**PROGRAMACIÓN ORIENTADA A OBJETOS**   
**Herencia e interfaces**   
**ADEMAS Java desde consola**   
**2019-02**   
**Laboratorio 3/6**

**Contexto:** En esta aplicación se presenta el escenario de un teatro, en nuestro caso el Teatro Colón.

**Conociendo: [En lab03.doc y TeatroColon.asta ]**

**1. En el directorio descarguen los archivos contenidos en TeatroColon.zip. Revisen el código de la aplicación**

**a) ¿Cuántos paquetes tiene?** Tiene dos paquetes: aplicación y presentación.

**b) ¿Cuántas clases tiene en total? ¿Cuántas tienen fuentes?** Tiene 6 clases en total. Tienen fuentes: la clase teatro, la clase persona, la clase actor, la clase TeatroColonGUI y la clase enEscena.

**c) ¿Cuál es la clase ejecutiva? ¿Por qué?** La clase ejecutiva es la interfaz, porque esta clase es la que se encarga de ejecutar el programa, la cual debe estar en el paquete de presentación y en este caso es la única.

**2. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?**

* En el paquete Presentación hay una clase, la clase TeatroColonGUI, esta clase ofrece los siguientes servicios, abre una ventana con los botones actúen, decidan y corten, sin embargo, ninguno hace nada dado que las funciones de teatro que utilizan los botones no han sido implementadas.
* En el paquete Aplication hay 4 clases y una interfaz, la clase luz no ha sido implementada por lo que solo existe en este momento, la interfaz enEsena la implementa la clase actor y es utilizada por la clase Teatro, en este momento, el teatro solo se crea, dado que ninguno de sus servicios ha sido implementado, en la clase Persona todos sus servicios han sido implementados y presenta los siguientes, muevaBrazo, muevaPierna, getColor, getPosicionBrazo, getPosicionPierna, muevase, getPosicionX y getPosicionY, la sub clase de persona actor también cumple sus servicios, siendo estos mensaje, actue, corte, puedeMover

**Arquitectura general: [En lab03.doc y TeatroColonasta]**

**1. Consulte el significado de las palabras package e import de java. ¿Qué es un paquete? ¿Para qué sirve?**

Un **package**(paquete) es una agrupación de clases afines. Al crear un package, veremos que creamos una carpeta. La organización del proyecto será por tanto similar a la organización de archivos: en un paquete podremos tener por ejemplo clases de tipo A, en otro, clases de tipo B y así sucesivamente. A su vez, un paquete puede contener subpaquetes: por ejemplo el paquete A puede contener a los subpaquetes A.1, A.2 y A.3. Una clase puede definirse como perteneciente a un package y puede usar otras clases definidas en ese o en otros packages. Los packages delimitan el espacio de nombres (space name).

El nombre de una clase debe ser único dentro del package donde se define. Dos clases con el mismo nombre en dos packages distintos pueden coexistir e incluso pueden ser usadas en el mismo programa. También resulta importante por la implicación que los packages tienen en la visibilidad y acceso del código entre distintas partes de un programa. Cuando se referencia cualquier clase dentro de otra se asume, si no se indica otra cosa, que ésta otra está declarada en el mismo package.

Supongamos que definimos la clase JefeDeAdministracion perteneciente al package Directivos. Esta clase usa la clase Persona. El compilador asume que Persona pertenece también al package Directivos, y tal como está hecha la definición, para que la clase Persona sea accesible (conocida) por el compilador, es necesario que esté definida en el mismo package. Si esto no fuera así, es necesario hacer accesible el espacio de nombres donde está definida la clase Persona a nuestra nueva clase. Esto se hace con la cláusula **import**.

En una clase puede haber tantas sentencias import como sean necesarias. Las cláusulas import se colocan después de la cláusula package (si es que existe) y antes de las definiciones de las clases. También es posible hacer accesibles los nombres de un package sin usar la cláusula import calificando completamente los nombres de aquellas clases pertenecientes a otros packages. Sin embargo, si no se usa import es necesario especificar el nombre del package cada vez que se usa el tipo Persona, lo cual puede resultar un tanto engorroso. La cláusula import en esta acepción simplemente indica al compilador dónde debe buscar clases adicionales cuando no pueda encontrarlas en el package actual y delimita los espacios de nombres y modificadores de acceso.

**2. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?**

Tenemos un proyecto en este caso llamado Teatro Colon que organiza su estructura separando la parte de aplicación de la presentación dejando un paquete distinto para cada una. Y en cada uno de estos paquetes encontramos sus respectivos directorios.

Además, en la carpeta del directorio existen 2 carpetas con el nombre de los paquetes, en las carpetas están las clases de los paquetes

**3. Inicie el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos. En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)**

**Arquitectura detallada: [En lab03.doc y TeatroColonasta]**

**1. Usando ingeniería reversa prepararen el proyecto para MDD. Presente el diseño estructural actual de la aplicación (diagrama de clases). Las clases de la capa de presentación sólo deben tener los elementos públicos.**

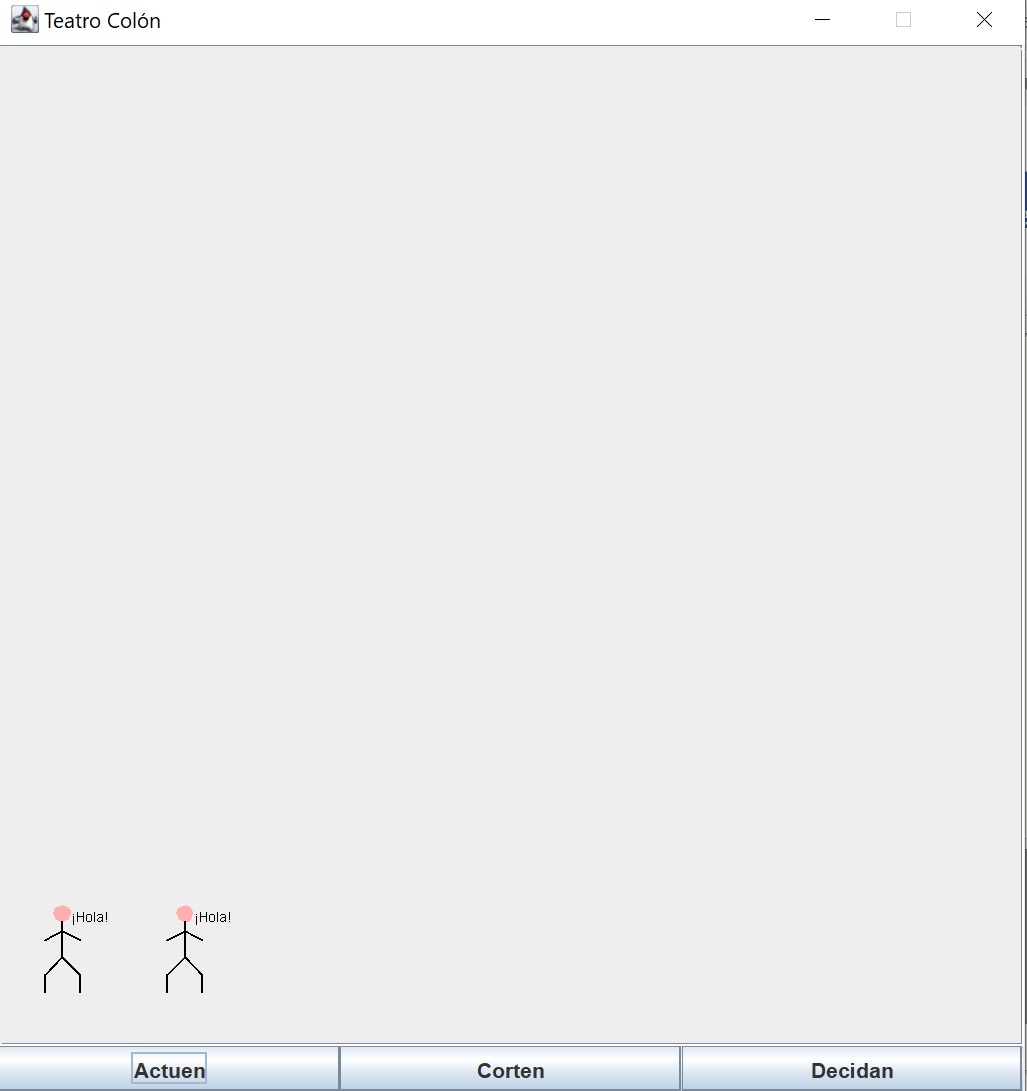
**2. Adicione en las fuentes la clase de pruebas necesaria para BDD. (No lo adicione al diagrama de clases) ¿En qué paquete debe estar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?** Deberían estar en el paquete de presentación porque este paquete es el responsable de ejecutar el proyecto en su totalidad. Debe estar asociado a la interfaz que es la clase ejecutiva del proyecto.

**Ciclo 1. Actúan y descansan los actores normales: [En lab04.doc y \*.java] (NO OLVIDE BDD - MDD)**

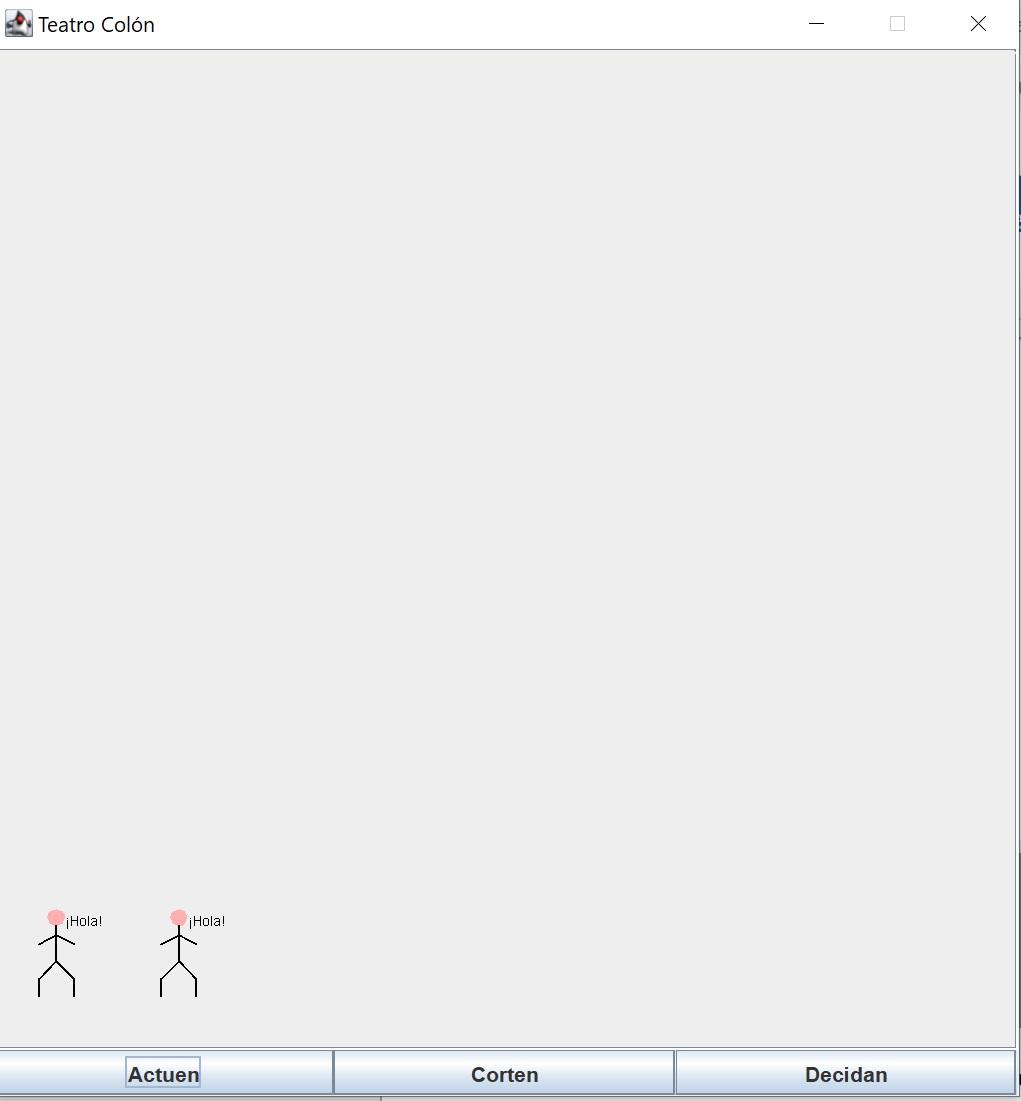
**1. Estudie la clase Teatro. ¿Qué tipo de colección se usa para albergar los elementos? ¿Puede recibir actores? ¿Por qué?** Un ArrayList el cual debe recibir actores debido a que hasta el momento es el único que implementa la interfaz enEscena.

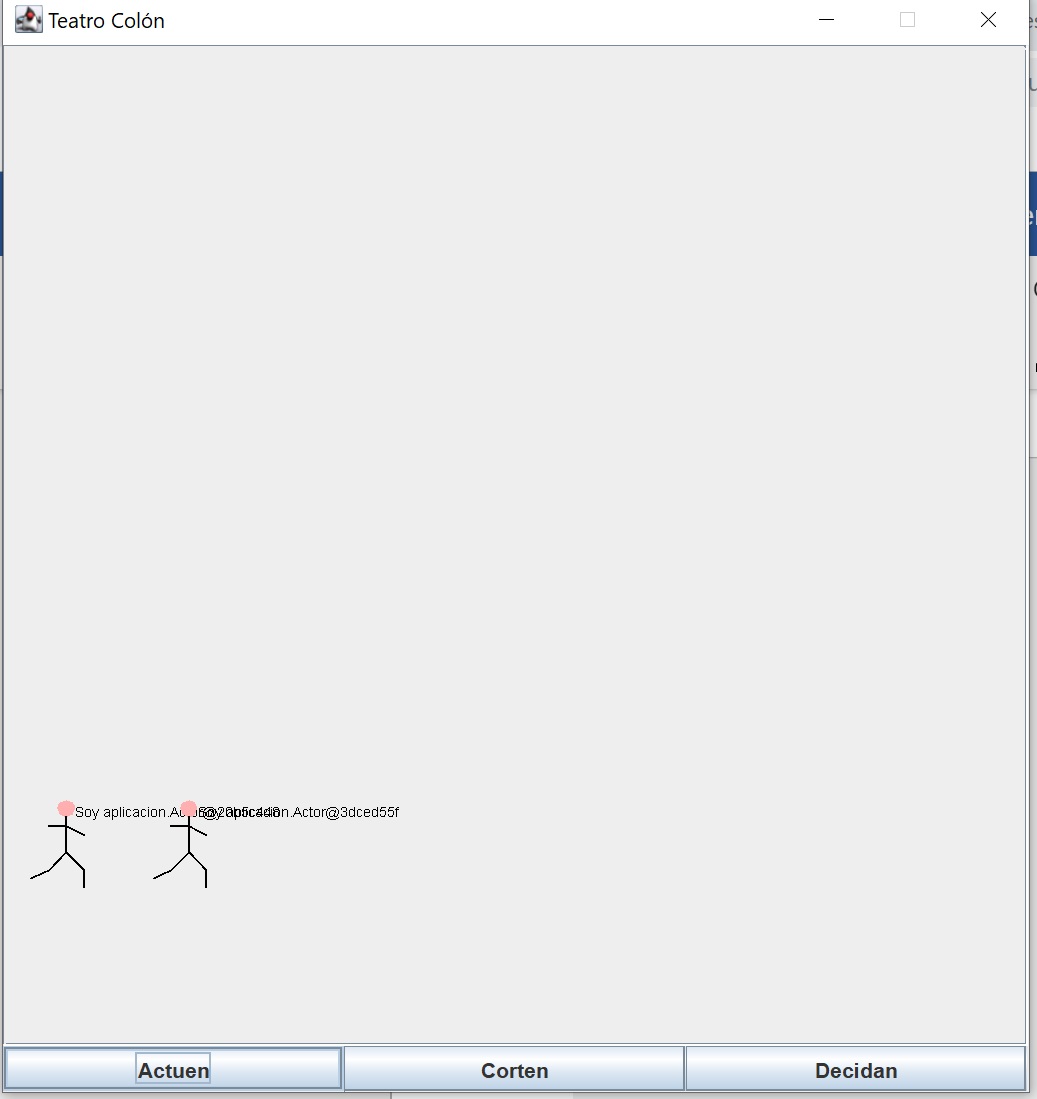
**2. Estudie el código de la clase Actor; ¿de qué color es? ¿qué palabras dice? ¿cómo entran en acción? ¿qué hacen cuando corta? ¿cómo deciden?** Es de color negro, dice “¡Hola!”, entran en acción con el metodo actue(), cuando se hace corte() se llama a los métodos: muevaBrazo(lo que le ingresa), muevaPierna(lo que le ingresa) (dos veces) y ahora dice “ ”. Decide con el método de la interfaz enEscena decida(), y deciden.

**3. En el método algunosEnEscena de la clase Teatro cree dos actores en diferentes posiciones y acondiciónelos al Teatro llámelos romeo y julieta. Ejecute el programa y capture la pantalla. ¿Qué pasa ahora? ¿Pídales que entren en acción? ¿Qué pasa? ¿Por qué?** Ahora al ejecutar si hace algo el programa, y nos muestra a nuestros dos nuevos actores diciendo “Hola” en las coordenadas dadas. Al pedirles que entren en acción, no hacen nada porque el actuen() no está implementado.

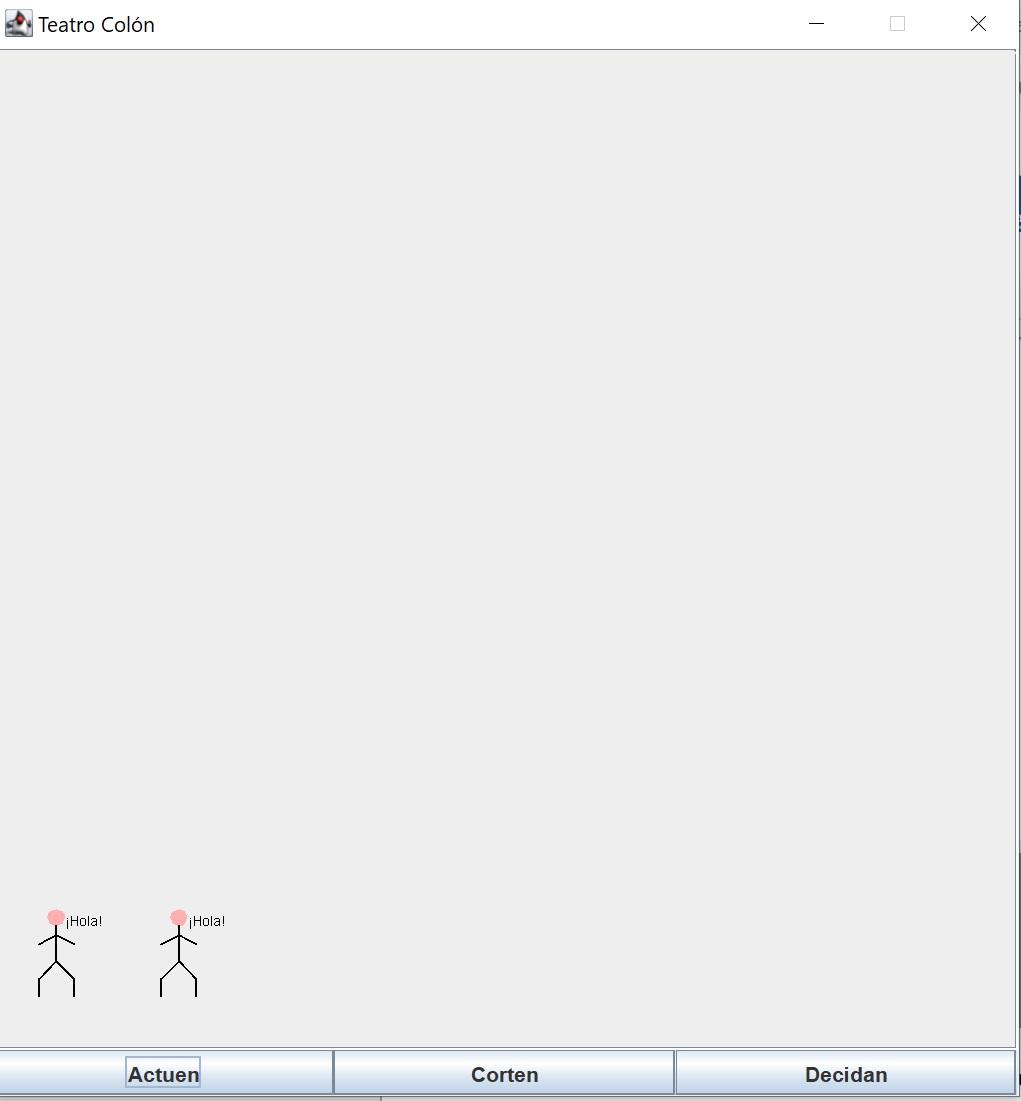


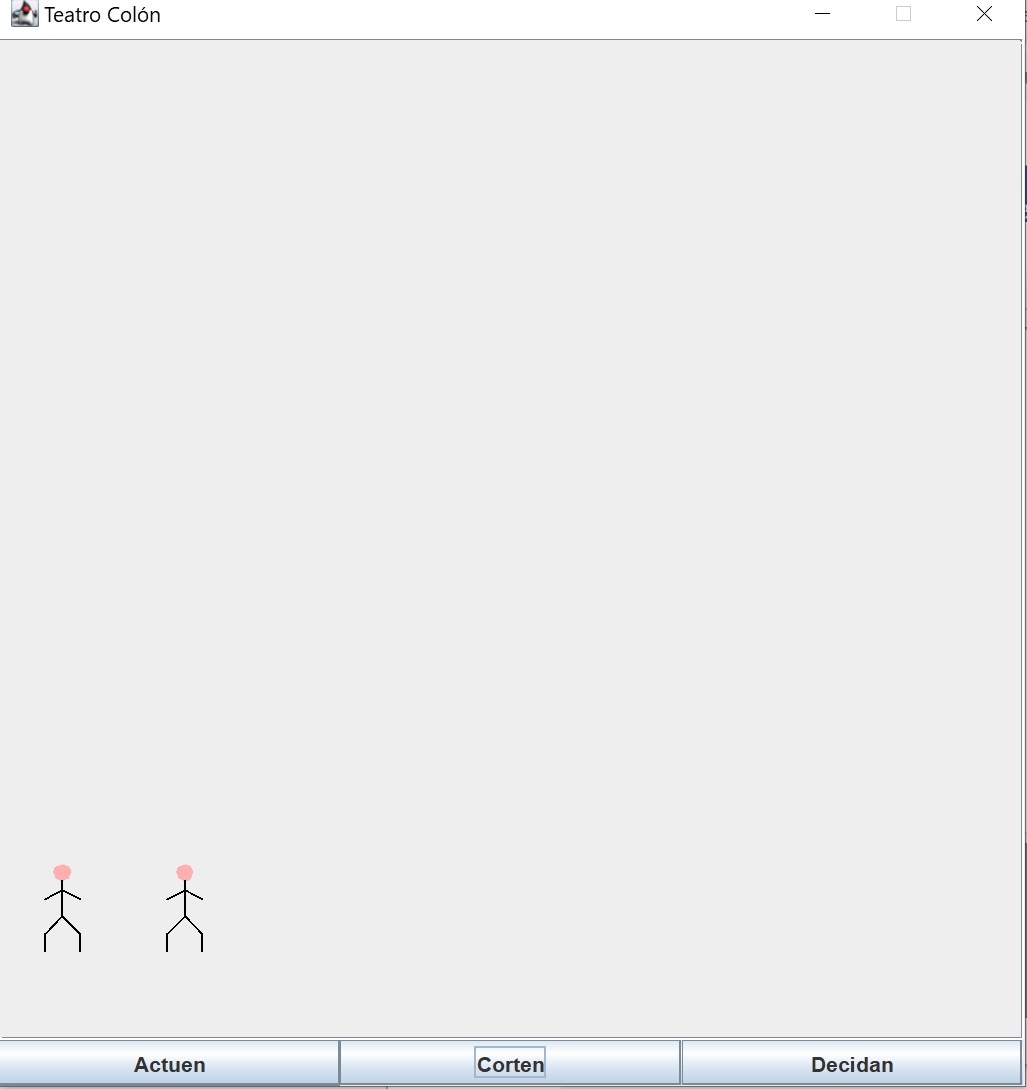
**4. En este punto vamos a construir (diseño y código) el método que atiende el click del botón Accion de la interfaz: el método llamado accion() de la clase Teatro. Ejecute el programa y haga tres click en el botón Accion. ¿Cómo actúan romeo y julieta? Capture la pantalla inicial y la final.** Se mueven y cambian sus palabras por “soy ...” y en los puntos suspensivos va el espacio de memoria asignado.



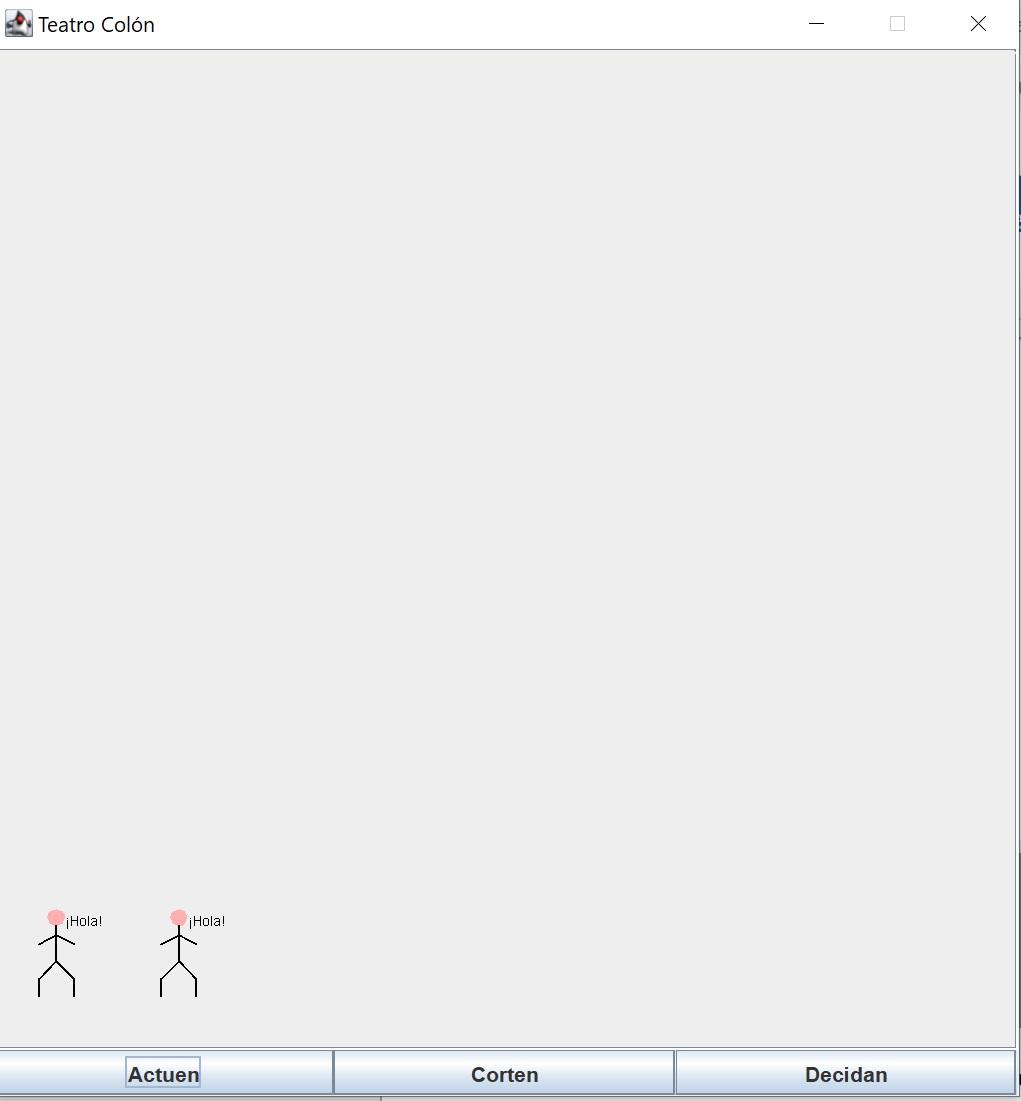


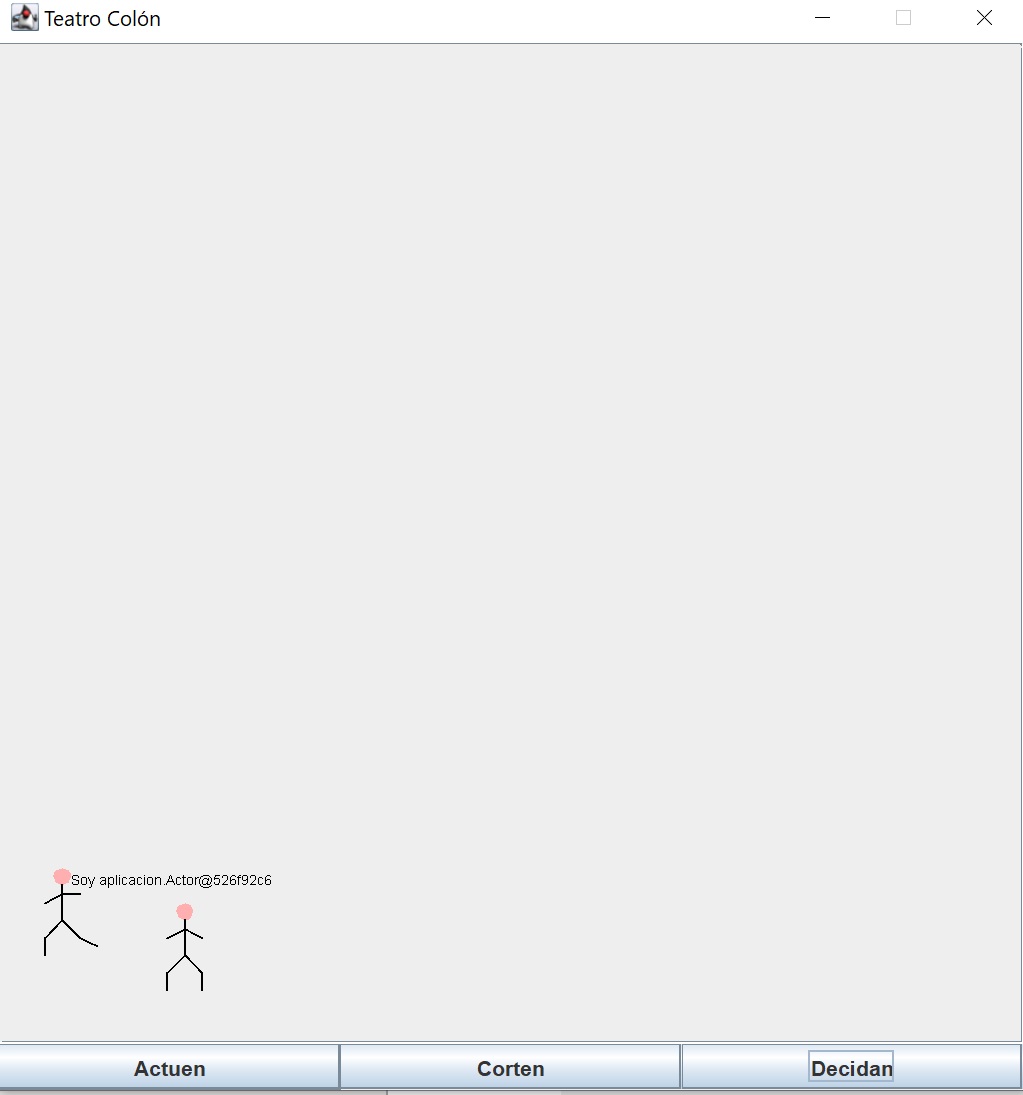
**5. En este punto vamos a construir (diseño y código) el método que atiende el click del botón corten de la interfaz: el método llamado corten() de la clase Teatro. Construya el método, ejecute el programa y haga click en el botón Corten. ¿Como quedan todos los actores después de esta orden? Capture la pantalla inicial y la final.** Los actores quedan en la posicion default de persona y sus palabras cambian a ser nada.





**6. En este punto vamos a construir (diseño y código) el método que atiende el click del botón decidan de la interfaz: el método llamado corten() de la clase Teatro. Construya el método, ejecute el programa y haga click en el botón Decidan. ¿Como quedan todos los actores después de esta orden? Capture la pantalla inicial y la final.** Cada actor toma una decisión binaria aleatoria entre actuar y cortar. En este caso Romeo actúa y Julieta corta.

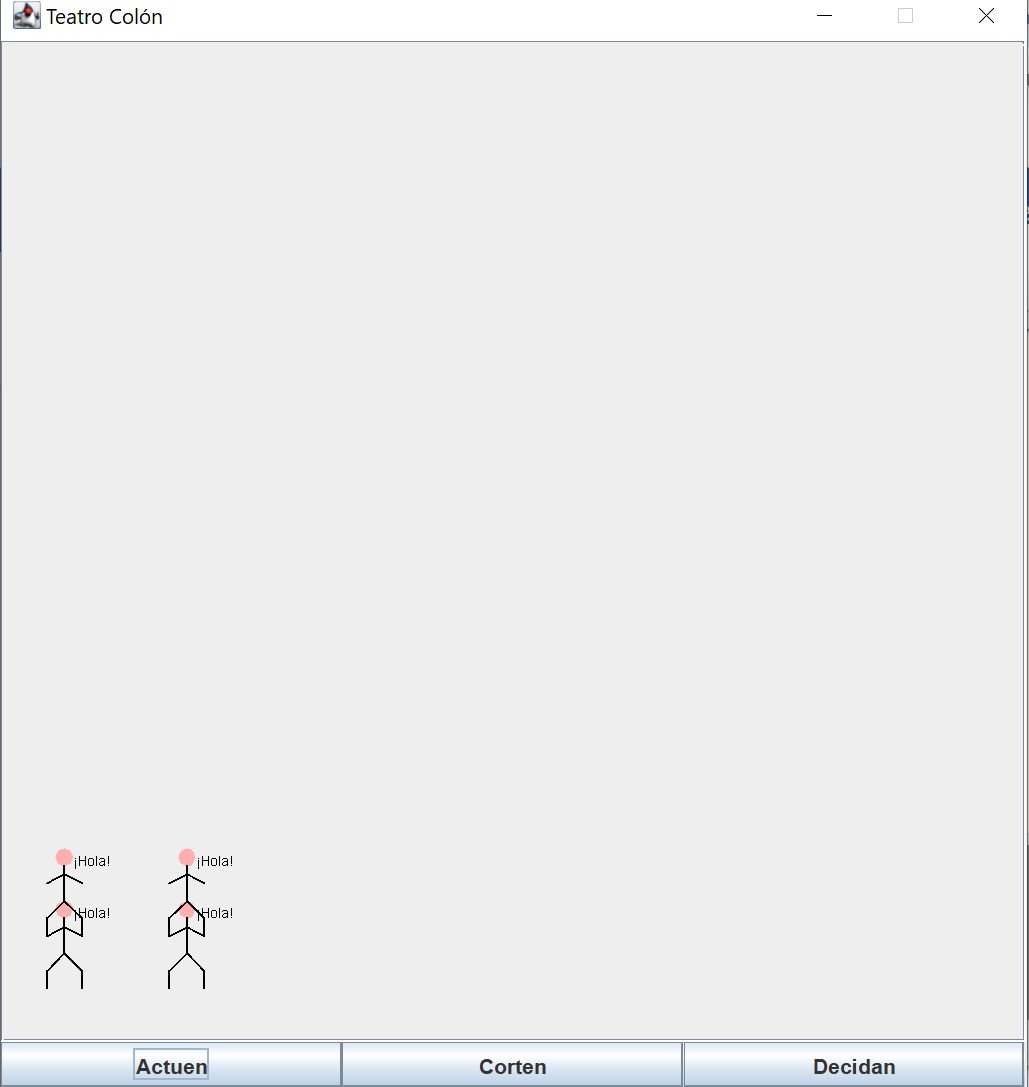


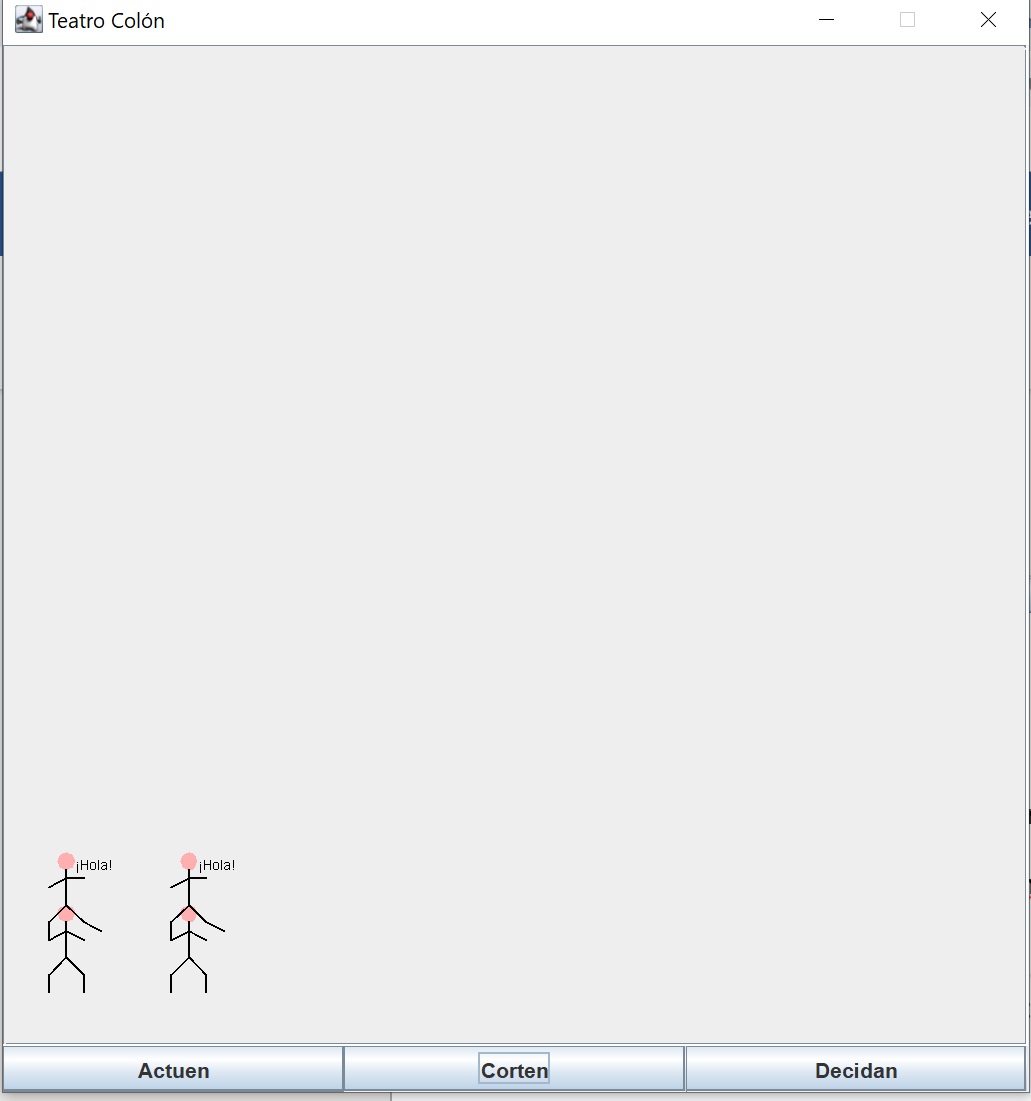


**Ciclo 2. Incluyendo a los actores necios: [En lab04.doc, TeatroColon.asta y \*.java] El objetivo de este punto es permitir recibir en el salón actores necios. (NO OLVIDE BDD - MDD)**

**1. Los actores necios normalmente llevan la contraria. Si se les pide acción cortan y si se les pide que corten entran en acción. Adicionalmente, cuando les piden decidir hacen lo que hicieron la última vez. Implemente este nuevo actor. ¿cuáles métodos se sobre-escriben (overriding)?** Se sobre-escriben los métodos actue, corte y decida.

**2. Adicione una pareja de actores necios, llámelos homer y bard, ejecute el programa y pídales a todos que actúen y que descansen. Capture la pantalla. ¿Qué pasa?**

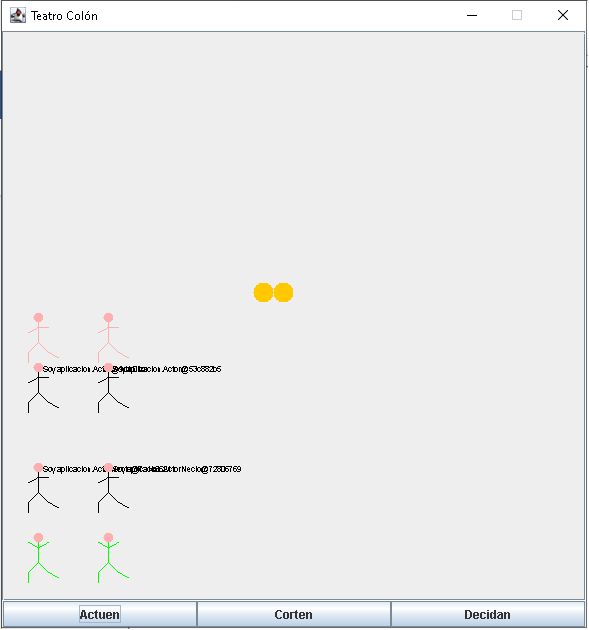


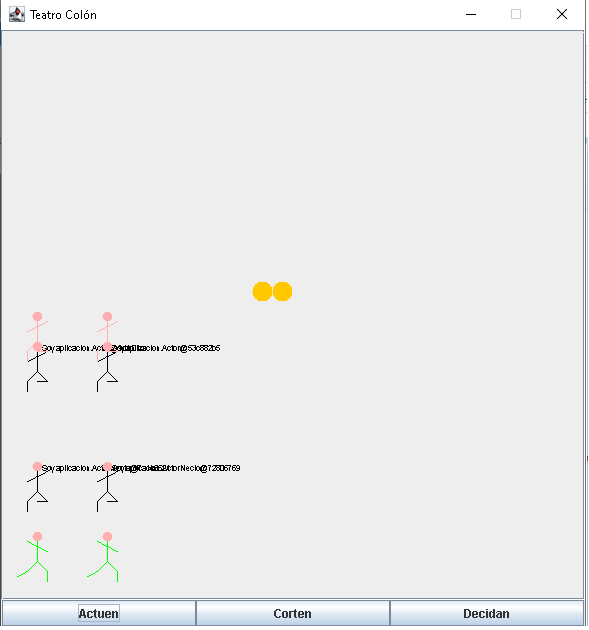


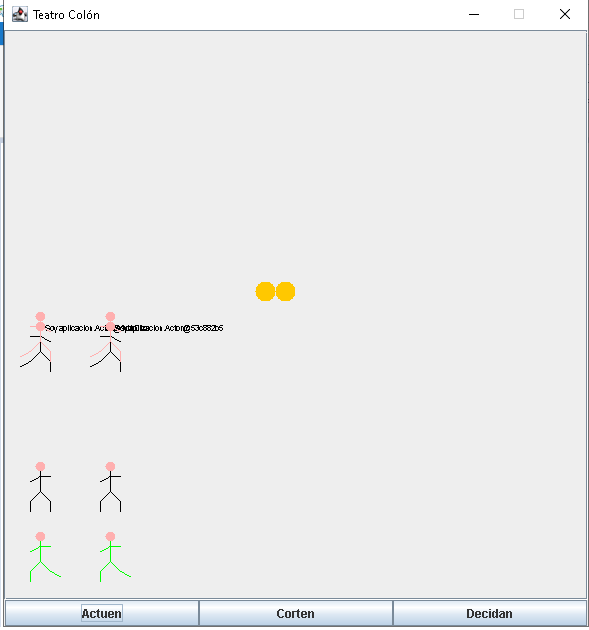
**3. Ahora, los necios quieren sorprender con su necedad; es decir, sólo son necios cada tres veces. ¿Qué modificaría para lograr este comportamiento? ¡Hágalo!** Creamos un contador, lo inicializamos en 1 en el creador de los actores necios y usamos el residuo (mood) de la división para que él actor sepa cuál es la tercera vez y actúe de forma necia. Ahora si nos piden actuar y cortar no se diferenciaría cual actor es necio y cual es normal.

**4. Nuevamente ejecute el programa y pídales a todos que entren en acción y corten. Capture la pantalla. ¿Qué pasa?**

Ahora actúan de una manera correcta cada 3 acciones, para las siguientes 3 imágenes se apretó el botón actúen y en la ultima los actores necios cortan





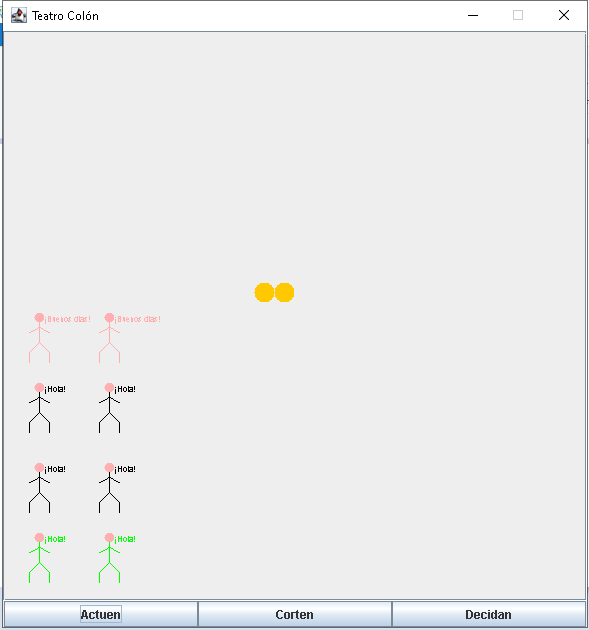


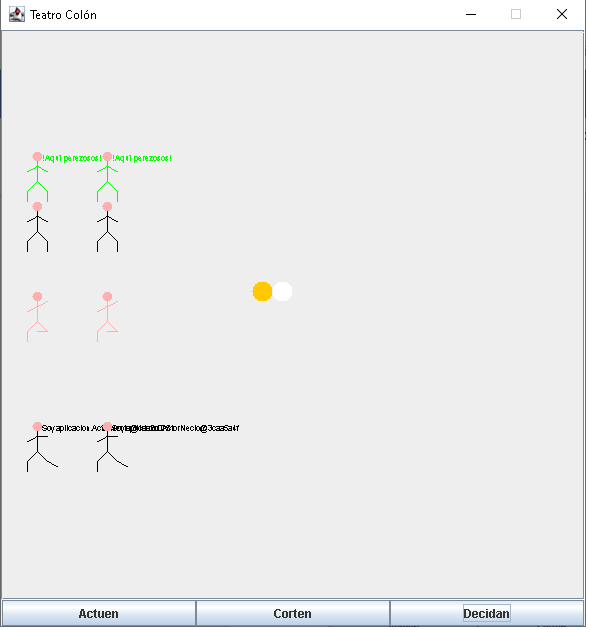
**Ciclo 3. Adicionando luces: [En lab03.doc, TeatroColon.asta y \*.java] El objetivo de este punto es incluir en el Teatro luces (sólo vamos a permitir este tipo de luces). Las luces se prenden cuando hay acción y se apagan cuando hay corte. (NO OLVIDE BDD – MDD)**

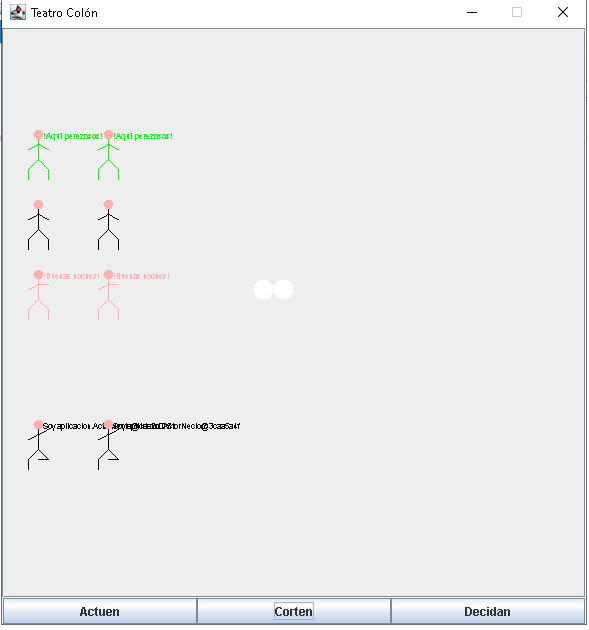
**5. Construya la clase Luz para poder adicionarla en el Teatro. ¿qué cambios incluyó?** Definimos la clase luz y le hicimos implementar la interfaz enEsena, en este punto los cambios se hicieron en la clase luz, no hemos tocado la clase teatro.

**6. Para aceptar este elemento, ¿debe cambiar en el código del Teatro. en algo? ¿por qué?**  Para hacer que el elemento aparezca en el teatro, tenemos que crearlo y adicionarlo al array de elementos tipo enEsena, es el único cambio realizado en teatro hasta ahora

**7. Adicionen dos luces en el centro del Teatro, llámenlas centralDerecha y centralIzquierda, ejecuten el programa. ¿Qué pasa? ¿es correcto?**







Las luces aparecen en el teatro, son color naranja y en las 3 imágenes se ve el resultado de apretar los botones, en actúen las luces se encienden o permanecen encendidas, en el corten las luces se apagan o permanecen apagadas, se simboliza con cambiar su color a blanco, y en el decidan cada luz toma una acción.

**Ciclo 4. Creando un nuevo actor: el perezoso [En lab03.doc, TeatroColon.asta y \*.java] El objetivo de este punto es incluir el actor perezoso, considerando que: (NO OLVIDE BDD - MDD)**

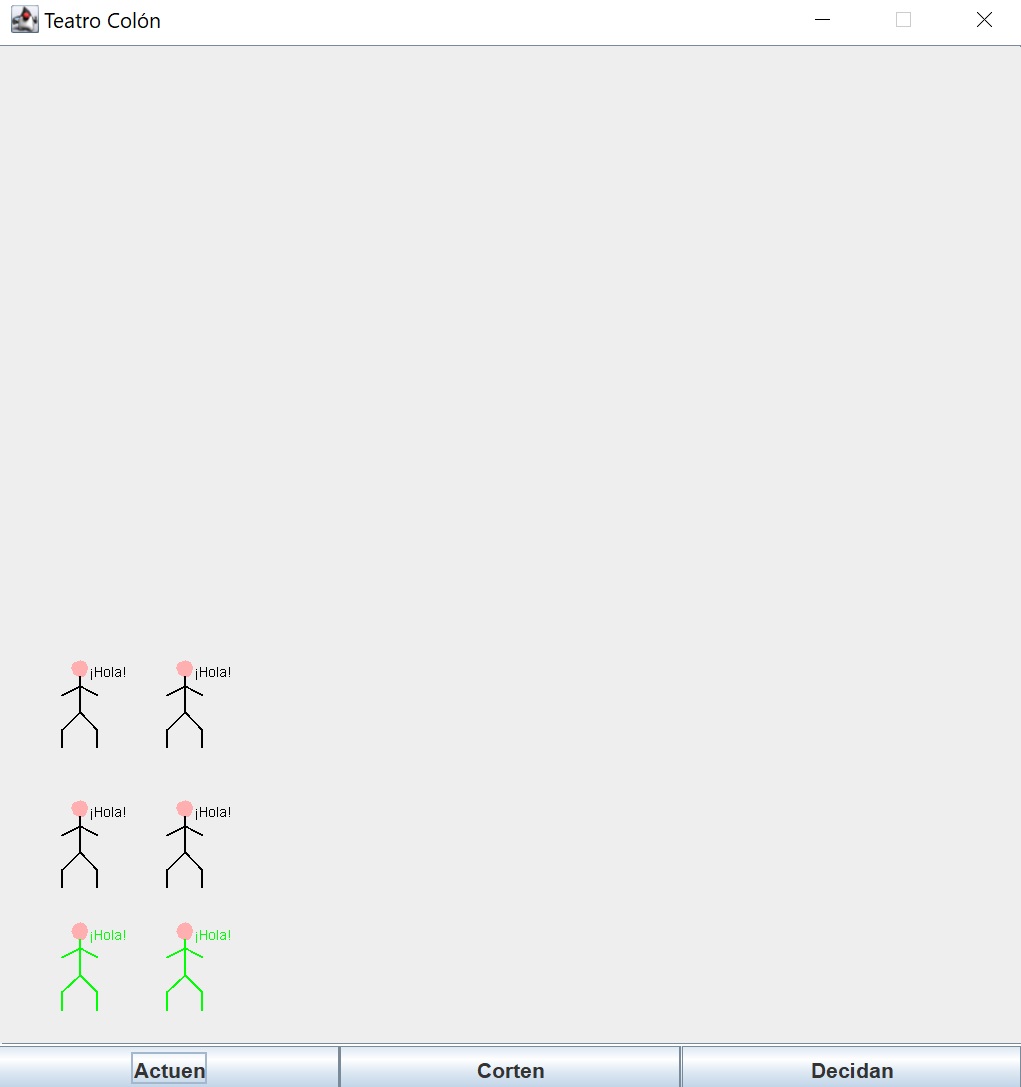
* **Está vestido con color verde**
* **En lugar de avanzar, como romeo y julieta, sólo mueve brazos y piernas (primero los sube y luego los baja)**
* **Descansa sentándose**
* **Siempre decide descansar.**
* **Es un actor muy silencioso cuando actúa. Cuando descansa llama a sus compañeros: “!Aquí perezosos!”**

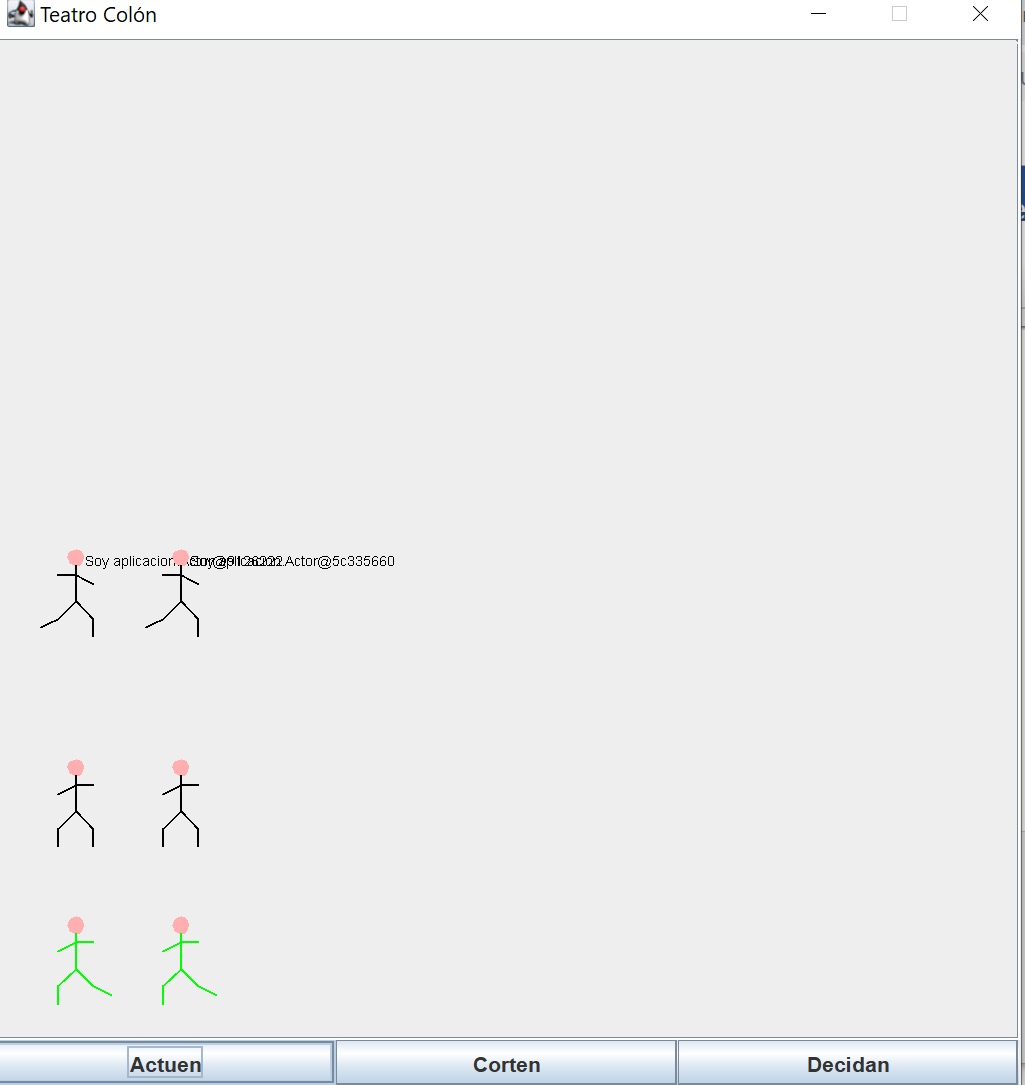
**1. Para implementar este actor, ¿qué cambios debería hacer al diagrama de clases? ¿a los de secuencia? Explique.**

Para este actor, hay que añadir su clase al diagrama de clases, es una clase que hereda de Persona e implementa enEsena, por lo que seria necesario hacer diagramas de secuencia especificos para los metodos de la clase que lo necesiten

**2. Implemente al actor Perezoso**

**3. Ahora si, adicione ahora una pareja de actores perezosos llámelos bella y edward, ejecute el programa y haga tres click en el botón. ¿Cómo quedan todos los actores? Capture la pantalla inicial y la final.** Los actores necios cortan al tercer actuar, los normales siguen actuando de igual forma y los perezosos se mueven y actúan diferente a los demás.





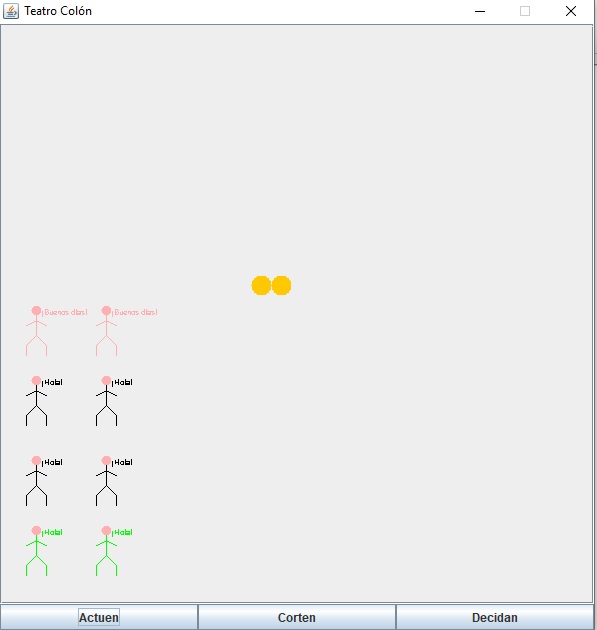
**Ciclo 5. Nuevo actor: Proponiendo y diseñando: El objetivo de este punto es permitir recibir en un nuevo tipo de actor. (NO OLVIDE BDD - MDD)**

**1. Propongan, describan e implementen un nuevo tipo de actor.**

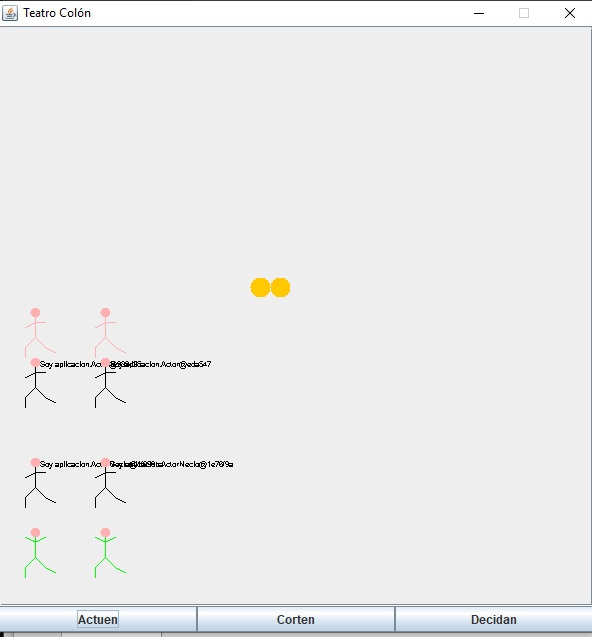
**2. Incluyan una pareja de ellos con el nombre de ustedes. Ejecute el programa con dos casos significativos. Explique la intención de cada caso y capture las pantallas correspondientes.**

Actor Ejemplar: Siempre decide descansar haciendo yoga porque su jornada es larga y agotadora, dice “mmmmm”. Es un excelente actor y por eso su vestuario es de color rosado, saluda con los buenos días al actuar, y se despide con buenas noches al cortar. Descansa sentado y levantando ambos brazos.

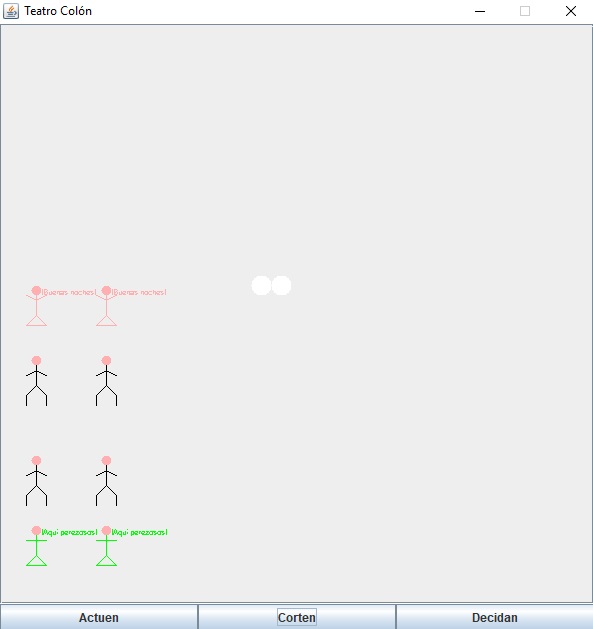
Inial:



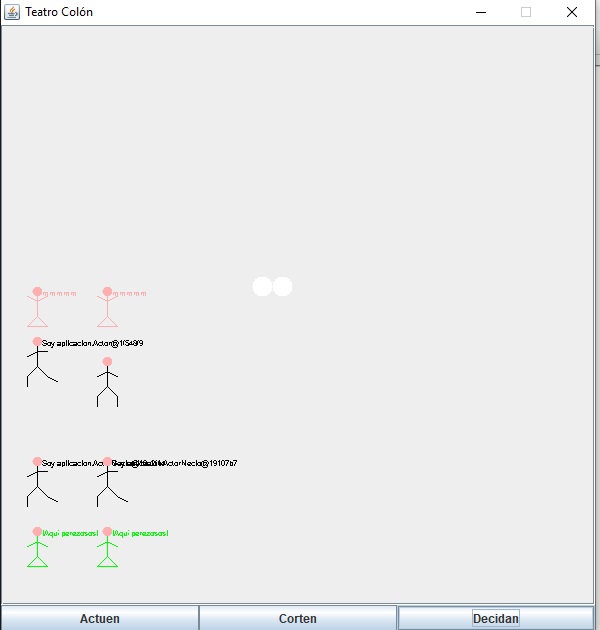
Actuar:



Cortar:



Decidan:

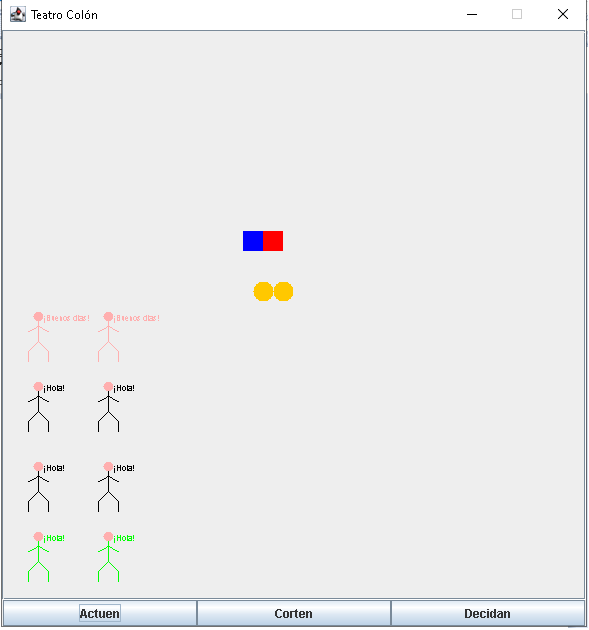


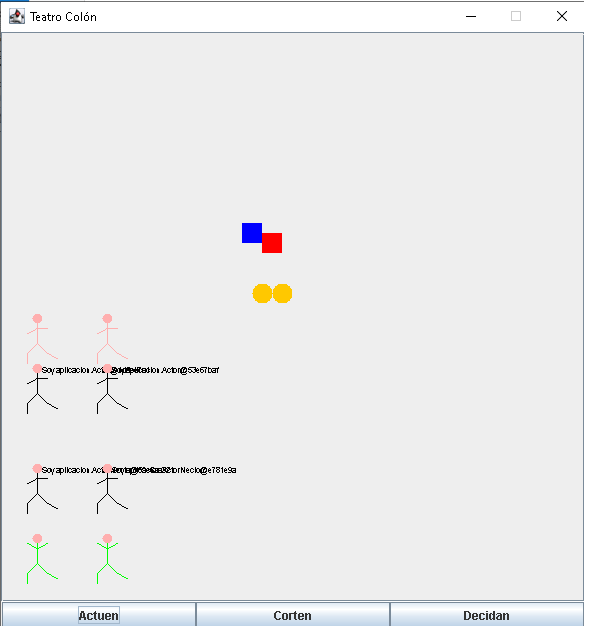
**Ciclo 6. Nuevo elemento: Proponiendo y diseñando. El objetivo de este punto es permitir recibir en un nuevo elemento en el escenario (NO OLVIDE BDD - MDD)**

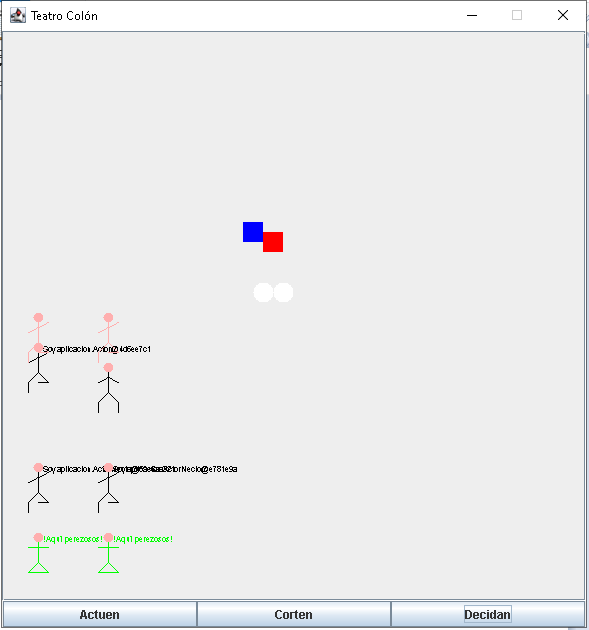
**1. Propongan, describan e implementen un nuevo tipo de elemento**

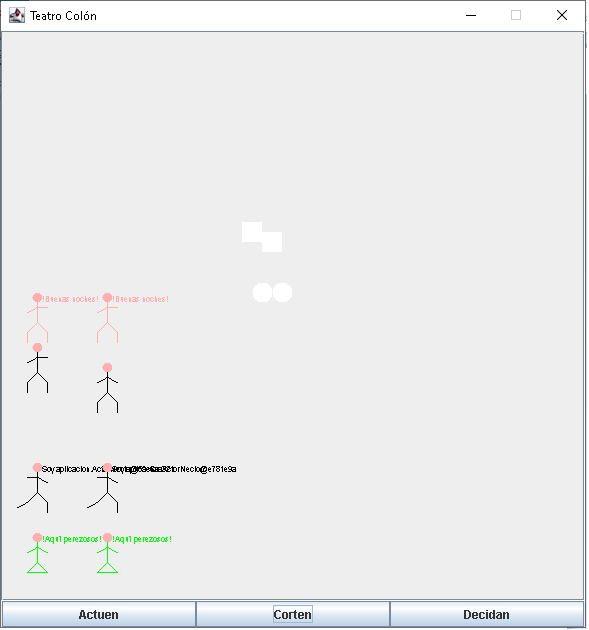
El nuevo elemento es la escenografía, que existe siempre que los actores estén actuando y desaparece cuando se hace corte, se mantiene presente incluso si algunos actores deciden cortar, por lo que solo desaparece cuando explícitamente se le dice corte, además algunas se mueven con los actores.

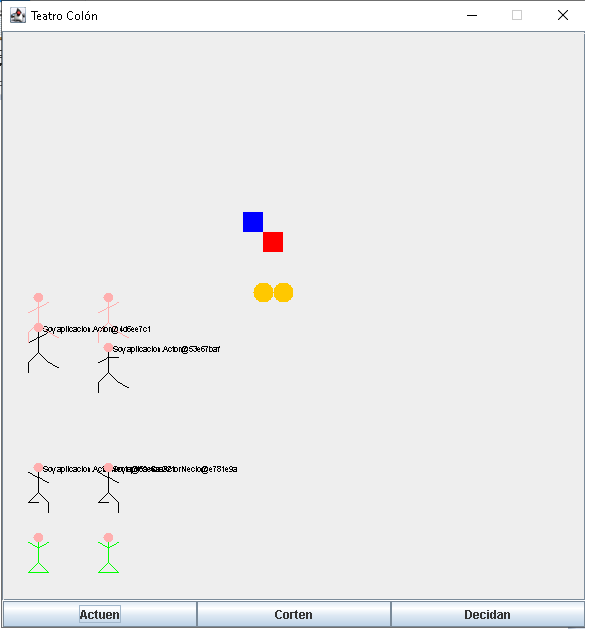
**2. Incluyan una pareja de ellos con los nombres semánticos. Ejecute el programa con dos casos significativos. Explique la intención de cada caso y capture las pantallas correspondientes.**









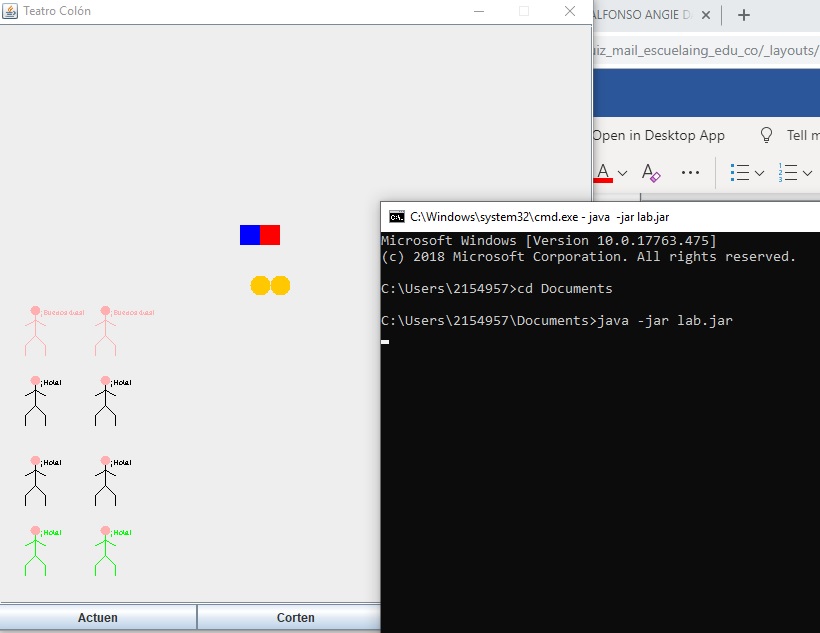


En las 5 imágenes anteriores, se ven gráficamente cómo se comporta la escenografía, se diferencia la móvil y la inmóvil con los colores azul y rojo respectivamente (imagen\_1), la primera vez que se aprete el botón actúen, la móvil se mueve y la inmóvil no (imagen\_2), cuando se aprete el decidan no hacen nada (imagen\_3), cuando se aprete el corten, cambian de color a blanco “sacándoles de escena” (imagen\_4), y una vez más se aprete el botón actúen (imagen\_5).

**Empaquetando la versión final para el usuario: [En lab03.doc, TeatroColon.asta , \*.java, TeatroColon.jar]**

**1. Revise las opciones de BlueJ para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.**

**2. Consulte el comando java para ejecutar un archivo jar. Ejecutelo ¿qué pasa?**



**3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.** Las ventajas es que se puede entregar de forma comprimida, en un solo archivo y este fácilmente es ejecutable sin que el cliente sea capaz de ver el código, además de que este es fácilmente portable a otras distribuciones de software debido a que solamente es necesario que este instalada la JVM (Java Virtual Machine)

**DE BLUEJ A CONSOLA: En esta sección del laboratorio vamos a aprender a usar java desde consola. Para esto se va a trabajar con el proyecto del punto anterior.**

**Comandos básicos del sistema operativo** [En lab03.doc] Antes de iniciar debemos repasar los comandos básicos del manejo de la consola.

**1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.**

DIR: Permite ver el contenido de un directorio

MD o MKDIR: Permite crear directorios

COPY: crear archivos

TREE: ver el árbol de directorios

XCOPY: Copiar directorios

REN o RENAME: Cambiar nombre de directorio y archivos

MOVE: mover archivos y directorios

DEL: Eliminar archivos

RD: Eliminar directorios

**2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio:**

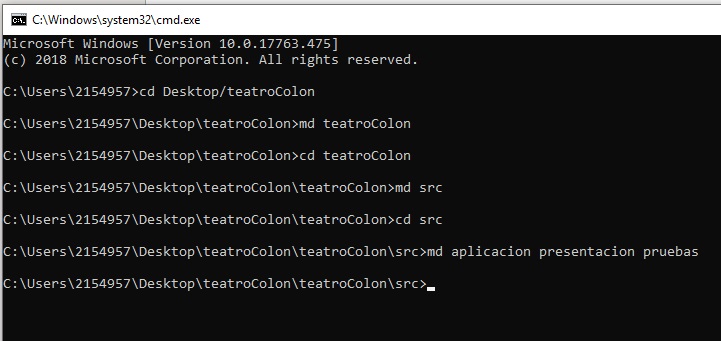
**teatroColon**

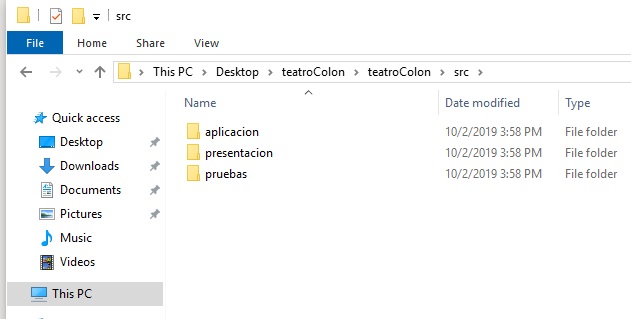
**src**

**aplicacion**

**presentacion**

**pruebas**





**3. En el directorio copien únicamente los archivos \*.java del paquete de aplicación . Consulte y capture el contenido de: src/aplicacion**

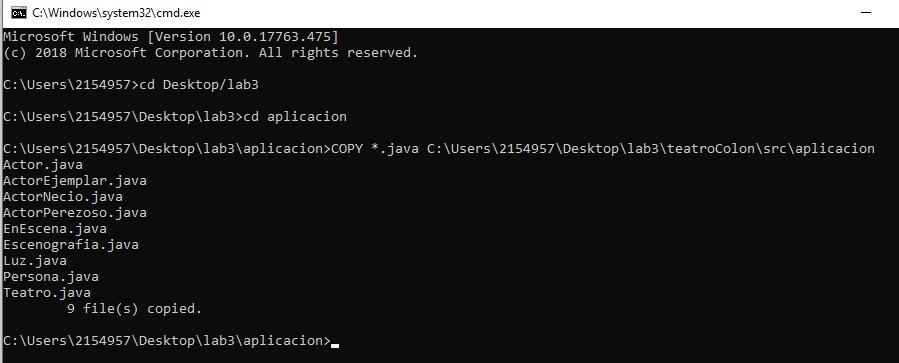
**Estructura de proyectos java: [En lab03.doc] En java los proyectos se estructuran considerando tres directorios básicos:**

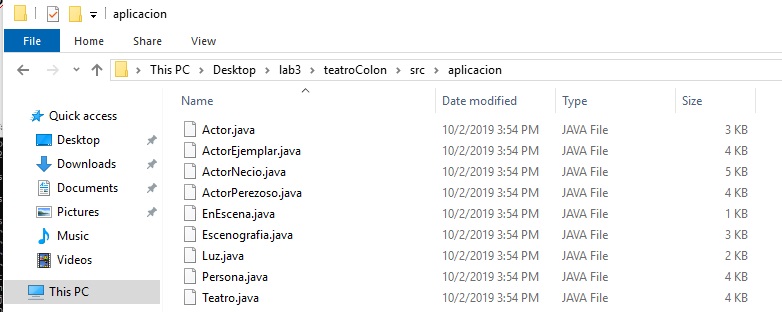
**automata**

**src**

**bin**

**docs**





**1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.**

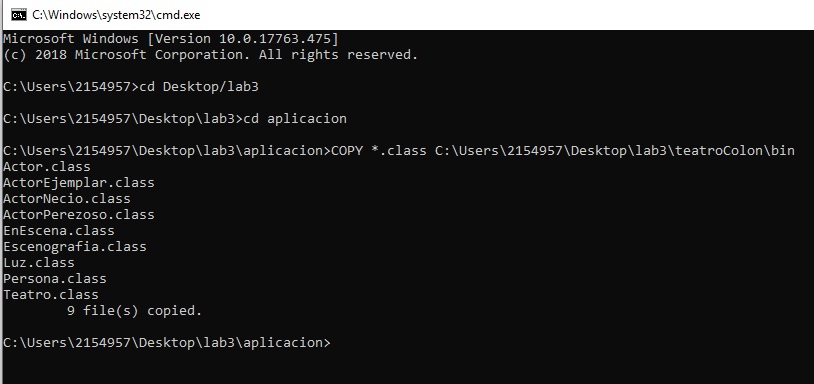
En src deben estar los códigos fuentes .java

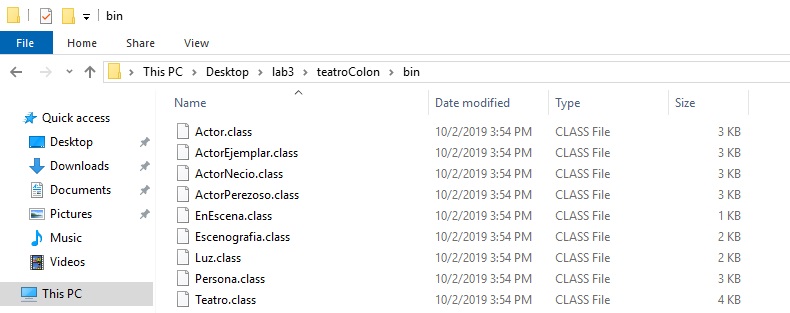
En bin deben estar los bytecodes de los códigos fuentes.

En docs. se encuentra la documentación del proyecto

**2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.**

En bin deben estar los bytecodes de los códigos fuentes, porque los archivos .class contienen el bytecode del proyecto.





**Comandos de java: [En lab03.doc]**

**1. Consulte para qué sirven cada uno de los siguientes comandos:**

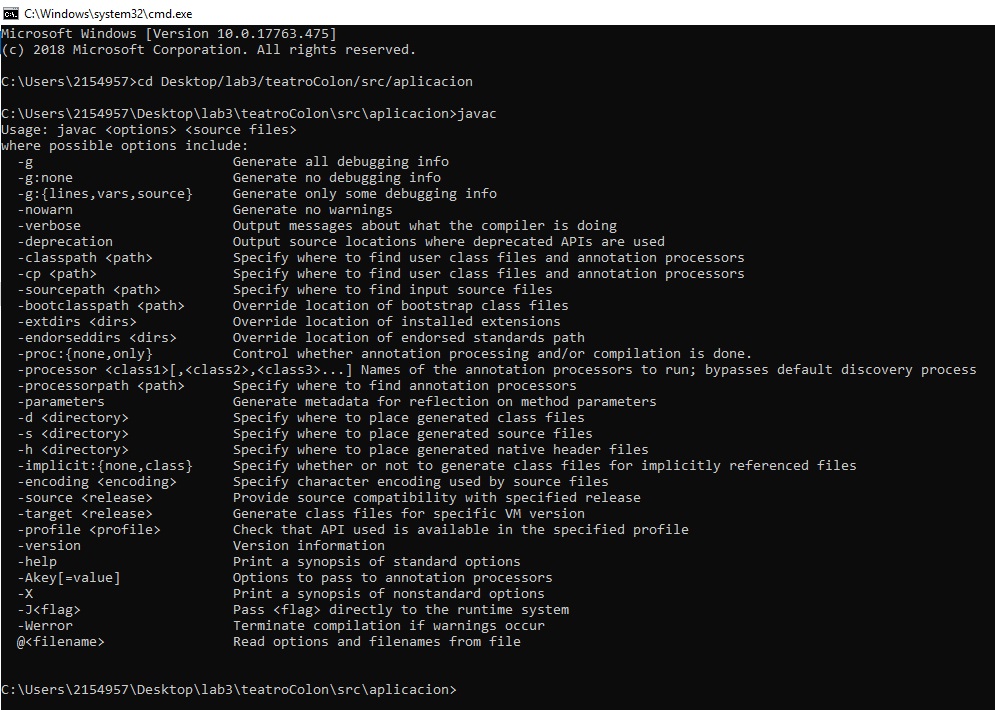
**Javac:** Compila los archivos con código fuente (.java) en byte-codes Java (.class)

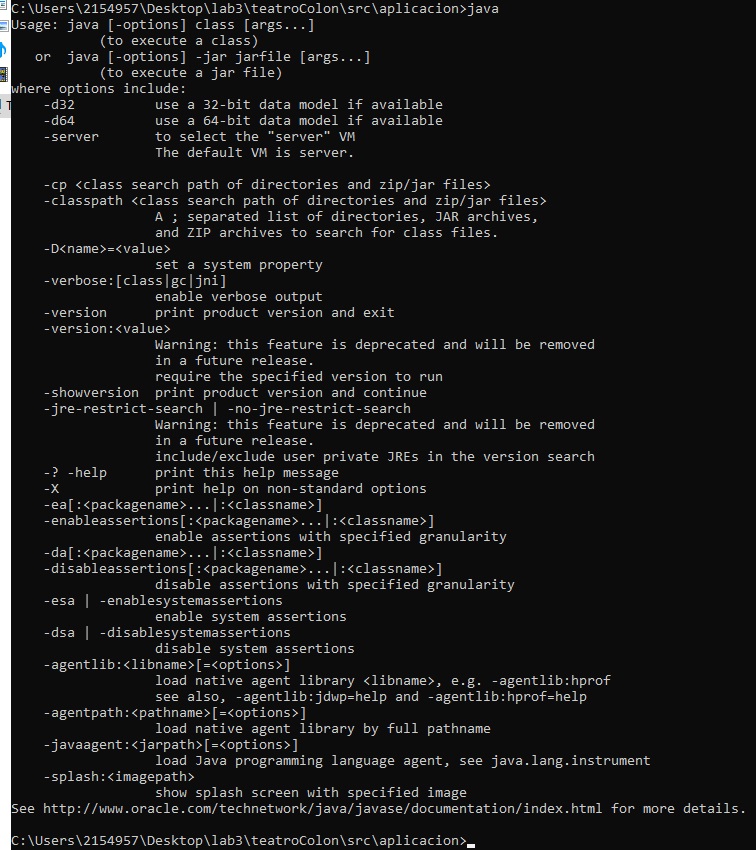
**Java:** Es el intérprete de Java; es un componente de la JVM capaz de interpretar los archivos (.class) en la computadora

**Javadoc:** Genera páginas de HTML de API de la documentación de los códigos fuente .java

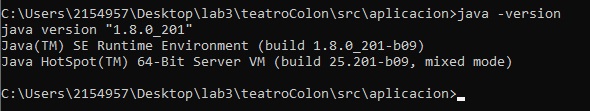
**Jar:** Comprime proyectos java en un solo archivo y también ejecuta estos archivos.

**2. Cree una sesión de consola y consulte en línea las opciones de los comandos java y javac. Capture las pantallas.**





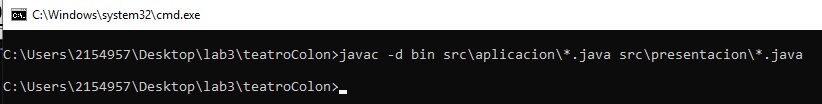
**3. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.**



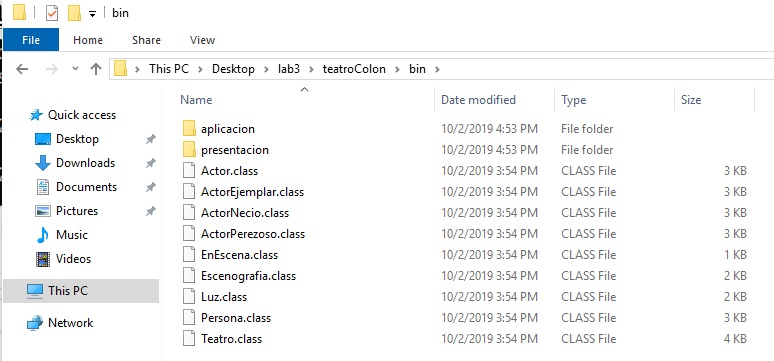
**Compilando: [En lab03.doc]**

**1. Utilizando el comando javac, desde el directorio raiz (desde automata con una sóla instrucción), compile el proyecto. ¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto? Tenga presente que se pide un único comando y que los archivos compilados deben quedar en los directorios respectivos.** Para compilar **todo** usamos:

javac -d bin src\aplicacion\\*.java src\presentacion\\*.java



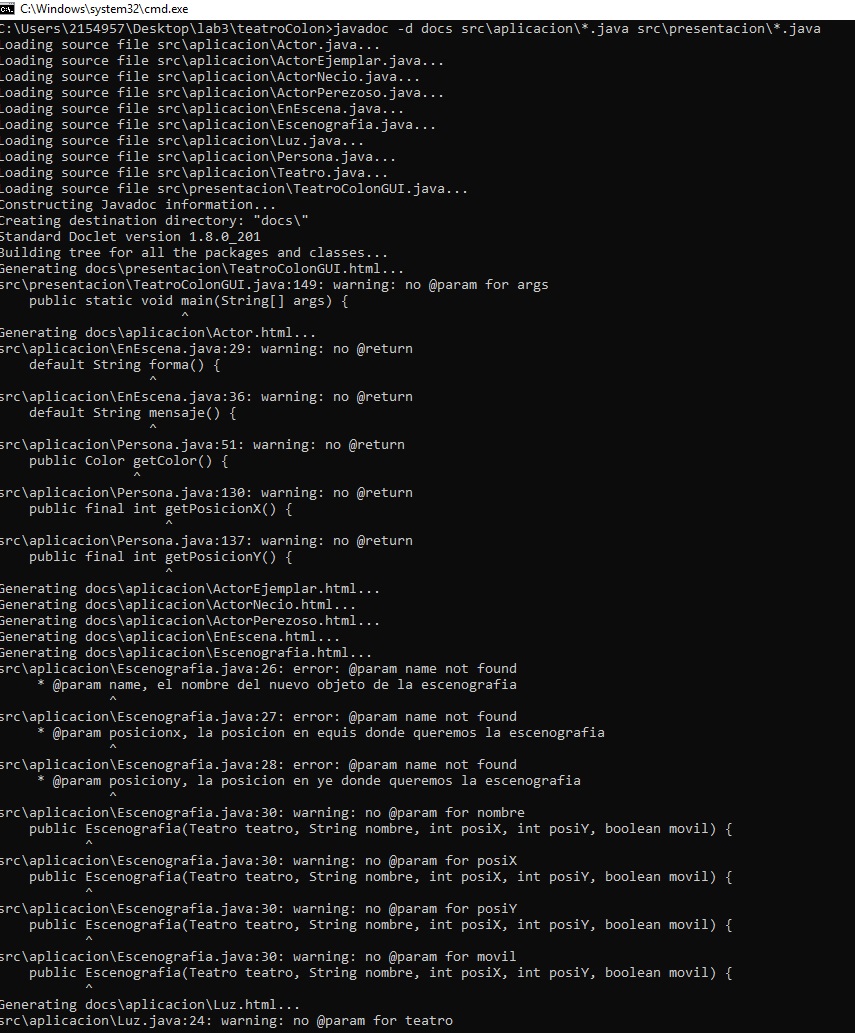
**2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?** Aparecieron nuevos archivos .class en los directorios bin de la aplicación y la presentación.



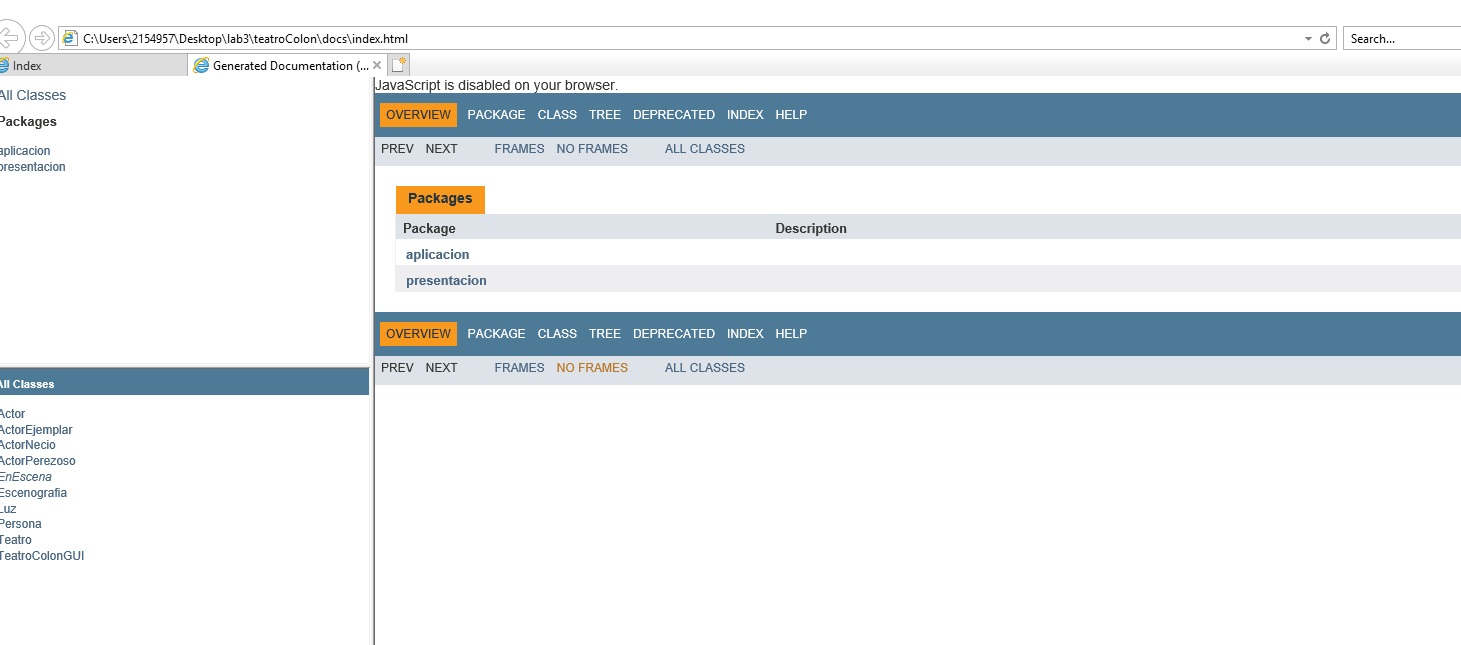
**Documentando: [En lab03.doc]**

**1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?** Usamos:

javadoc -d docs src\aplicacion\\*.java src\presentacion\\*.java

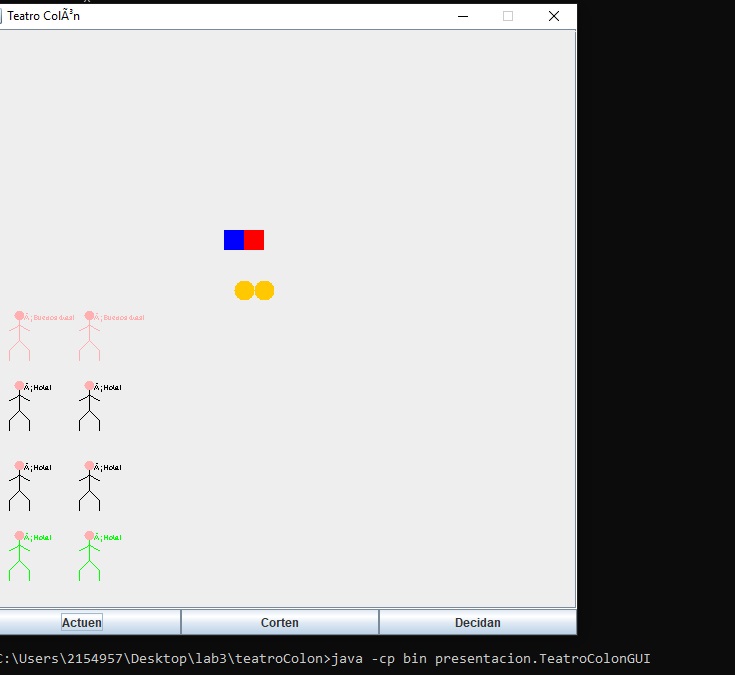


**2. ¿Cuál archivo hay qué abrir para empezar a navegar por la documentación? Ábralo y capture la pantalla.** El archivo index.html



**Ejecutando: [En lab03.doc]**

**1. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo utilizó este comando?**  Usamos: java -cp bin presentacion.TeatroColonGUI



**Probando: [En lab03.doc]**

**1. Adicione ahora los archivos del directorio pruebas y trate de compilar nuevamente el programa. Tenga en cuenta que estas clases requieren la librería junit 4.8. ¿Cómo se incluye un paquete para compilar? ¿Qué instrucción completa tuvo que dar a la consola para compilar?** No teniamos pruebas pero si hubieran pruebas seria:

javac -d bin -cp junit-4.12.jar src\aplicacion\\*java src\pruebas\\*java

Pero como no teniamos:

javac -d bin src\aplicacion\\*.java src\presentacion\\*.java

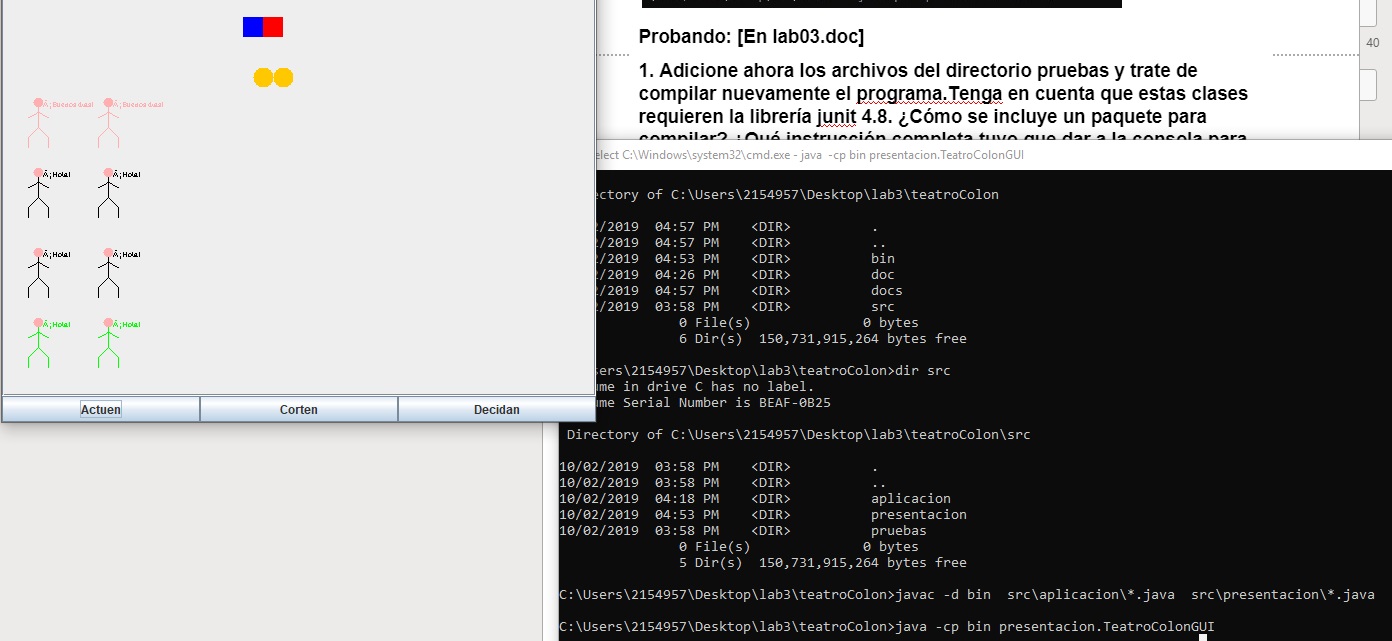


**2. Ejecute desde consola las pruebas . ¿Cómo utilizó este comando?. Puede ver ejemplos de cómo ejecutar el“test runner”en: http://junit.sourceforge.net/doc/cookbook/cookbook.htm**

Pero como no teniamos usamos:

java -cp bin presentacion.TeatroColonGUI

Y se ejecuta:

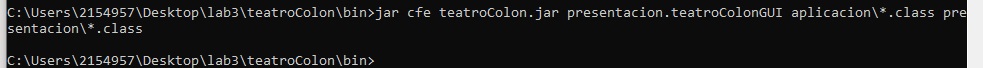


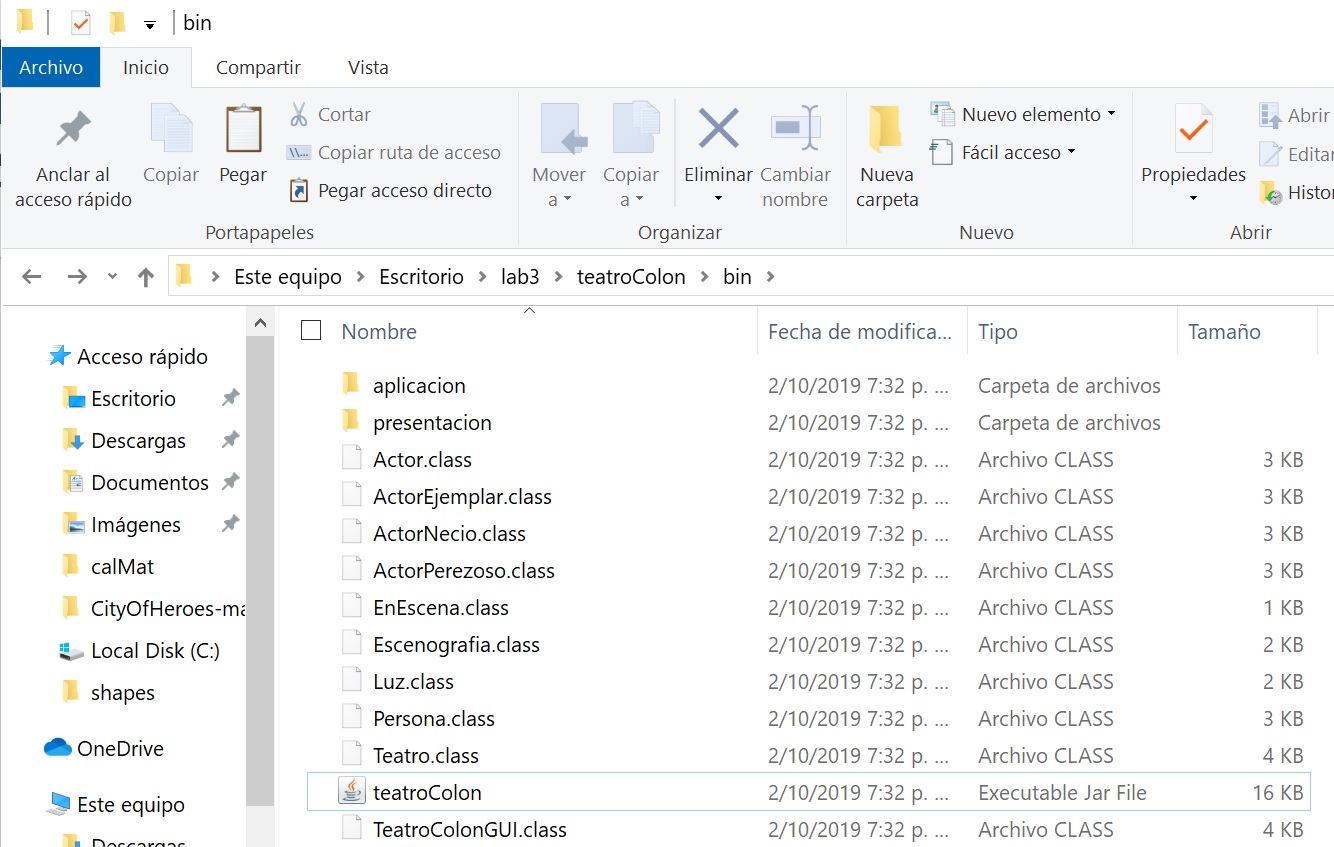
**3. Pegue en su documento el resultado de las pruebas**

**Empaquetanedo: [En lab03.doc]**

**1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual de java y su entorno de ejecución (JRE). ¿Cómo empaquetó jar?**

jar cfe teatroColon.jar presentacion.teatroColonGUI aplicacion\\*.class presentacion\\*.class





**2. ¿Cómo se ejecuta el proyecto empaquetado?**

C:\Users\2154957\Desktop\lab3\teatroColon\bin>java -jar teatroColon.jar

No pusimos screen de la pantalla porque en la universidad creamos el .jar y ya al momento de llegar a nuestras casas vimos que el jar se quedó en el computador de la escuela y desde casa no me reconoció los mismos comandos que use en la u. Pero pusimos el fragmento de código que usamos en la universidad para abrirlo. Y dejamos los directorios igualmente como el screen anterior.

**RETROSPECTIVA:**

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

10 horas Ernesto Camacho

14 horas Daniela Ruiz

**2. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)**

El laboratorio esta completo

**3. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importante?**

Dado que el proyecto se desarrolló en los ciclos definidos, y teniendo en cuenta que la practica xp de esta semana es que los proyectos se desarrollan en iteraciones, podemos decir con confianza que hacer uso de esta práctica nos facilitó bastante el desarrollo de aparte de blueJ lo cual nos facilitó trabajar en consola.

**4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?**

En el ciclo de probar en consola lograr que funcionara, dado que no entendíamos muy bien la instrucción, la forma en la que solucionamos el problema fue solicitando ayuda, así como una búsqueda por parte de cada quien para ver si encontrábamos algo.

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

Nos corregimos y apoyamos al momento de desarrollar el código.

* Ernesto, en lo personal creo que sigo siendo algo individualista en el desarrollo del código, lo que me llevo a tener roces tensos con Daniela, me comprometo a mejorar mi actitud hacia ella y a tener en cuenta sus ideas para el desarrollo de futuros proyectos.