PROGRAMACIÓN ORIENTADA A OBJETOS
I/O
2019-
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, salvar, importar y exportar.
3. Controlar las excepciones generadas al trabajar con archivos.
4. Vivenciar las prácticas .

-ENTREGA

- Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
- Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.
- En el foro de entrega de avance deben indicar los logros y los problemas pendientes por resolver.

DESARROLLO

Preparando

En este laboratorio vamos a extender el proyecto `bodyTIc` adicionando un menú barra con las opciones básicas de entrada-salida y las opciones estándar de iniciar 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.

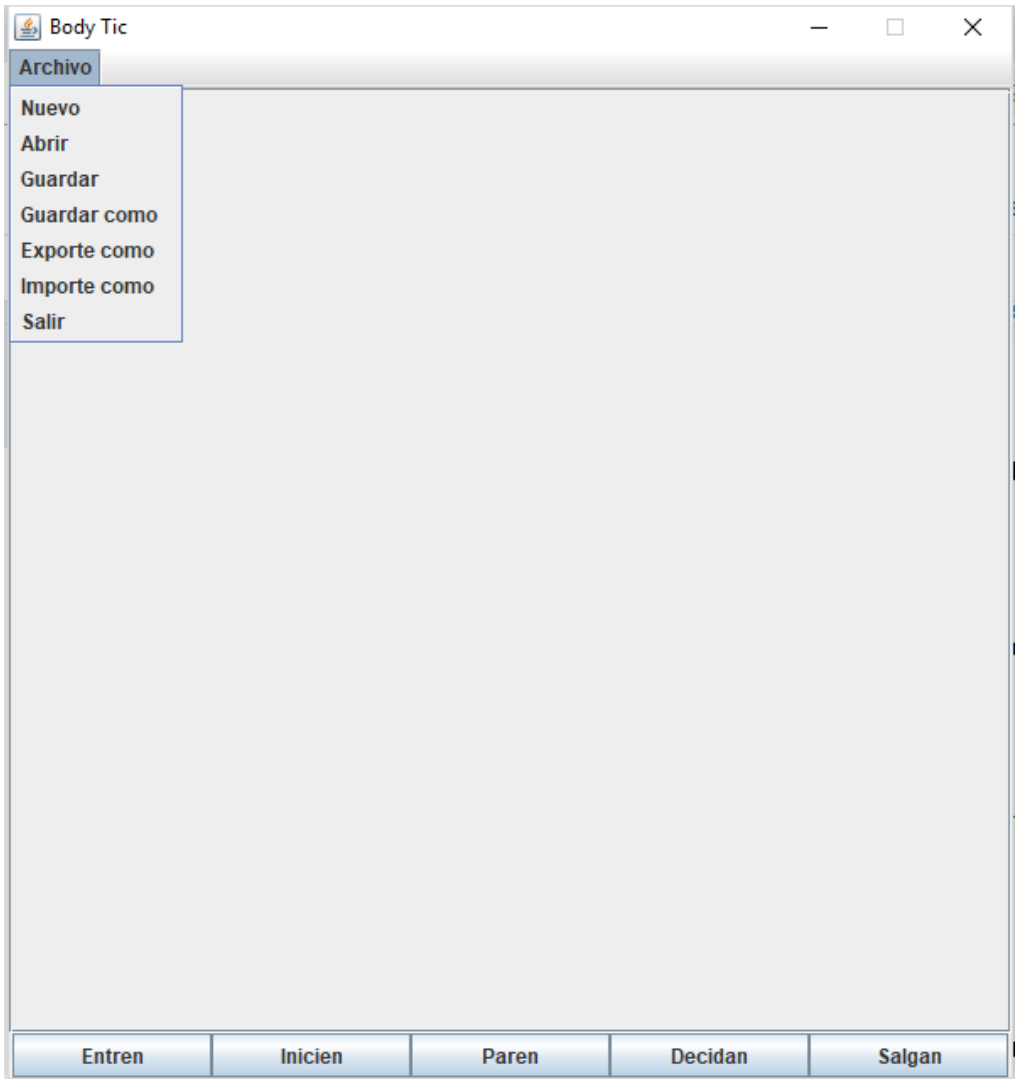
Creando la maqueta

[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 los métodos correspondientes a reiniciar y a las cuatro opciones básicas de entrada-salida (`salve como`,`abra`,`exporte como`,`importe`). Los métodos deben simplemente propagar una `bodyTIcExcepcion` con el mensaje: "Opción … en construcción". Los métodos de entrada salida 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 iniciar y salir. Para esto creen el método `prepareElementosMenu.` Capturen la pantalla correspondiente.


Body Tic
Archivo
Nuevo
Abrir
Guardar
Guardar como
Exporte como
Importe como
Salir
Entren
Inicien
Paren
Decidan
Salgan

3. **CONTROLADOR:** Construyan los controladores correspondientes a estas acciones. Para esto creen el método `prepareAccionesMenu` y los métodos base del controlador. Estos últimos métodos, por ahora, sólo deben llaman directamente el método correspondiente de la capa de aplicación (archivo nulo). Capturen una pantalla significativa.

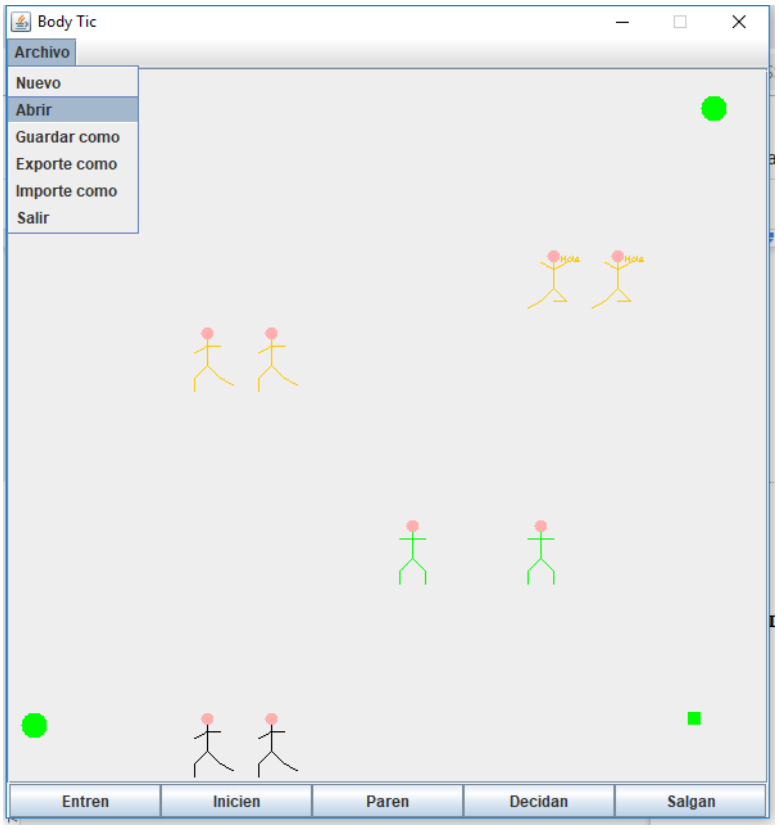
Implementando salir e iniciar

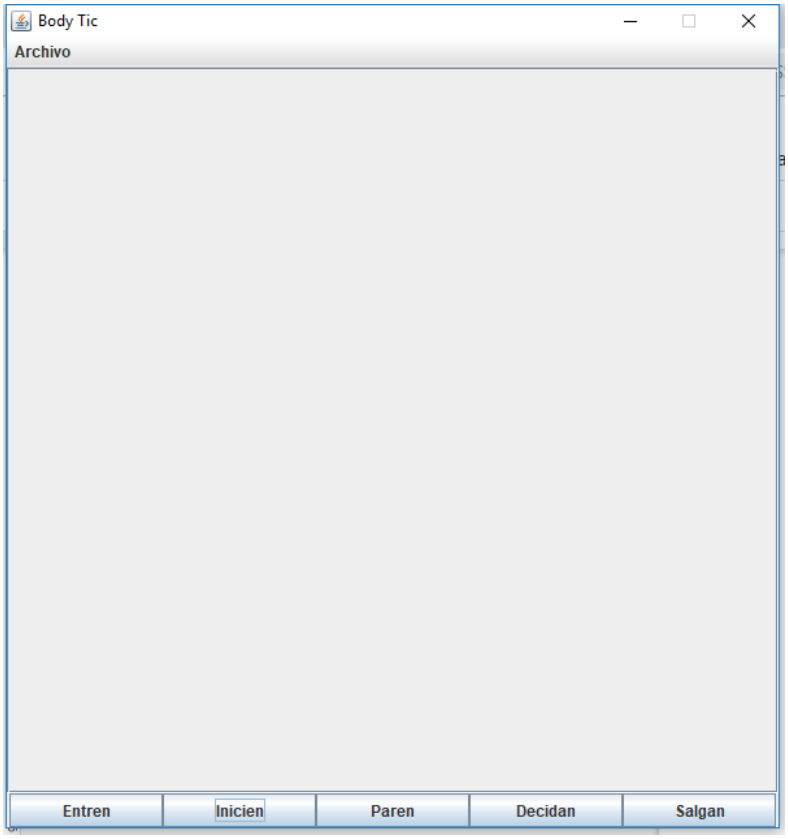
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salir e iniciar van a ofrecer los dos servicios estándar de las aplicaciones. El primero no requiere ir a capa de aplicación y el segundo sí.

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

2. Construyan el método `opcionIniciar` que crea un nuevo `bodyTIc`. Capturen una pantalla significativa.



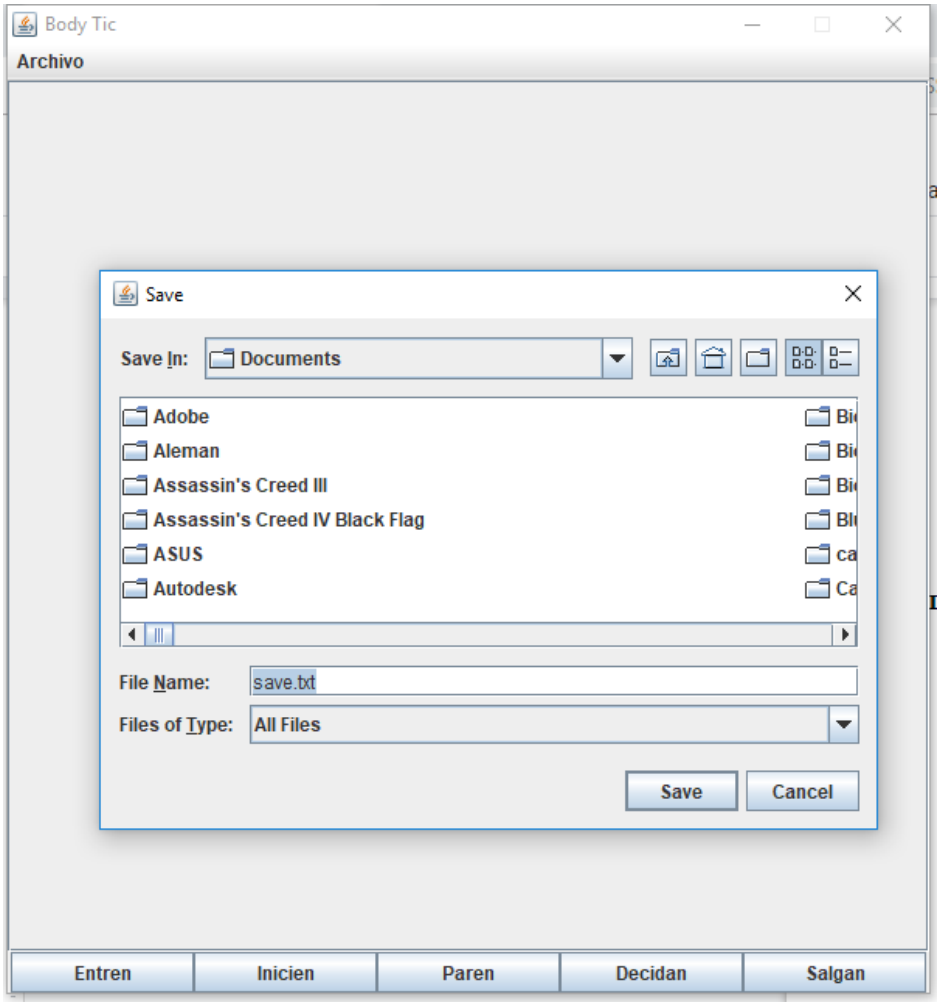


Implementando salvar y abrir

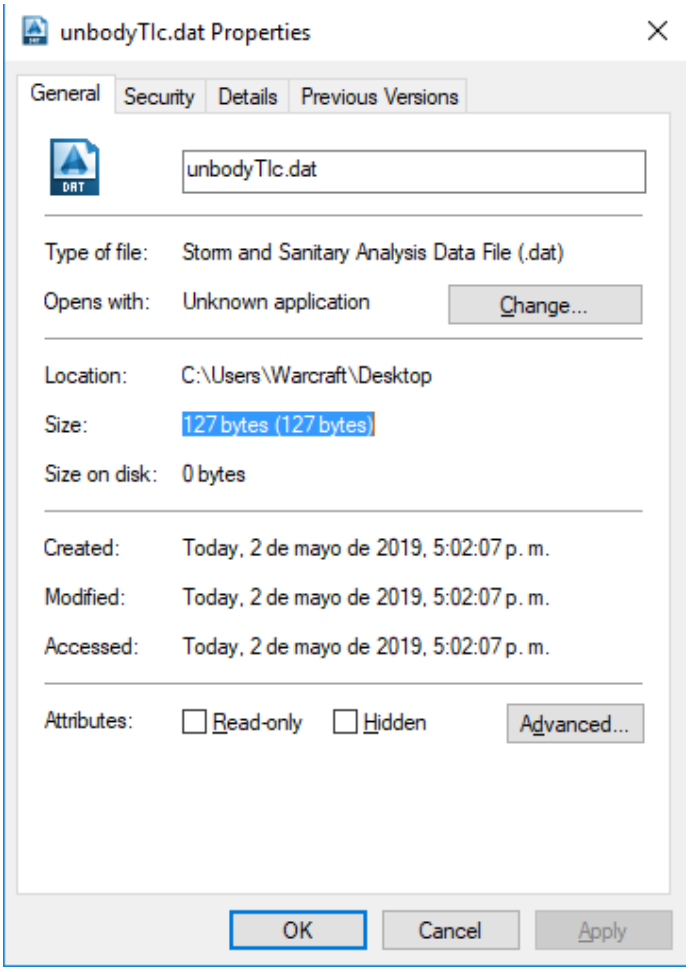
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salvar y abrir van a ofrecer servicios de persistencia de un `bodyTIc` como objeto. Los nombres de los archivos deben tener como apellido .dat.

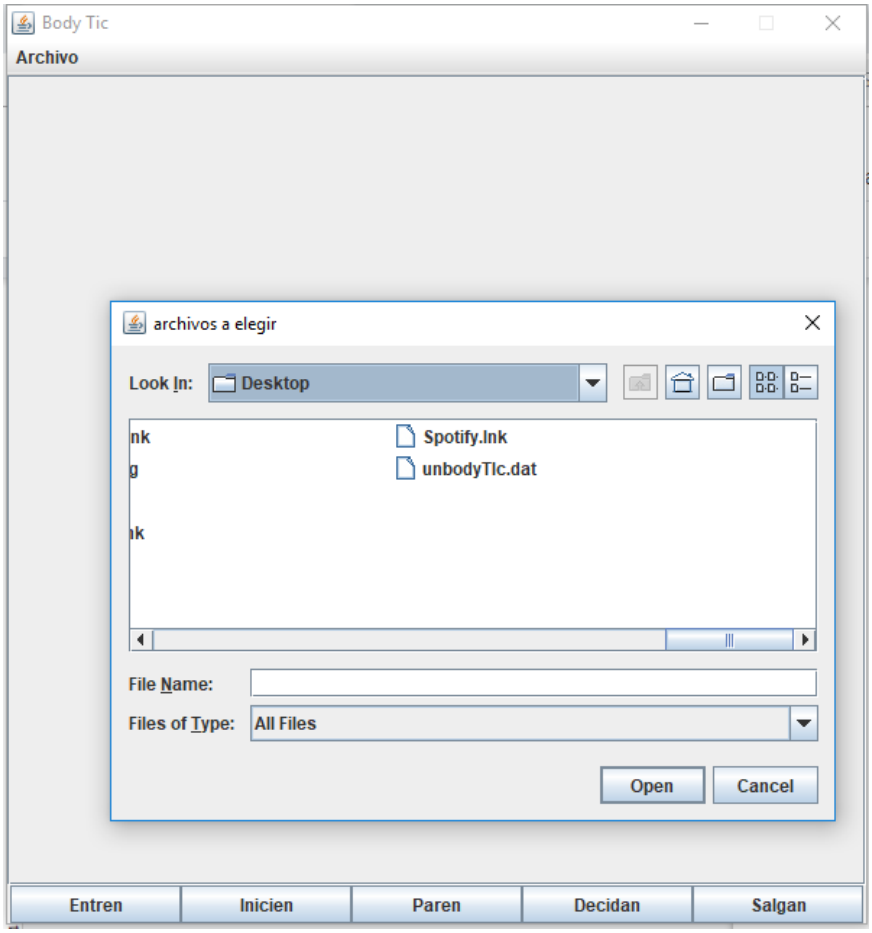
1. Construyan el método `opcionSalvar` que une de forma adecuada la capa de presentación con la capa de aplicación. Usen un `FileChooser` y atiendan la excepción. Ejecuten la aplicación probando las diferentes opciones del `FileChooser` y capturen una pantalla significativa.



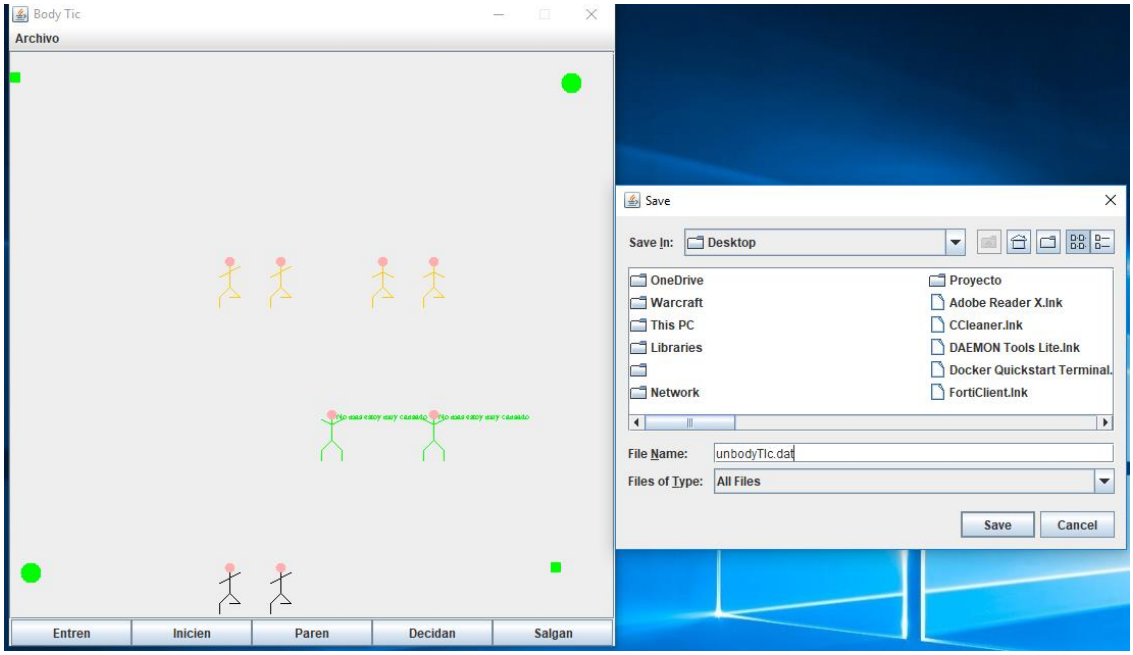
2. Construyan el método `salve` que ofrece el servicio de guardar en un archivo el estado actual del `bodyTIc`.
3. Validen este método guardando el `bodyTIc` inicial después de dos clics como `unbodyTIc.dat`. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?

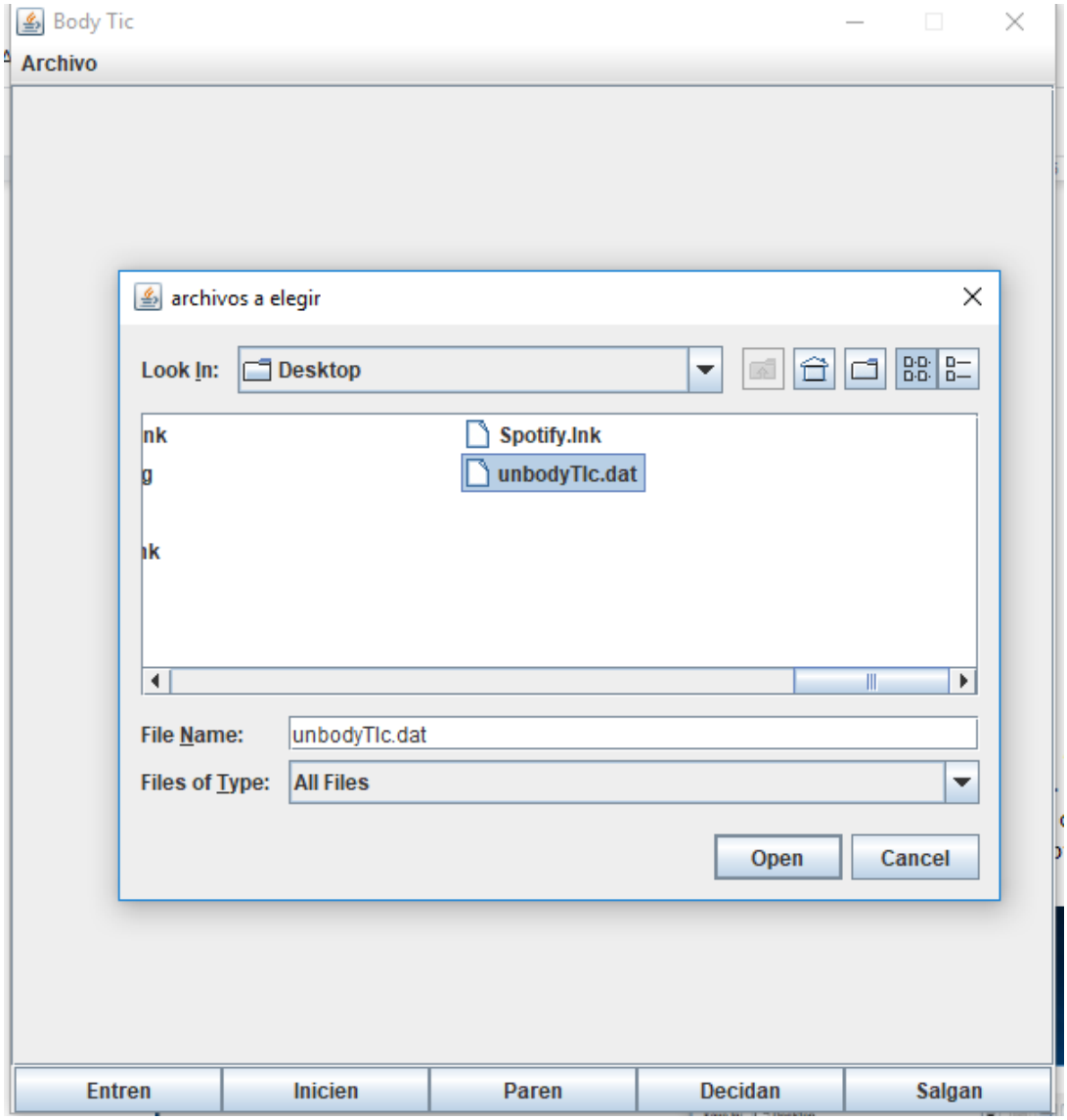


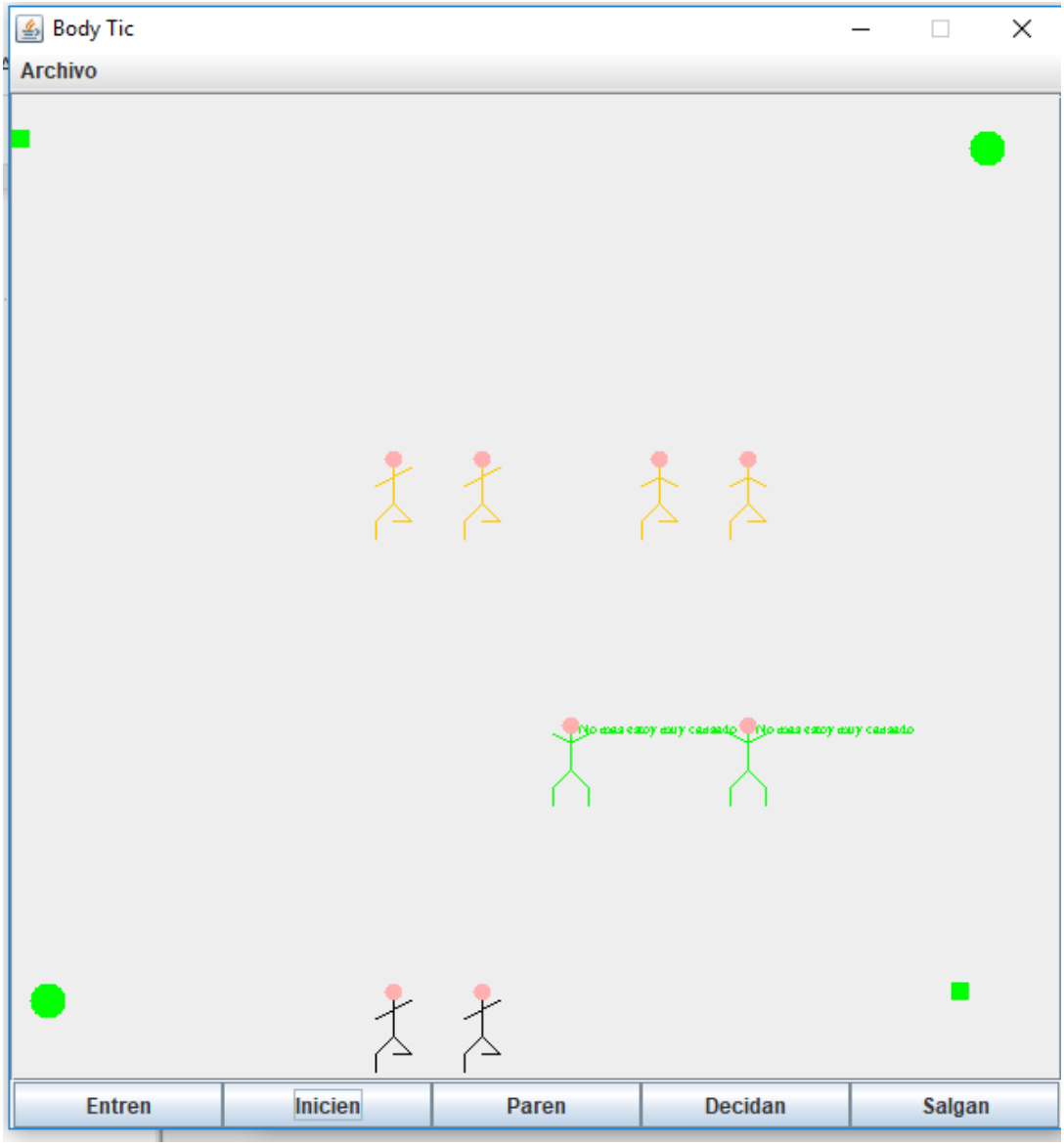
4. Construyan el método `opcionAbrir` que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación probando las diferentes opciones del `FileChooser` y capturen una pantalla significativa.



5. Construyan el método `abra` que ofrece el servicio de leer un `bodyTIc` de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general.
6. Realicen una prueba de aceptación para este método iniciando la aplicación, creando una nueva situación en el bodyTIc y abriendo el archivo `unbodyTIc.dat`. Capturen imágenes significativas de estos resultados.

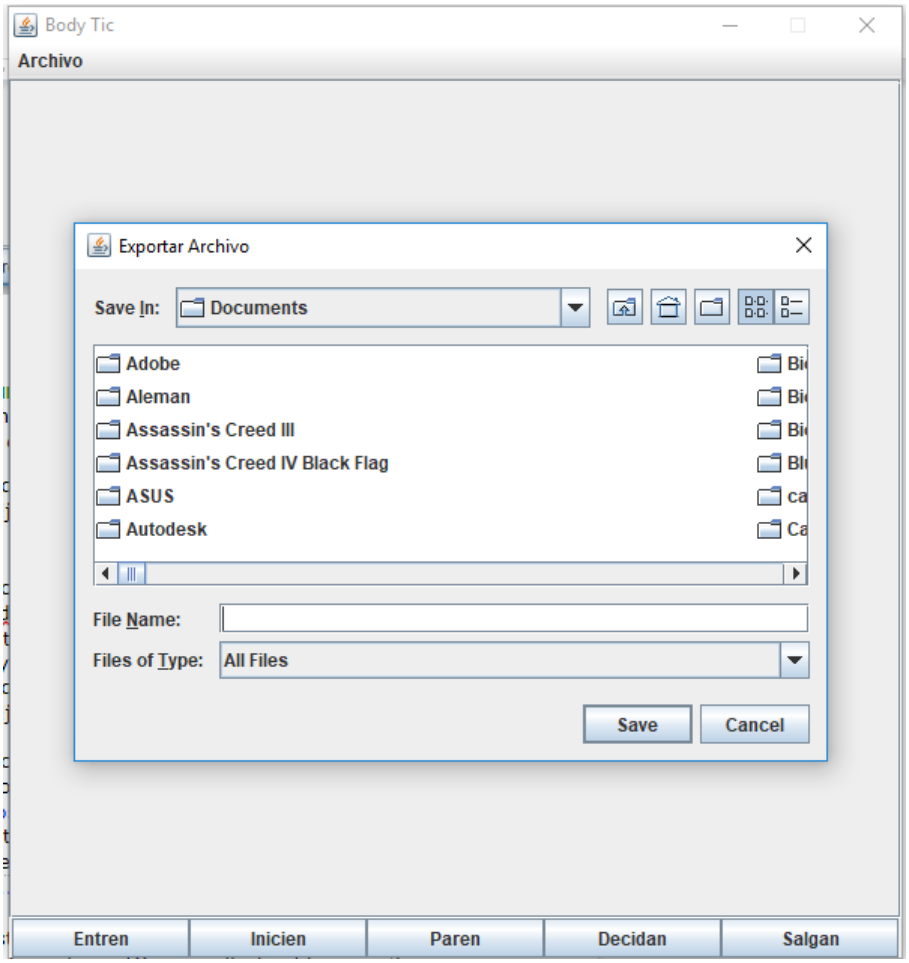



Body Tic
Archivo
archivos a elegir
Look In:
Desktop
nk
g
ık
Spotify.lnk
unbodyTlc.dat
File Name:
unbodyTlc.dat
Files of Type:
All Files
Open
Cancel
Entren
Inicien
Paren
Decidan
Salgan



Implementando importar y exportar
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]
Estas operaciones nos van a permitir importar información de un `bodyTIc` desde un archivo de texto y exportarlo. Los nombres de los archivos de texto deben tener como apellido .txt

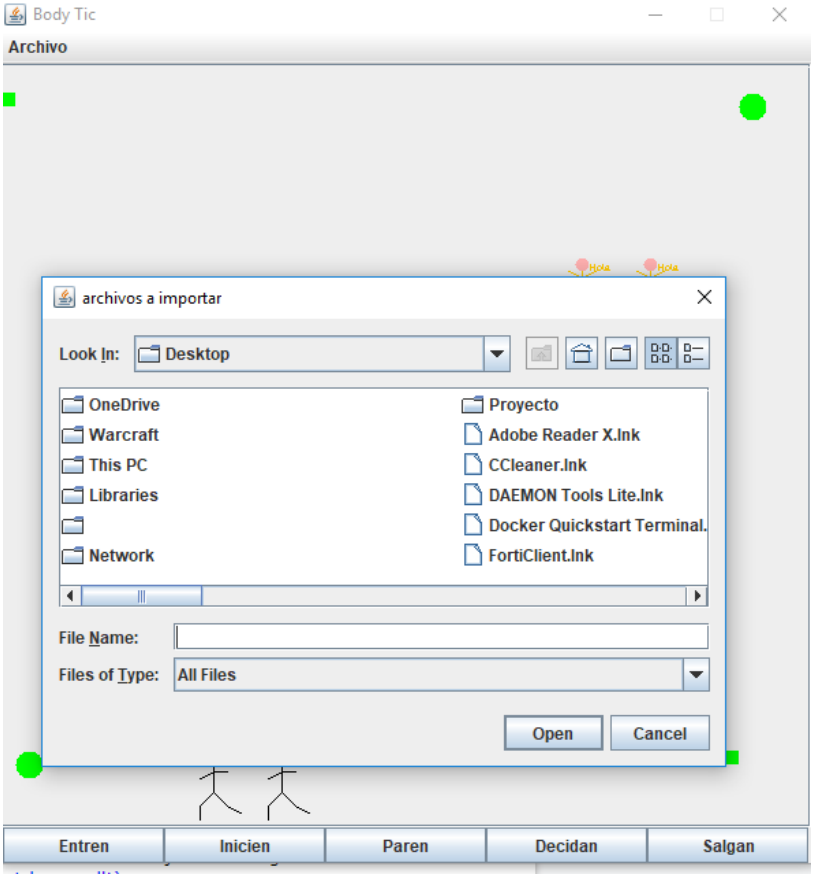
1. Construyan el método `opcionExportar` que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación y capturen una pantalla significativa.



2. Construyan el método `exporte` que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado del bodyTIc actual.
3. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como `unbodyTIc.txt`. Editen el archivo y analicen los resultados. ¿Qué pasó?

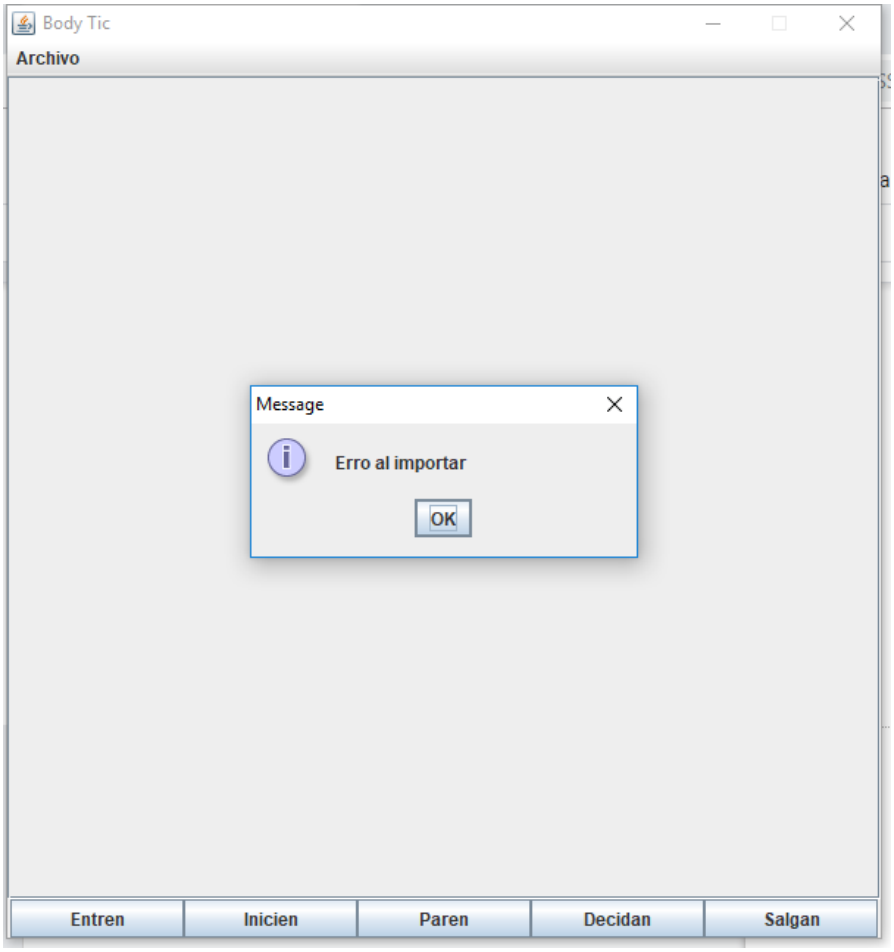
   Salen distintos errores, ya que pueden salir errores de sintaxis.

4. Construyan el método `opcionImportar` que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación y capturen una pantalla significativa.

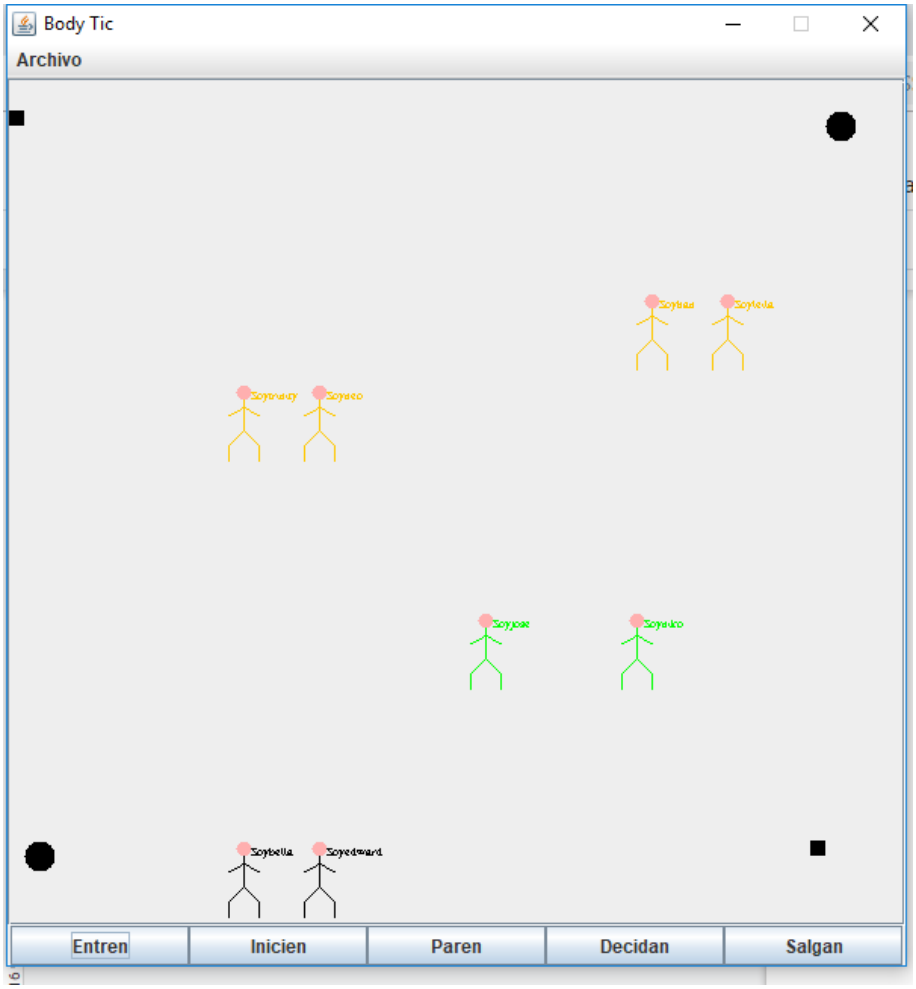


5. Construyan el método `importe` que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considere un mensaje de error general. (Consulten en la clase String los métodos trim y split)

6. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a `unbodyTIc.txt`. saliendo, entrando, creando un nuevo bodyTIc e importando el archivo `otrobodyTIc.txt`.. ¿Qué resultado obtuvieron? Capturen la pantalla final.

- 

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

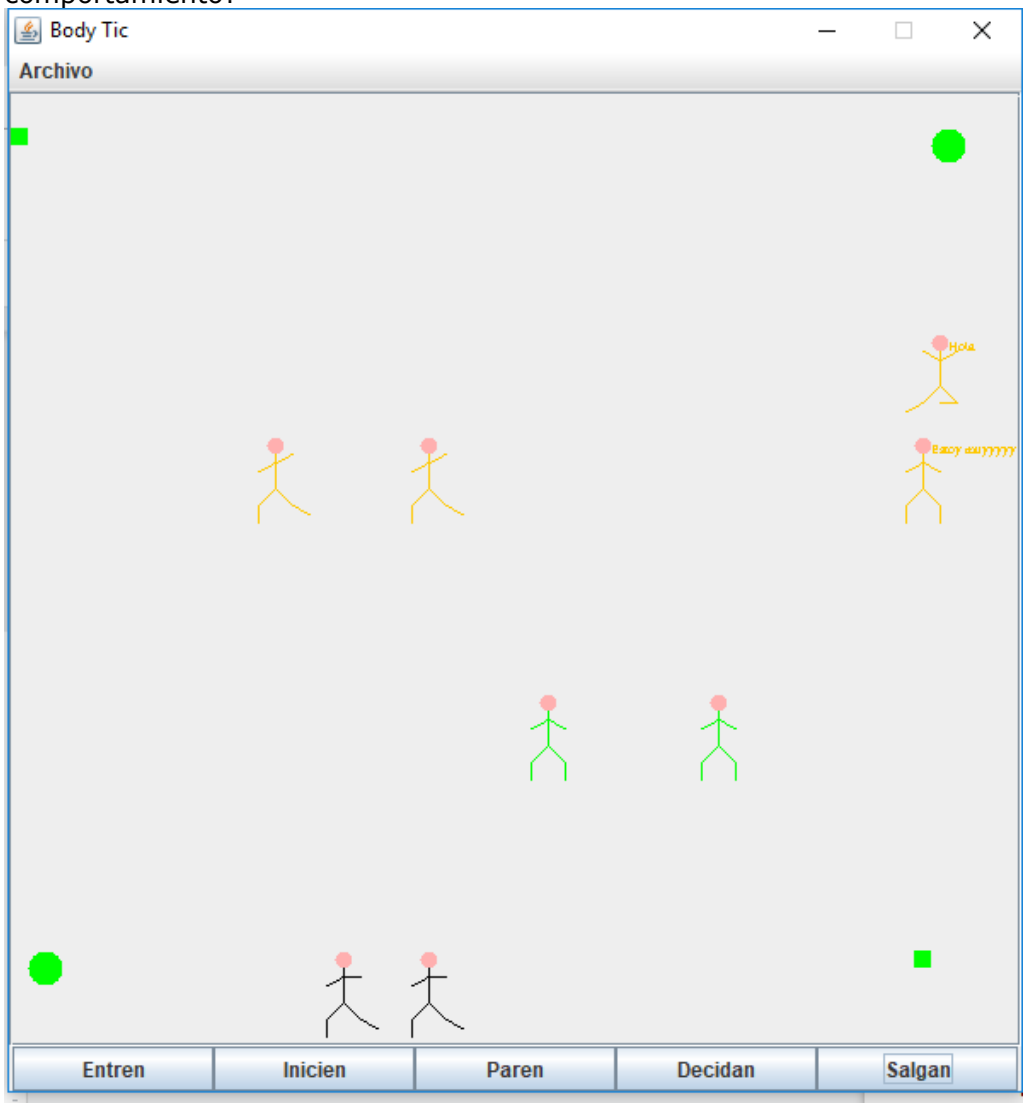


Obtuvimos un archivo de texto exportado esto quiere decir que los objetos se abren en el programa común y corriente con sus condiciones iniciales.

Analizando comportamiento

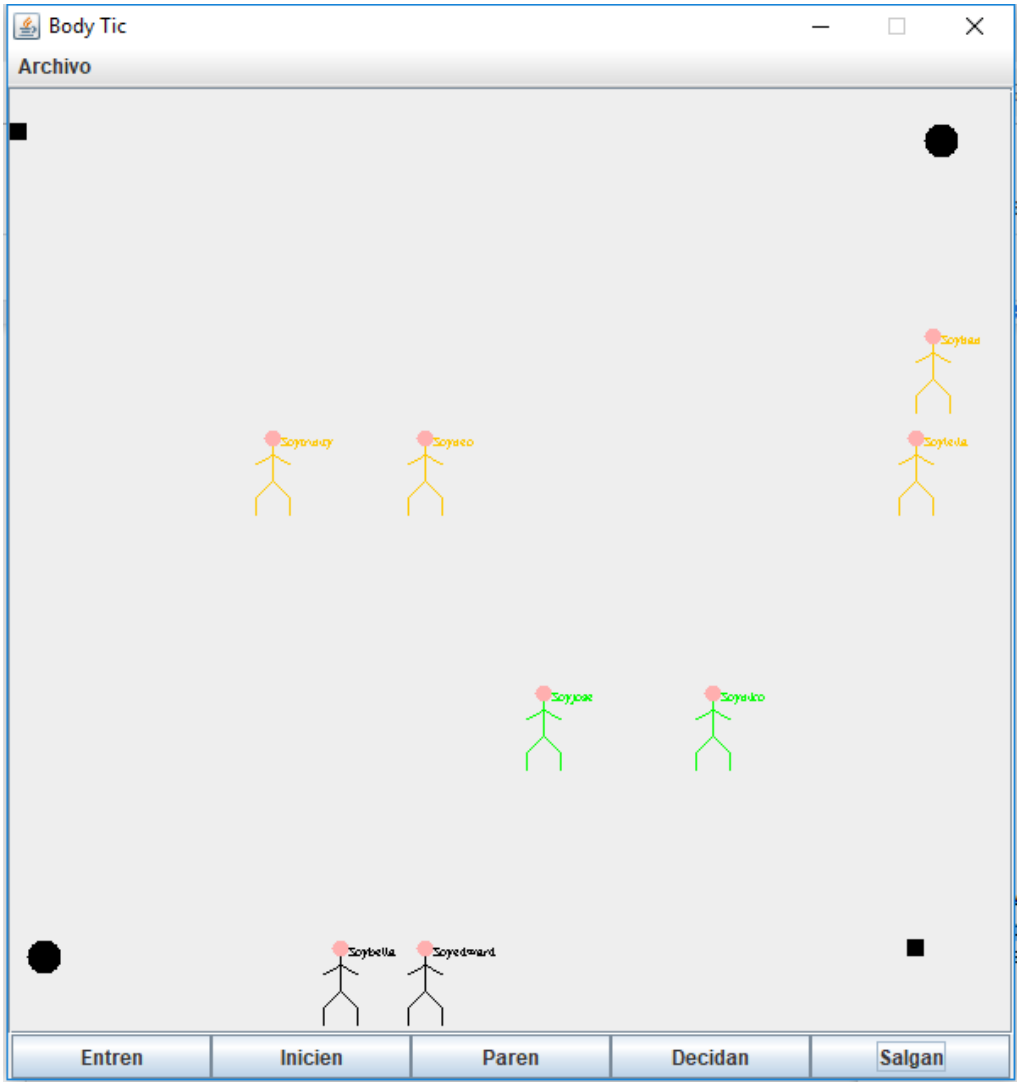
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

1. Ejecuten la aplicación, tres clics, salven a un archivo cualquiera y ábranlo. ¿Describa el comportamiento?



Al momento de guardar nuestro documento prueba.dat, se guarda los objetos en la última posición que quedaron. En este caso al momento de abrirlo se abre el archivo en la última acción en la que se guardó.

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

3. 

Se exporta un documento prueba.txt

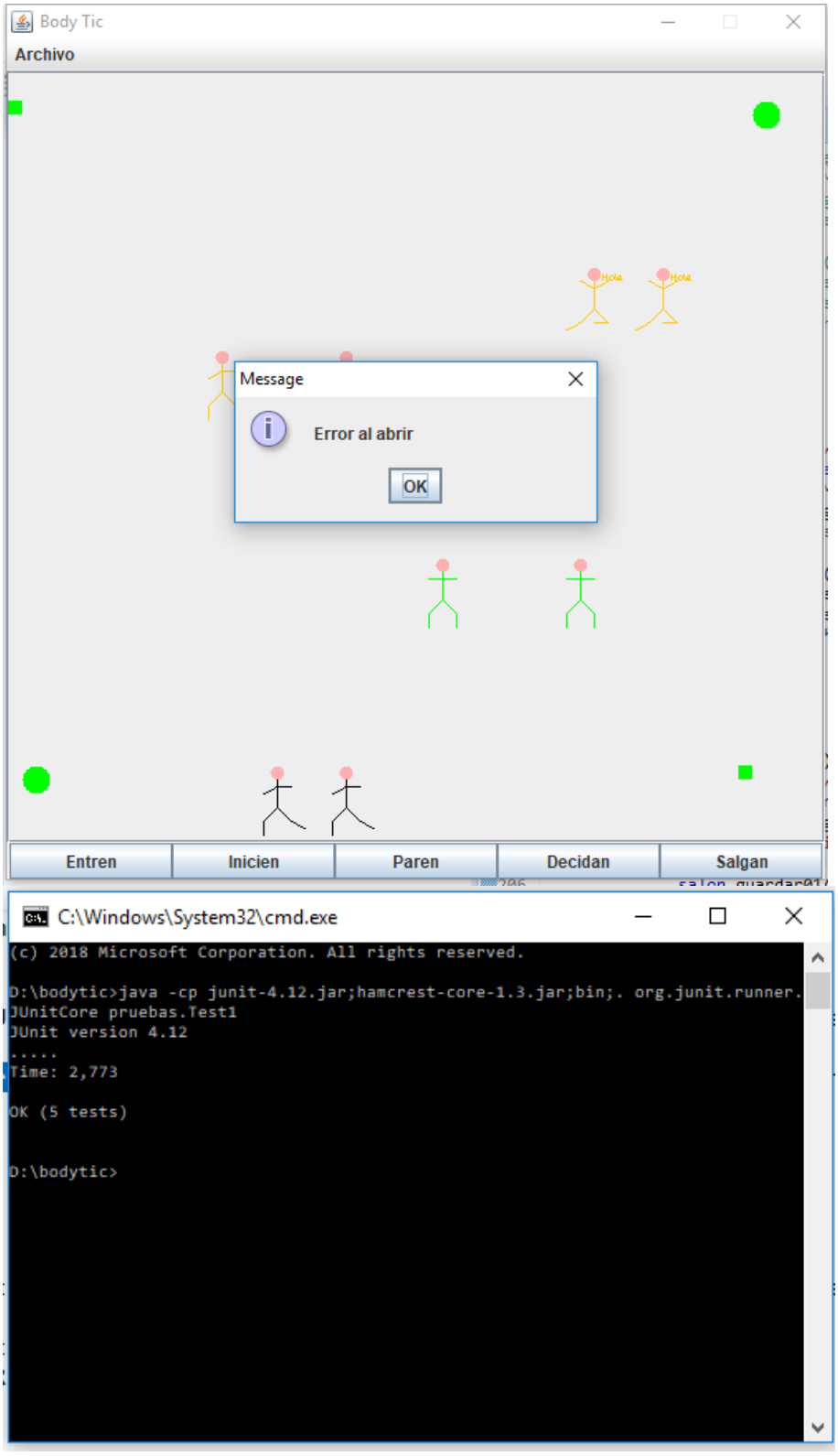
4. ¿Qué diferencias ven el comportamiento 1. y 2.? Expliquen los resultados.

   La diferencia es que cuando se guarda, quedan en los últimos movimientos en los cuales se guardó el documento. En cambio cuando se exporta e importa el documento se abre como uno nuevo.

Perfeccionando salvar y abrir

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

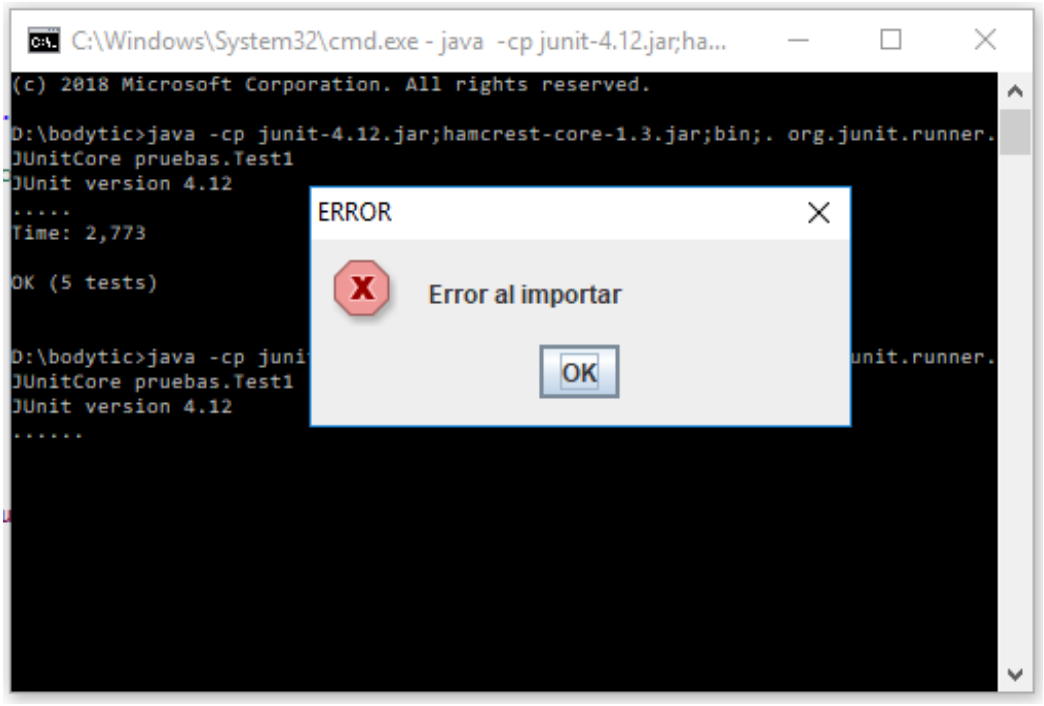
1. Copien las versiones actuales de `abra` y `salve` y renómbrenlos como `abra01` y `salve01`
2. Perfeccionen el manejo de excepciones de los métodos `abra` y `salve`.
3. Realicen una prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.



Perfeccionando importar y exportar.

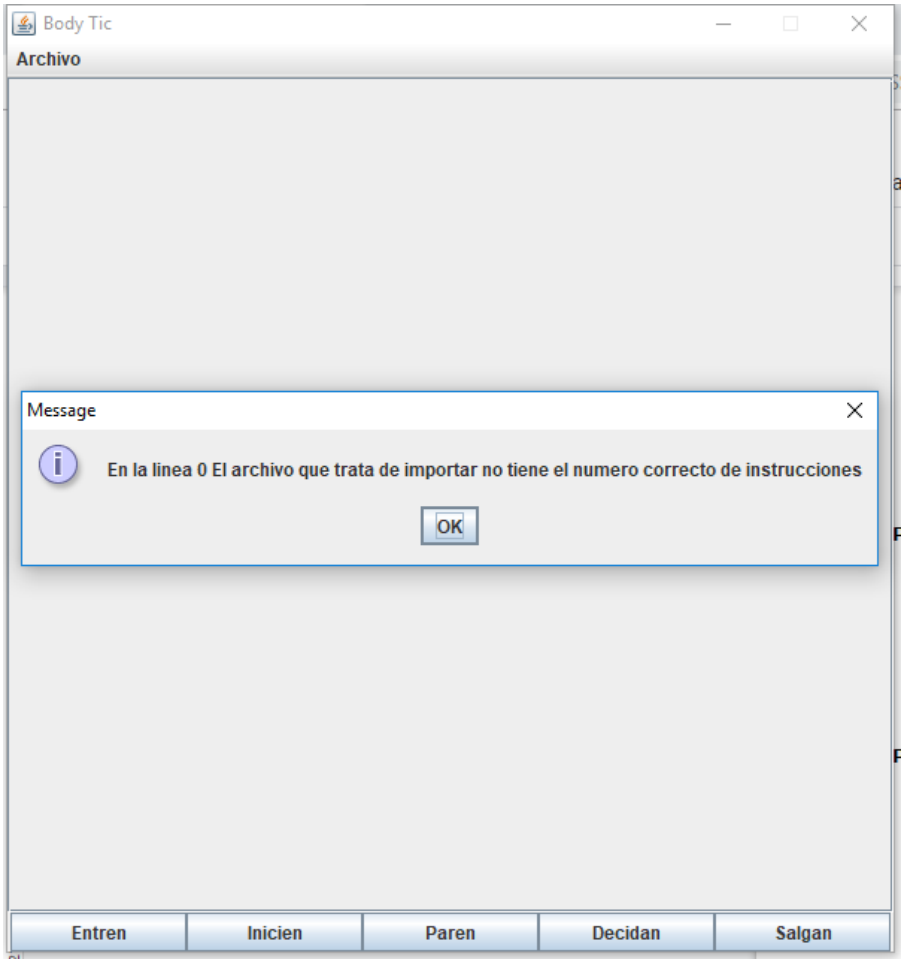
[En lab06.doc, *.asta , bodyTIcErr.txt *.java] [NO OLVIDEN BDD y MDD]

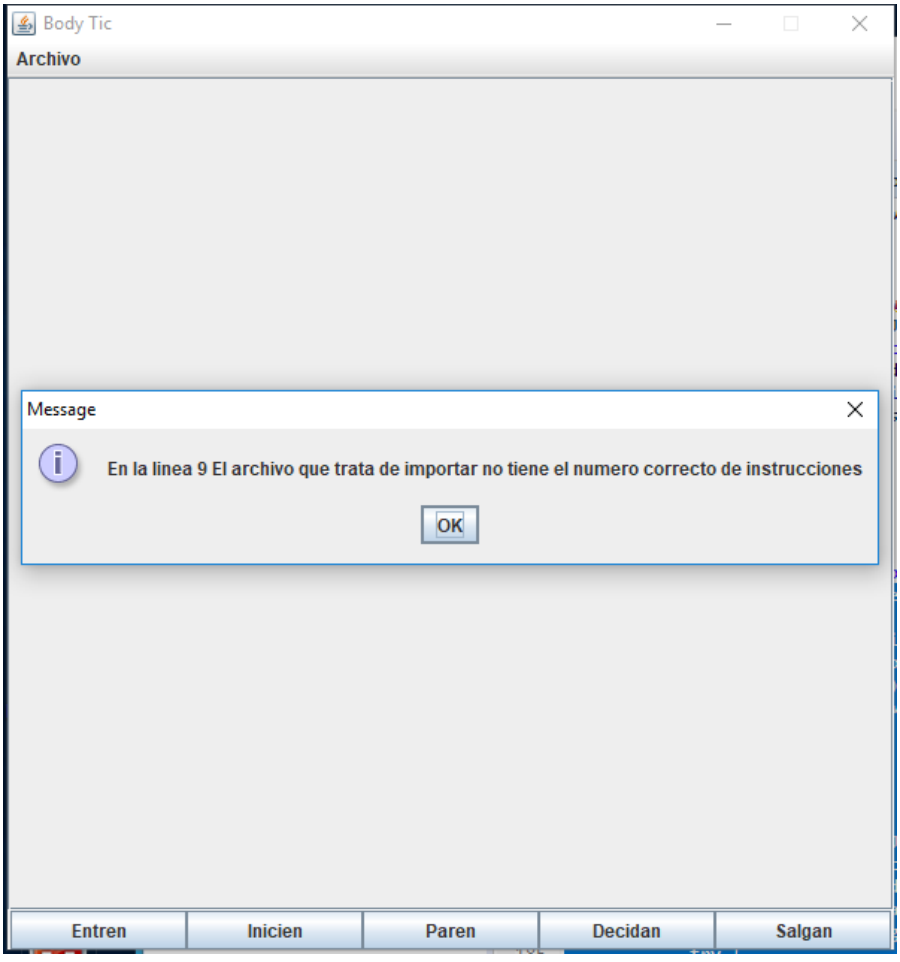
1. Copien las versiones actuales de `importe` y `exporte` y renómbrenlos como `importe01` y `exporte01`
2. Perfeccionen el manejo de excepciones de los métodos `importe` y `exporte`.
3. Realicen una prueba de aceptación para validar cada una de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.



Perfeccionando importar. Hacia un minicompilador.
[En lab06.doc, *.asta , bodyTIcErr.txt *.java] [NO OLVIDEN BDD y MDD]

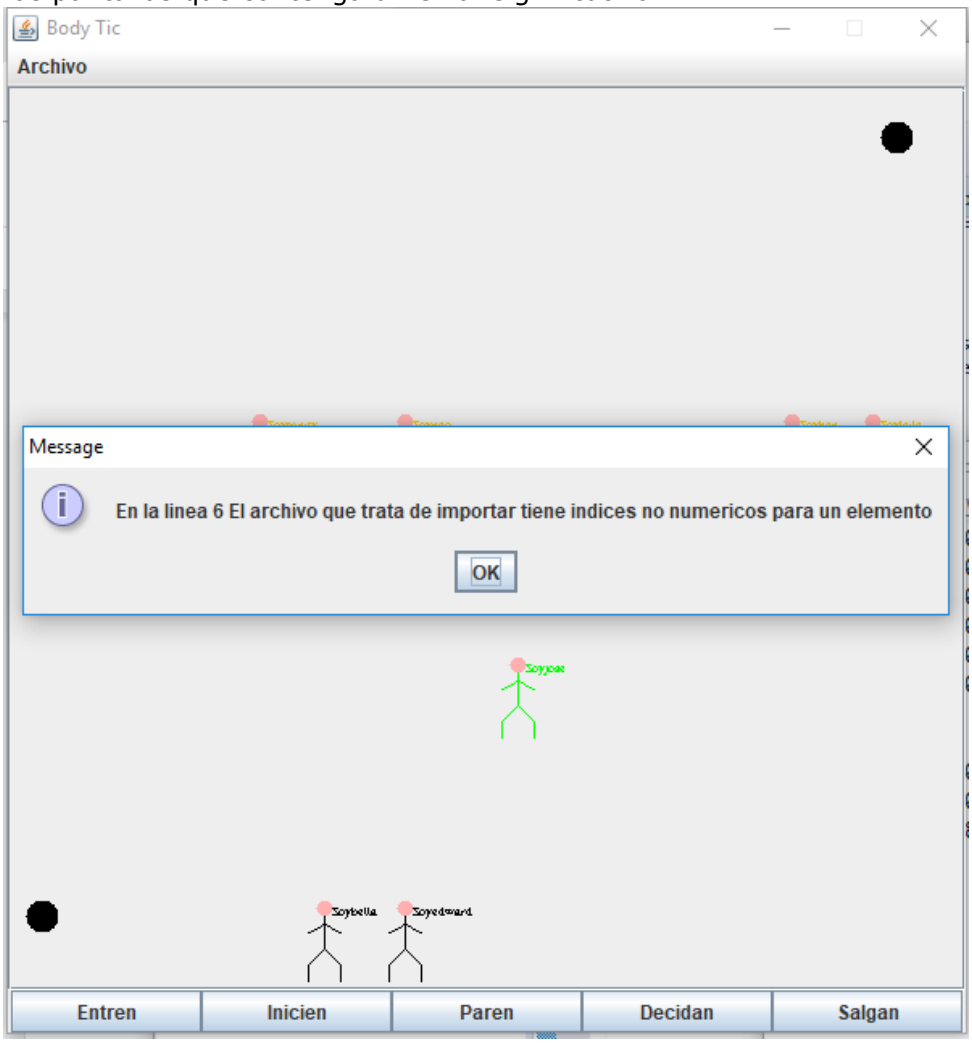
1. Copien las versiones actuales de `importe` y `exporte` y renómbrenlos como `importe02` y `exporte02`
2. Perfeccionen el método `importe` para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palabra que tiene el error y causa de error.
3. Escriban otro archivo con errores, llámelo bodyTIcErr.txt, para ir arreglándolo con ayuda de su "importador". Presente las pantallas que contengan los errores.


Body Tic
Archivo
Message
En la linea 0 El archivo que trata de importar no tiene el numero correcto de instrucciones
OK
Entren
Inicien
Paren
Decidan
Salgan


Body Tic
Archivo
Message
En la linea 9 El archivo que trata de importar no tiene el numero correcto de instrucciones
OK
Entren
Inicien
Paren
Decidan
Salgan

Perfeccionando importar. Hacia un minicompilador flexible.
[En lab06.doc, *.asta , bodyTIcFlex.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de `importe` y `exporte` y renómbrenlos como `importe03` y `exporte03`
2. Perfeccionen los métodos `importe` y `exporte` para que pueda servir para cualquier tipo de elementos creados en el futuro (Investiguen cómo crear un objeto de una clase dado su nombre)
3. Escriban otro archivo de pruebas, llámelo `bodyTIcErrG.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)
   22 horas
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
   Casi completo por excepción de unas pocas cosas que no pudimos culminar
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
   La programación a par fue la práctica XP mas utilizada por nuestro grupo.
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
   El poder entender el funcionamiento de las funciones de java como lo son el exportar e importar con su funcionamiento basico.
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
   La forma correcta de guardar e importar, ya que al principio generaba muchos errores

cada vez que se utilizaba una de estas.

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

   Tuvimos buena comunicación frente a los diversos temas y nos comprometemos a tener un poco mas de conocimiento frente a el lenguaje java.