# PROGRAMACIÓN ORIENTADA A OBJETOS I/O 2019-02 Brayan Burgos – Daniel Alfonso 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 como,  
importar y exportar.  
3. Controlar las excepciones generadas al trabajar con archivos.  
4. Experimentar las prácticas XP: Refactor whenever and wherever possible.  
All code must pass all unit tests before it can be released. -

# DESARROLLO

## Preparando

En este laboratorio vamos a extender el proyecto teatroColon 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]  
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, abra, exporte, importe). Los métodos deben simplemente propagar una teatroColonExcepcion 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.

3. CONTROLADOR: Construyan los controladores correspondientes a estas acciones. Para esto creen el método prepareAccionesMenu y los métodos base del controlador (opcionSalver,opcionAbir, opcionExportar, opcionImportar, opcionIniciar, opcionCerrar), Estos últimos métodos, por ahora, sólo deben llamar directamente el método correspondiente de la capa de aplicación. Capturen una pantalla significativa.

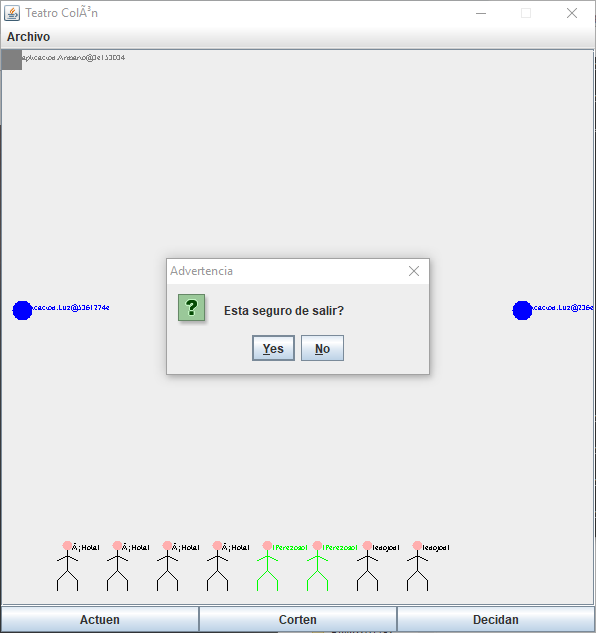
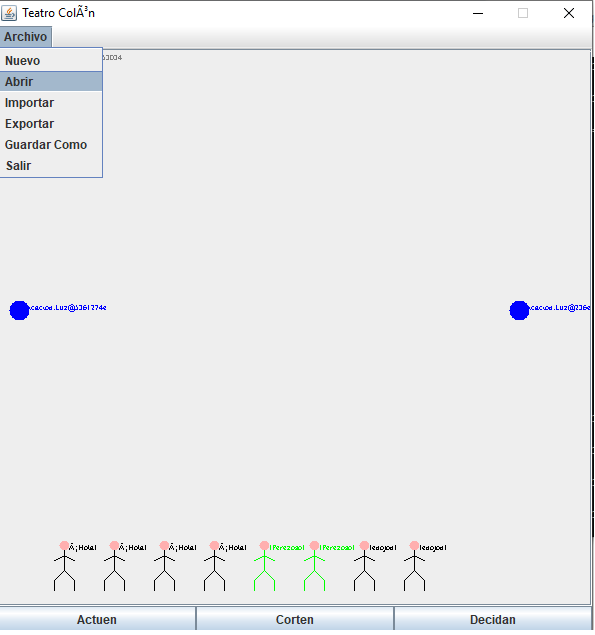
## Implementando salir e iniciar

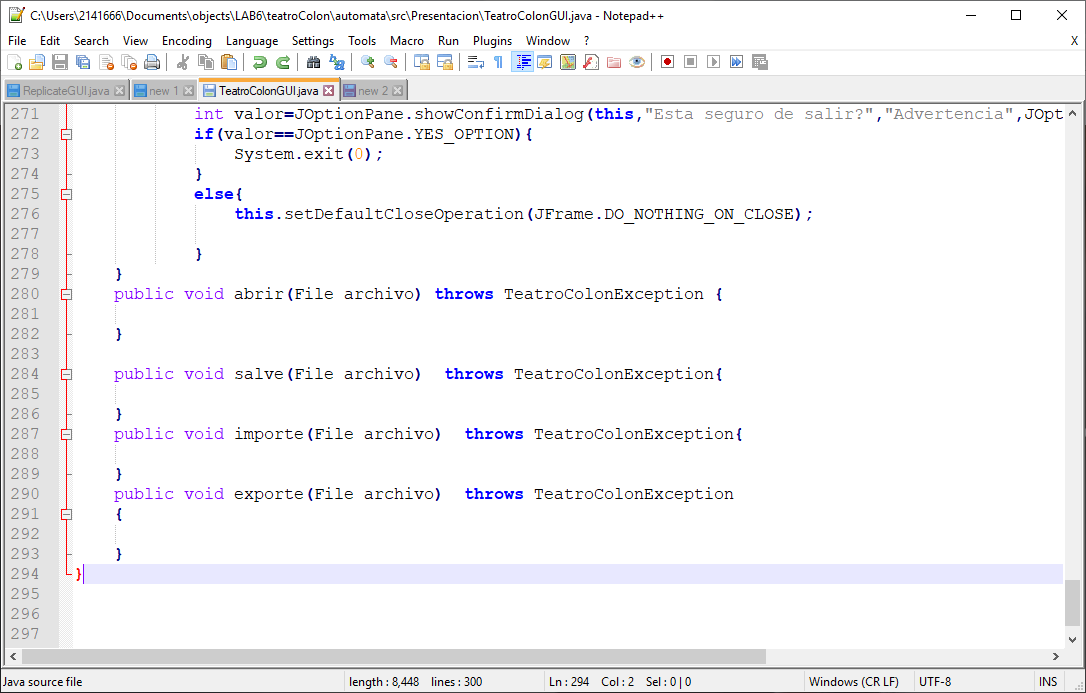
[En lab06.doc, \*.asta y \*.java] [NO OLVIDEN BDD y MDD]

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 teatroColon. Capturen una pantalla significativa.





## Implementando salvar y abrir

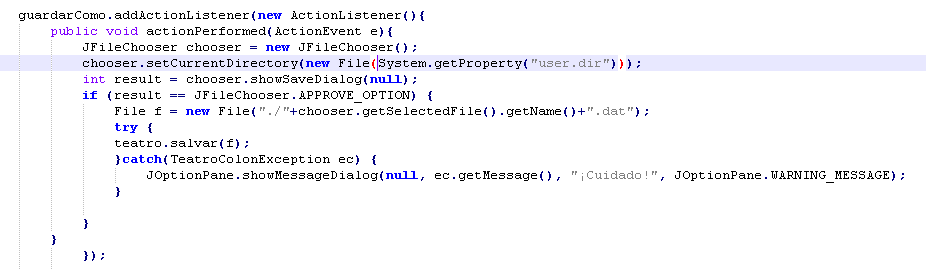
[En lab06.doc, \*. asta y \*.java] [NO OLVIDEN BDD y MDD]

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

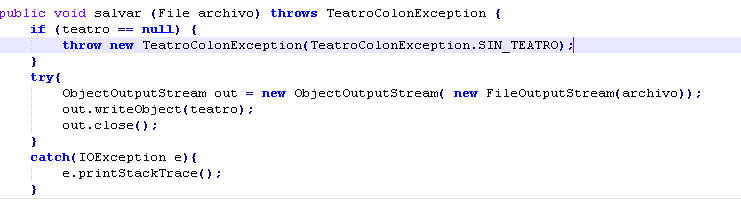
1. Construyan el método opcionSalvar que une de forma adecuada la capa de presentación con la capa de aplicación. Usen un FileChooser y atiendan la excepción. Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una pantalla significativa.

Además de lo que se muestra, las clases que usan Teatro y Teatro en la capa de aplicación deben implementar la interfaz serializable e importar java.io.Serializable

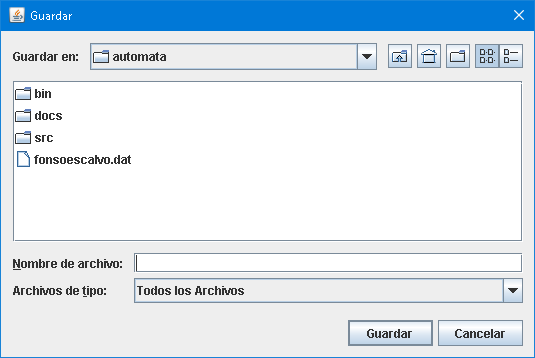
En Presentación



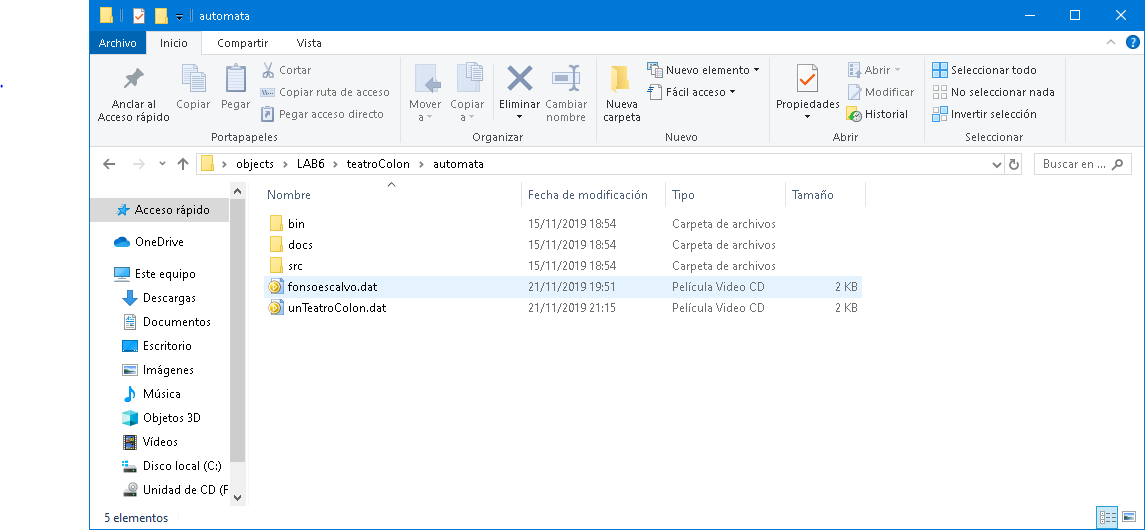
En Aplicación



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

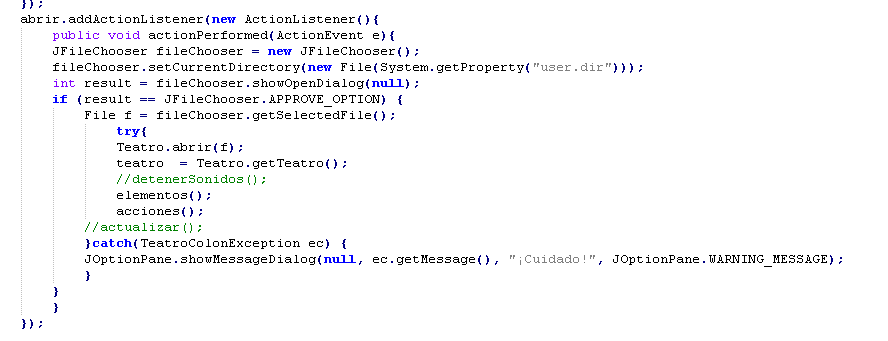


Damos en archivo, guardarComo y guardaremos el archivo con el nombre unTeatroColon.dat y podemos ver que el archivo pesa 2 KB

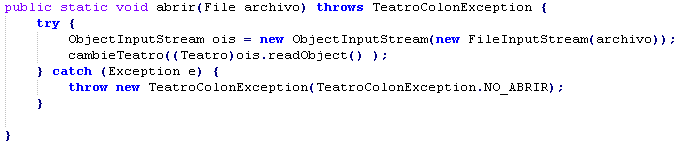


4. Construyan el método opcionAbrir que une de forma adecuada la capa de  
presentación con la capa de aplicación. Ejecuten la aplicación probando las diferentes opciones del FileChooser y capturen una pantalla significativa.

En Presentación



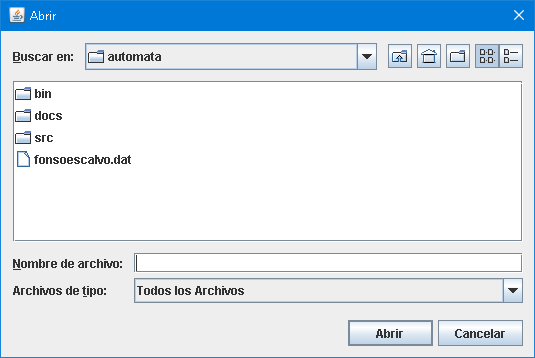
En Aplicación

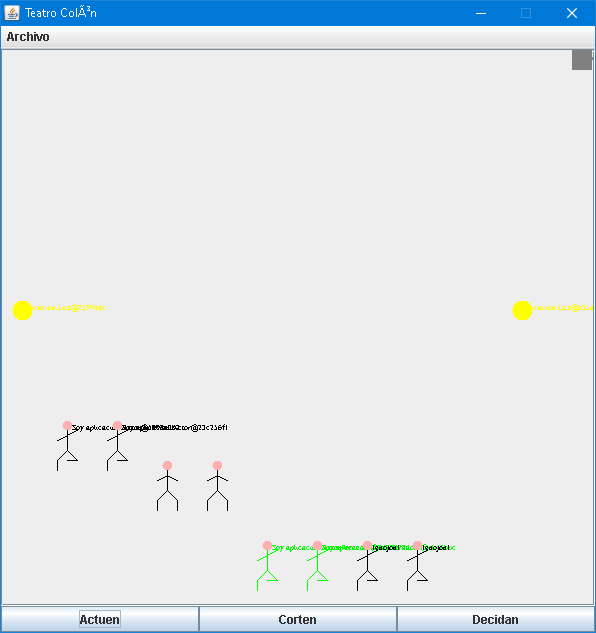


Antes de abrir



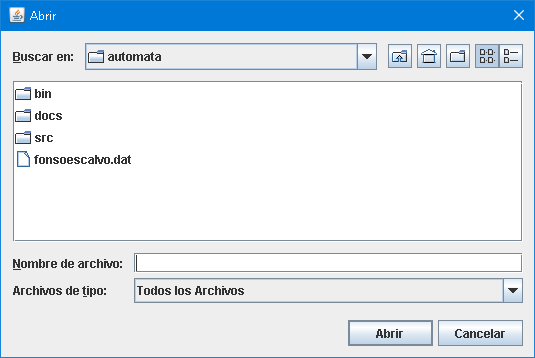
Al abrir archivo creado:

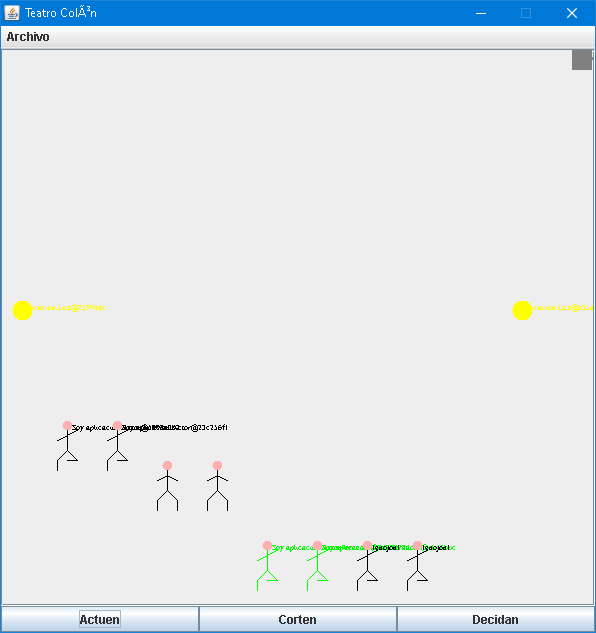




1. Construyan el método abra que ofrece el servicio de leer un teatroColon de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general.

Al abrir archivo creado:





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

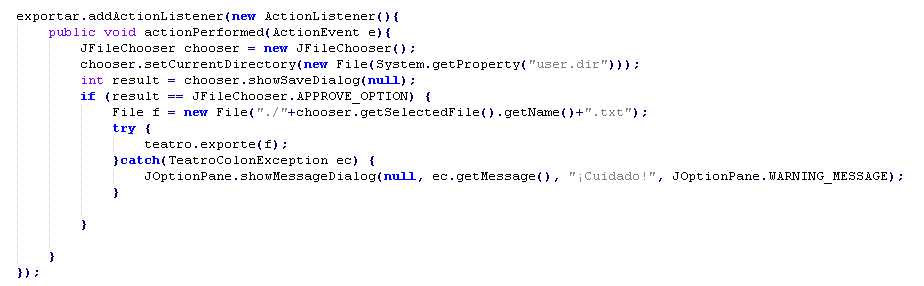
## Implementando importar y exportar

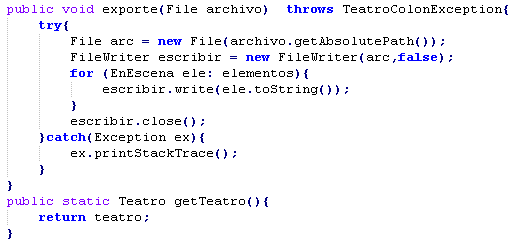
[En lab06.doc, \*. asta y \*.java] [NO OLVIDEN BDD y MDD]

Estas operaciones nos van a permitir importar información de un teatroColon desde un archivo  
de texto y exportarlo. Los nombres de los archivos de texto deben tener como apellido .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.  
Normal 20 20 Romeo  
Luz 50 50

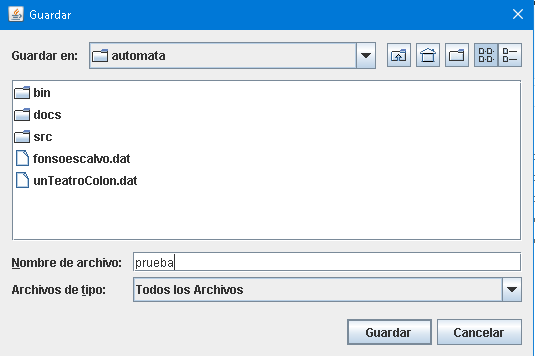
1. Construyan el método opcionExportar que une de forma adecuada la capa de presentación con la capa de aplicación. Ejecuten la aplicación y capturen una pantalla significativa.

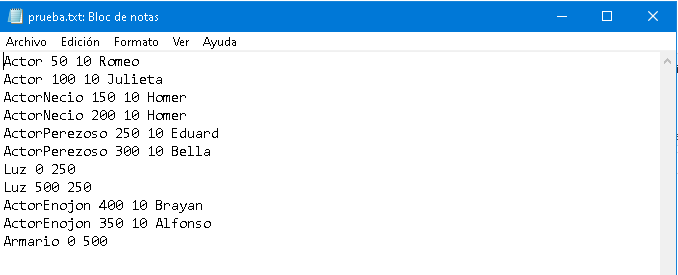
En presentación como vemos llamamos el método de teatro que se llama exporte y pasamos el archivo.   


En aplicación   


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

Ahora al ejecutar la aplicación y exportar obtendremos el siguiente resultado

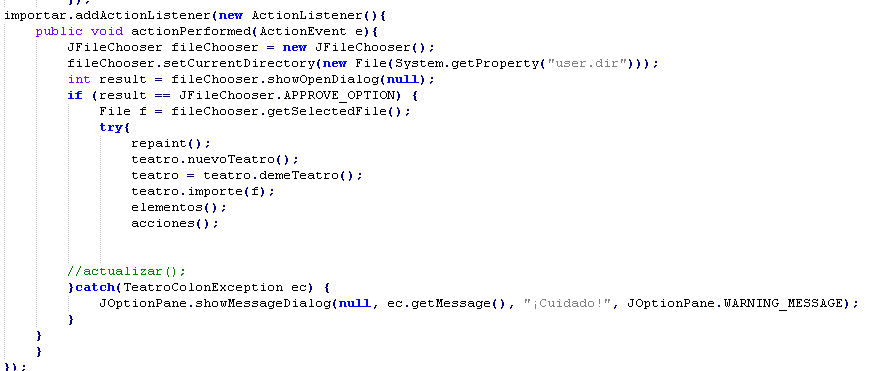


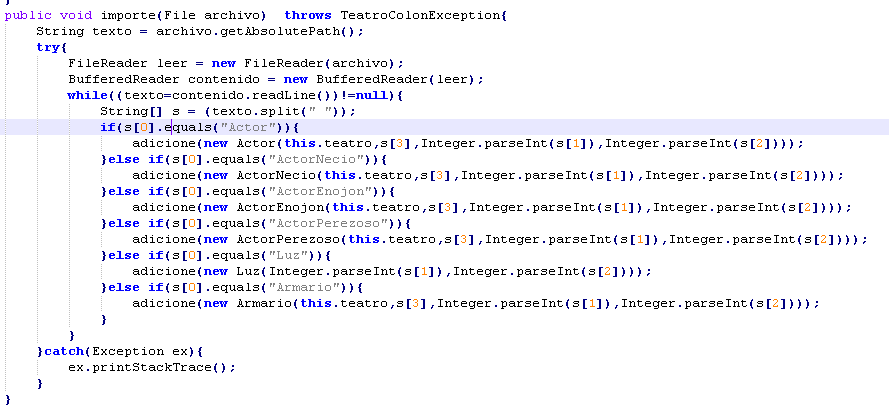


3. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como unteatroColon.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?

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.

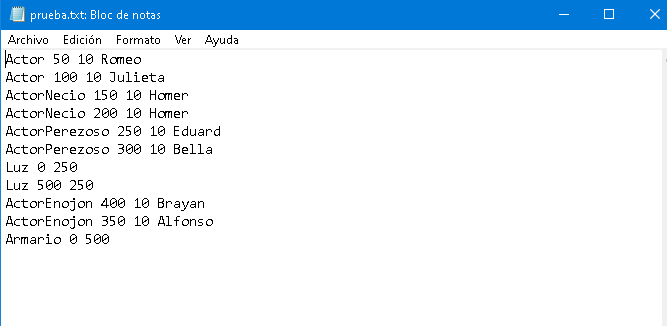
En Presentación

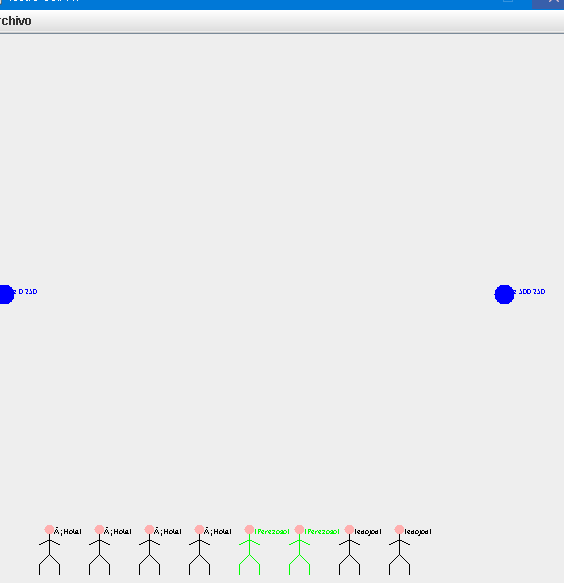


En aplicación   
 

1. 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)

El método trim no toma en cuenta las líneas en blanco de comienzo y fin y el Split guarda en una lista los elementos separados por comas, espacios o etc.

Al importar un teatro:  




Se observa que se importa bien el teatro de forma correcta se deja el archivo de prueba en los recursos.

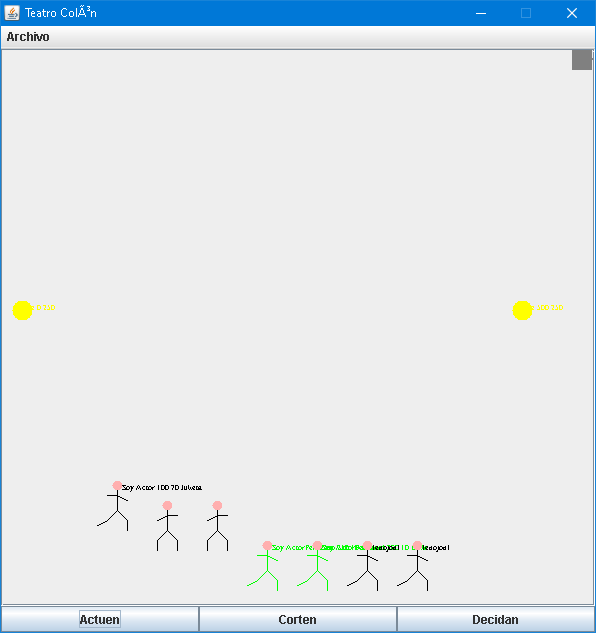
6. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a unteatroColon.txt. saliendo, entrando, creando un nuevo teatroColon e importando el archivo otroteatroColon.txt. ¿Qué resultado obtuvieron? Capturen la pantalla final.

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

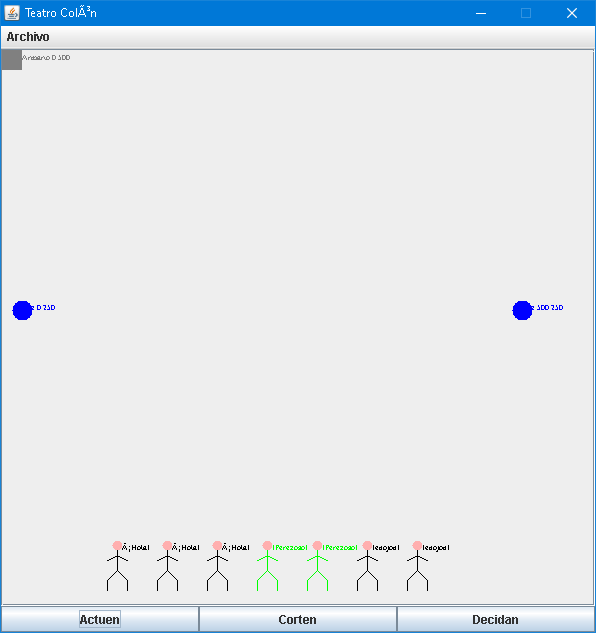
## Analizando comportamiento

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

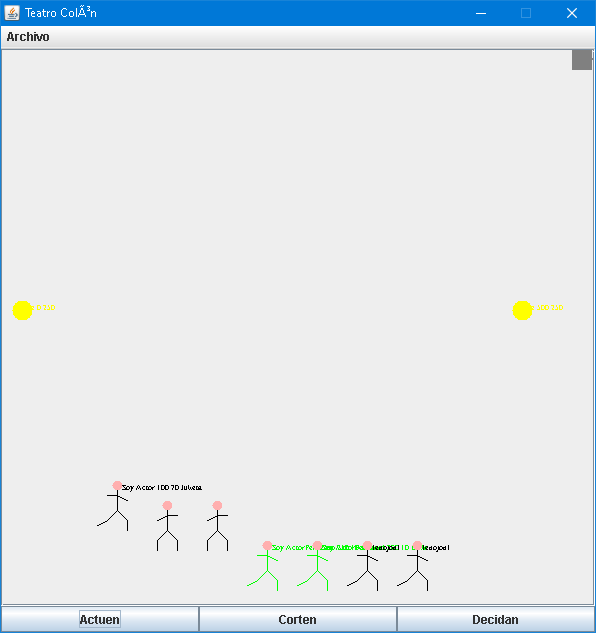
1. Ejecuten la aplicación, den tres clics, salven a un archivo cualquiera y ábranlo. Describan el comportamiento



Lo guardamos damos a nuevo

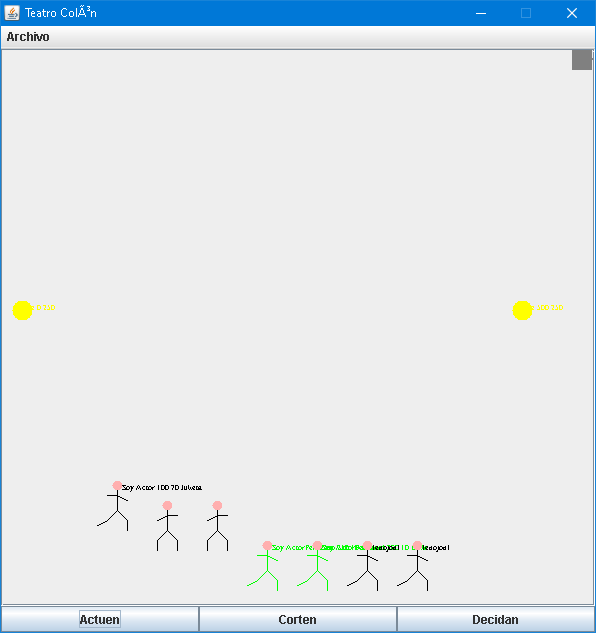


Abrimos el archivo prueba1.dat y obtenemos



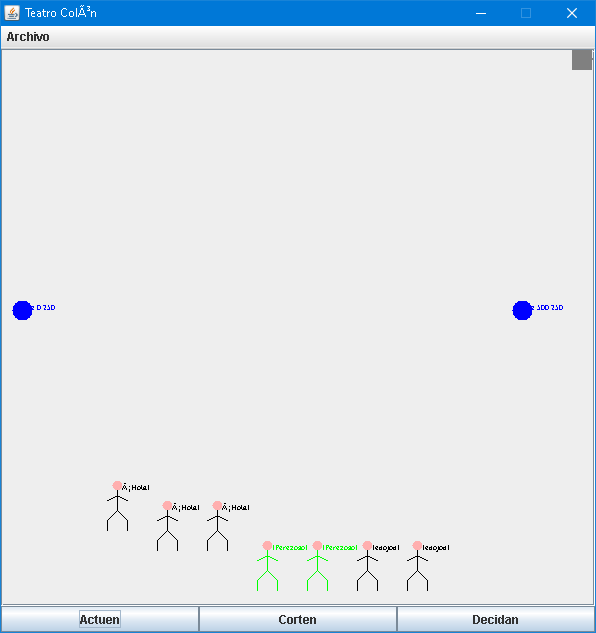
En el comportamiento de guardar, estamos “guardando el estado de un objeto” en un archivo cuando lo abrimos leemos este objeto y los casteamos al objeto que deseamos por asi decirlo para volver a “tenerlo” guardamos la instancia del objeto como tal sus atributos, etc.

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



Damos 3 click y exportamos a nuestro archivo prueba1.txt

Al importarlo obtenemos lo siguiente



Podemos observar que en la forma de exportar, guardamos la construcción del objeto es decir su posición, su nombre, pero no su instancia como tal al momento de importarlo volvemos a crear el objeto desde cero, en cambio a diferencia de cuando abrimos un archivo o un objeto quedara en la misma instancia en la que lo guardamos.

## 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 detallando los errores.

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, teatroColonErr.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 detallando los errores.

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 mini compilador.

[En lab06.doc, \*. asta, teatroColonErr.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 teatroColonErr.txt, para ir arreglándolo con ayuda de su “importador”. Presente las pantallas que contengan los errores.

## Perfeccionando importar. Hacia un mini compilador flexible.

[En lab06.doc, \*. asta, teatroColonFlex.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 teatroColonErrG.txt, para probar la flexibilidad. Presente las pantallas que contenga un error significativo.

# RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el proyecto por cada uno de ustedes? (Horas/Hombre)

El tiempo invertido fueron 12 horas por cada uno

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

El mayor logro para nosotros es, que cada vez vamos entendiendo más lo que es la programación orientada a objetos y logramos aplicar, herencias, superclases, serializacion y demás

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

El tiempo de entrega a pesar de haber sido extendido, tuvimos bastante problema con el tiempo que pudimos invertir, además del manejo de la sintaxis de las estructuras pedidas para avanzar de manera óptima el proceso del LABORATORIO 6

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

Como equipo siempre escuchamos la opinión del otro y tratamos de llegar a un acuerdo sobre lo acordado. La utilización de la programación en parejas. Nos comprometemos a esforzarnos y seguir con la disciplina que logramos en este proyecto para lograr los objetivos satisfactoriamente

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

Las dos prácticas que conocemos las utilizamos, las iteraciones las utilizamos para dividir en partes más pequeñas los problemas y las más útil fue la programación a pares que ayuda a entenderse y lograr mejores trabajos de calidad en pareja