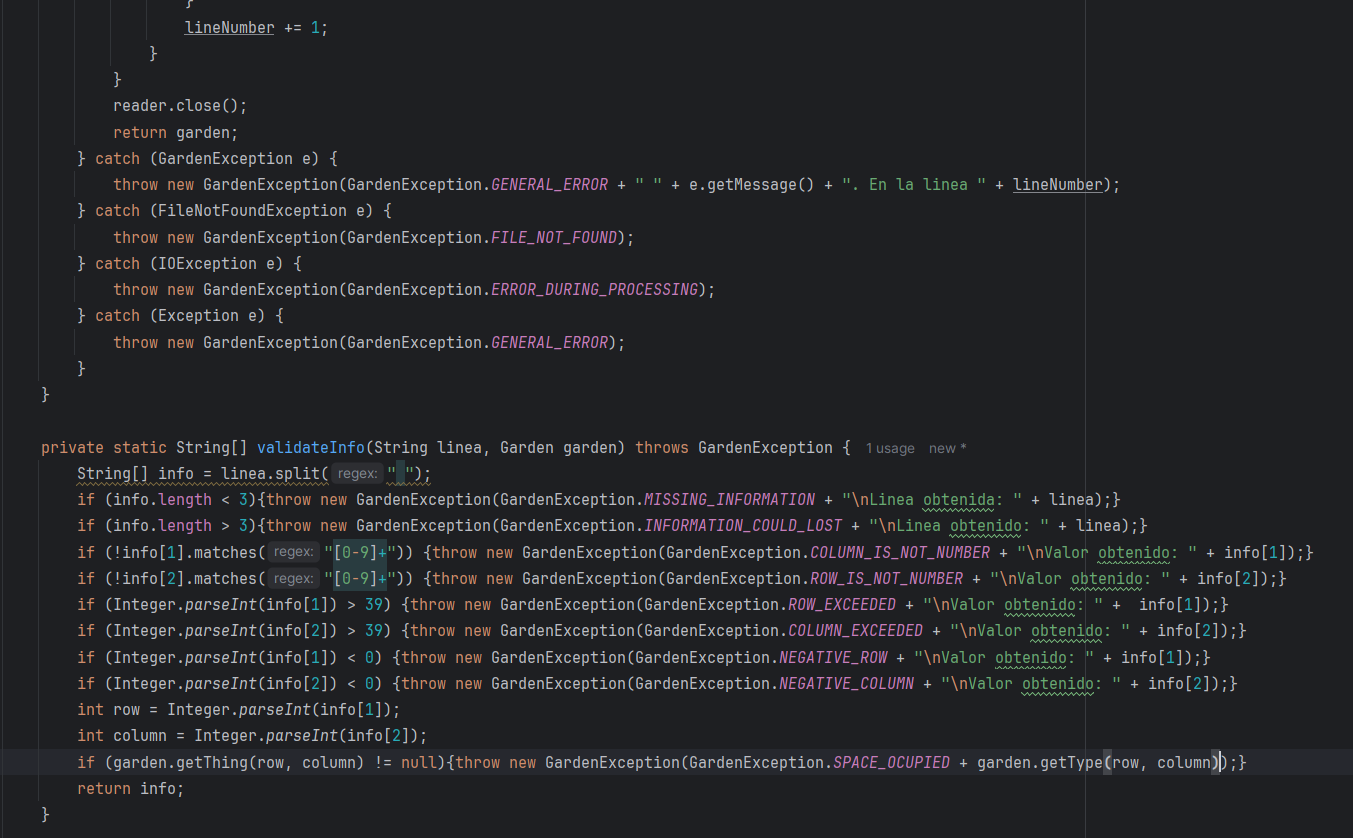
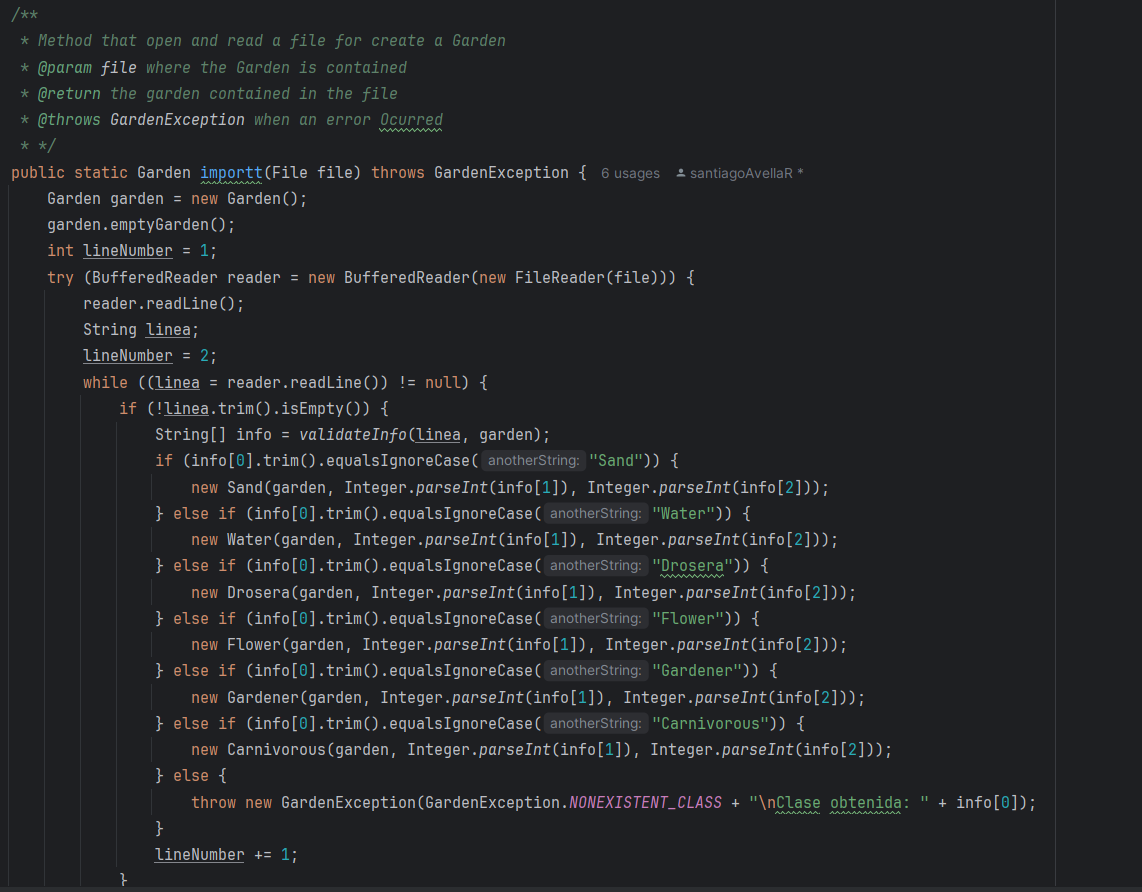
# **Perfeccionando importar. Hacia un minicompilador.**

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

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

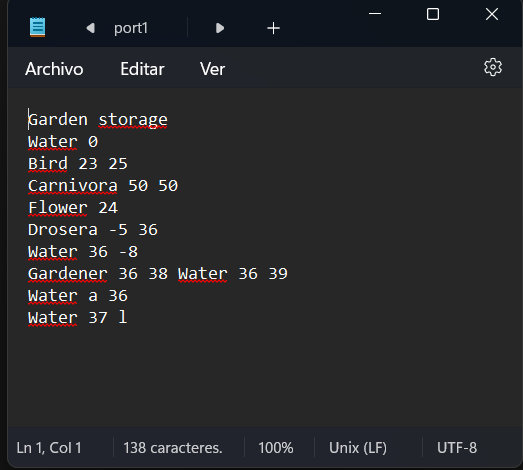
export02

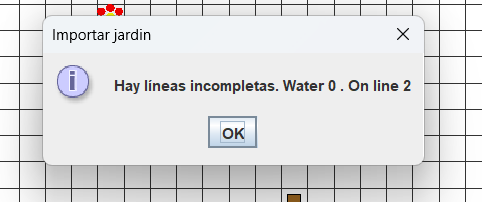
1. Perfeccionen el método **import** para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palabra que tiene el error y causa de error.

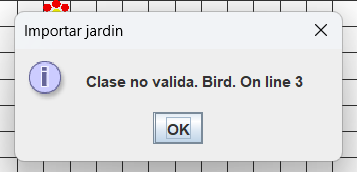


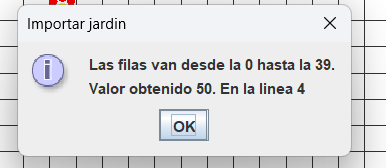
1. Escriban otro archivo con errores, llámelo gardenErr.txt, para ir arreglándolo con ayuda de su “importador”. Presente las pantallas que contengan los errores.

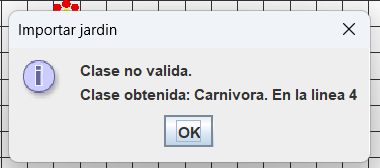
Caso inicial

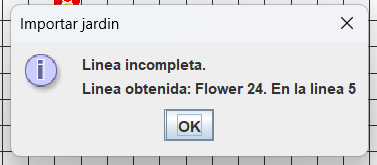


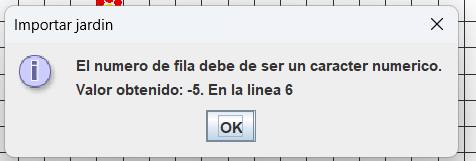


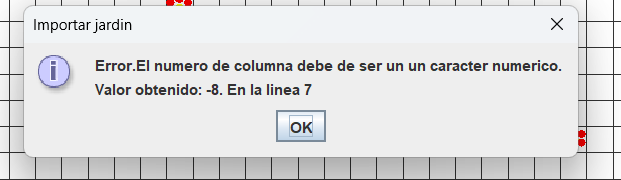


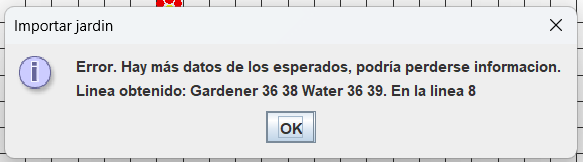


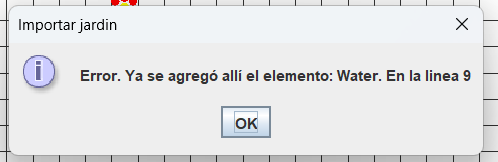


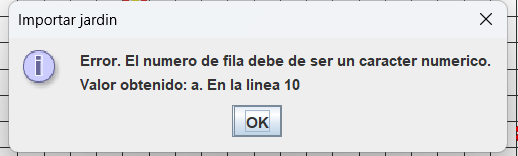


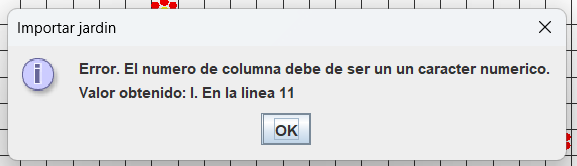




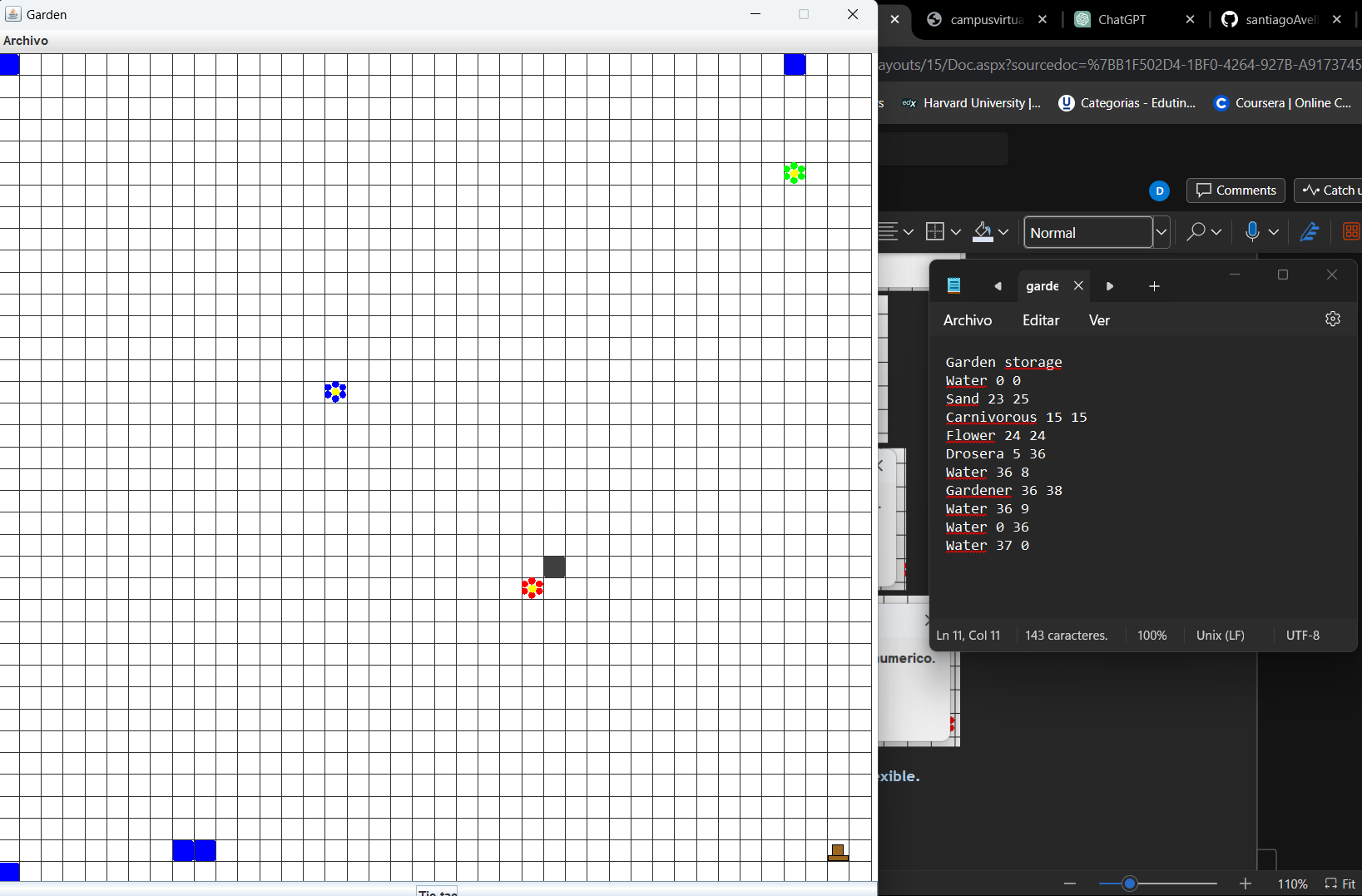








Resultado final



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

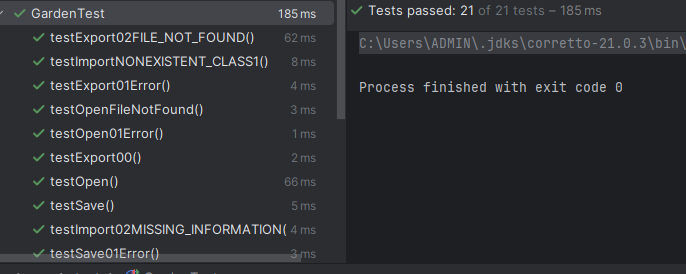
**[En lab06.doc, \*.asta , gardenFlex.txt \*.java] [NO OLVIDEN BDD y MDD]**

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

export03

1. 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)



1. Escriban otro archivo de pruebas, llámelo gardenErrG.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)

Alrededor de 10 horas por cada integrante

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

Terminado. Debido a que nos organizamos de una manera óptima y eficiente.

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

Collective ownership, debido a que de esta manera debemos fuimos mas claros en los cambios realizados por cada uno, para que así fuera mas fácil de entender para el otro y si había un error era mas fácil solucionarlo gracias a la interacción de los 2.

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

Lograr crear objetos a partir del nombre, debido a que la información que había no siempre fue muy clara, lo que a veces nos hacia retroceder.

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

Ninguno.

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

Trabajar en equipo, buena comunicación, organización, nos comprometemos a seguir mejorando.