**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Herencia e interfaces**

**ADEMAS Java desde consola**

**2019-02**

**Laboratorio 3/6**

**Eduard Arias – Juan Diaz**

**Conociendo**

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

a) ¿Cuántos paquetes tiene?

Tiene dos paquetes, apicación y presentación

b) ¿Cuántas clases tiene en total? ¿Cuántas tienen fuentes?

Tiene 6 clases en total.

c) ¿Cuál es la clase ejecutiva? ¿Por qué?

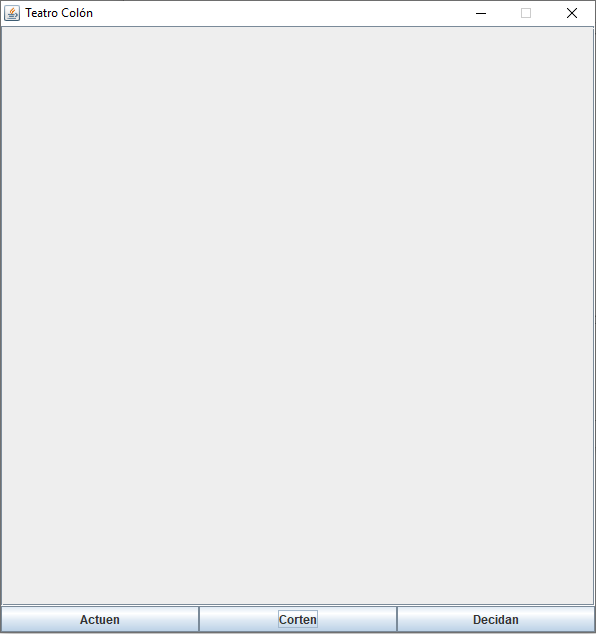
La clase ejecutiva es TeatroColonGUI, ya que esta en la capa de presentación y se encarga d hacer gráfica la parte lógica.

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

qué?

Abre una Ventana en gris vacía con tres botones en la parte inferior. Niguno realiza alguna acción.

Esto se debe a que los métodos que usan los botones no estan programados.



**Arquitectura general.**

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

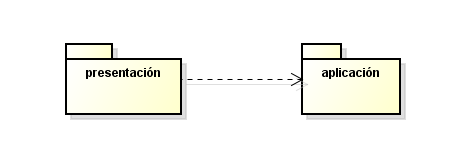
- Package: Es paquete. Hace referencia a un directorio que contiene clases que se relacionan entre si.

- Import: Es importar. Se usa cuando se requiere importar una clase de un paquete específico. Se pueden usar las clases de Java aquí.

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

Tiene dos carpetas y el ejecutable de BlueJ. En las carpetas se encuentran los archivos de las clases de java y sus archivos de compilación. Los paquetes son los mismos directorios, presentación y aplicación.

**3.** Inicie el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.



**Arquitectura detallada.**

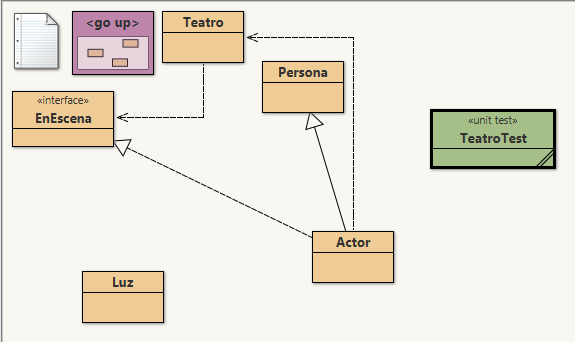
**1.** Usando ingeniería reversa prepararen el proyecto para MDD. Presente el diseño estructural actual de la aplicación (diagrama de clases). Las clases de la capa de presentación sólo deben tener los elementos públicos.

Diagramas en el archiveo .astah

**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?

Se crea la clase en el paquete de aplicación porque ahí esta la lógica y serivicios del programa.

Esta asociado a las clase Teatro porque es la clase principal de aplicacion ya que prove los servicios.



**Ciclo 1. Actuan y descansan los actores normales**

**1.** Estudie la clase Teatro. ¿Qué tipo de colección se usa para albergar los elementos?

¿Puede recibir actores? ¿Por qué?

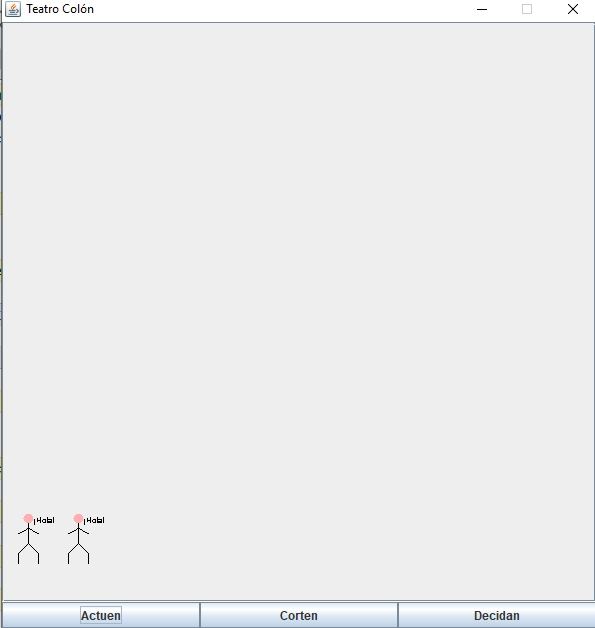
Utiliza un ArrayList de interfaces EnEscena. Sí puede recibir actores porque estos implementan la interfaz.

**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?

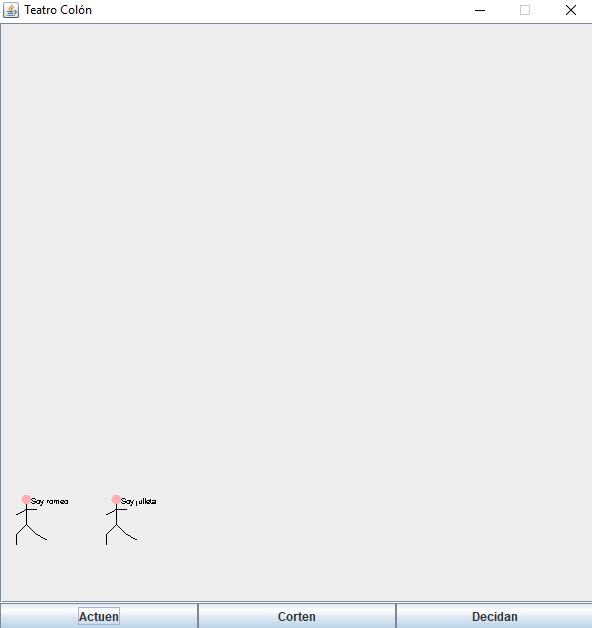
Es de color negro; dice “Hola!”; entran en acción con el método actue(), al cortar se mueven a una posición inicial y no dicen nada, y para decidir primero miran la posicion de sus extremidades y verifica si las puede mover.

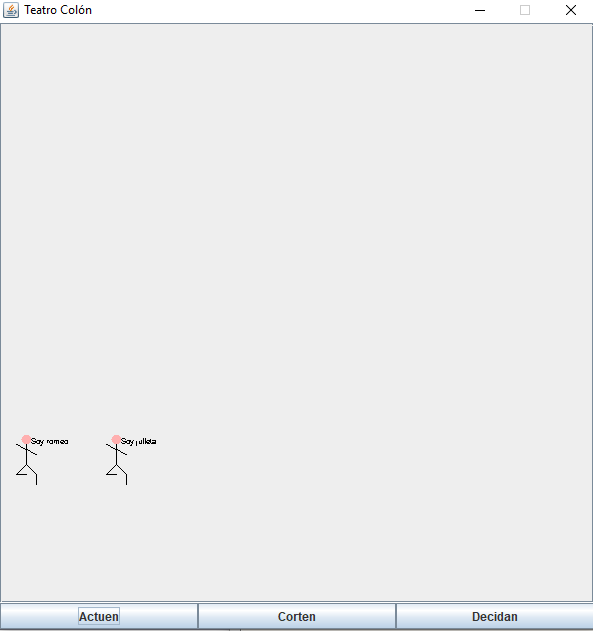
**3.** En el método algunosEnEscena de la clase Teatro cree dos actores en diferentes posiciones y acondiciónelos al Teatro llámelos romeo y julieta. Ejecute el programa y capture la pantalla. ¿Qué pasa ahora? ¿Pidales que entren en acción? ¿Qué pasa? ¿Por qué?

Los actores se pueden visualizar dentro del teatro. Al decirles que entren en acción no pasa nada porque el método accion() no esta implementado.

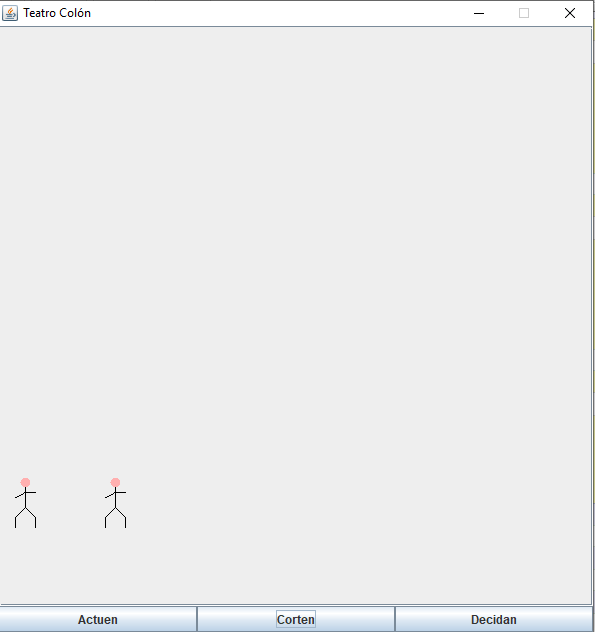
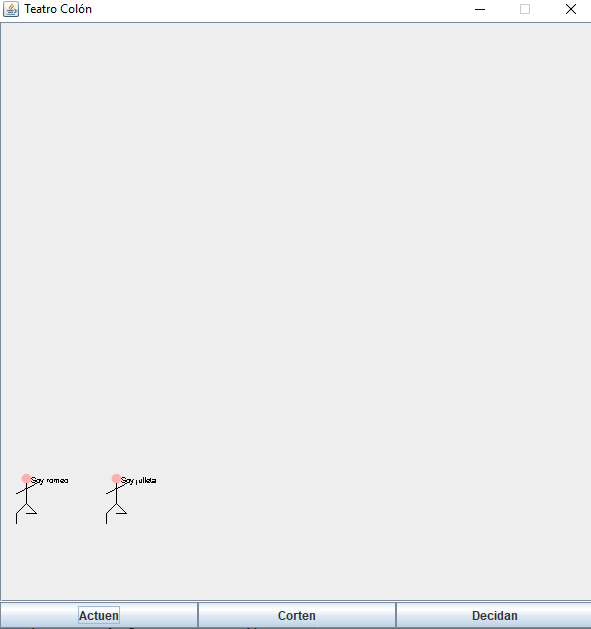


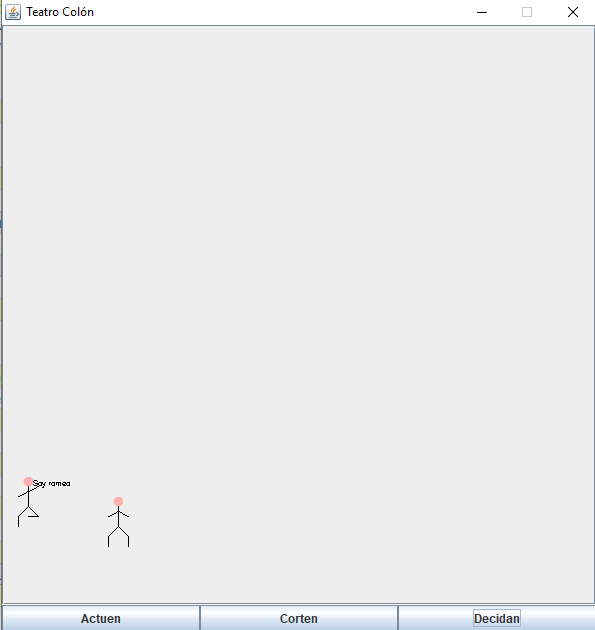
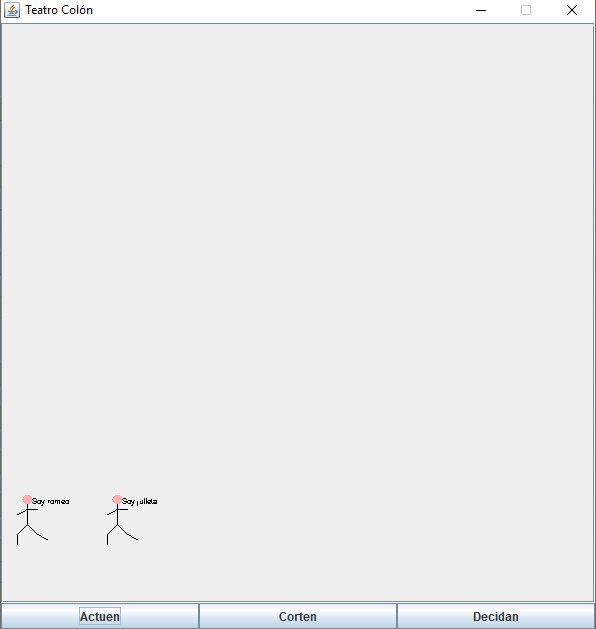
**4.** En este punto vamos a construir (diseño y código) el método que atiende el click del botón Accion de la interfaz: el método llamado accion() de la clase Teatro. Ejecute el programa y haga tres click en el botón Accion. ¿Cómo actúan romeo y julieta? Capture la pantalla inicial y la final.





**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.



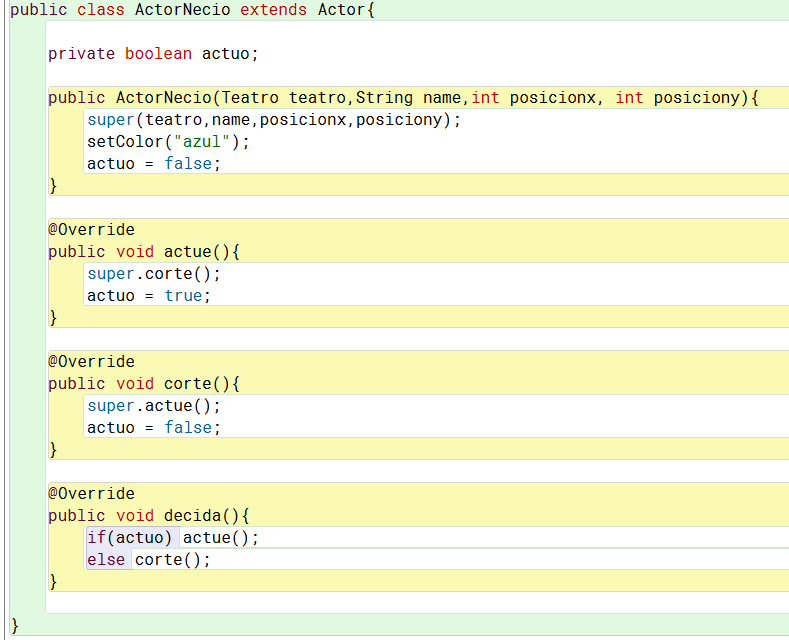
**6.** En este punto vamos a construir (diseño y código) el método que atiende el click del botón decidan de la interfaz: el método llamado decidan() de la clase Teatro. Construya el método, ejecute el programa y haga click en el botón Decidan. ¿Como quedan todos los actores después de esta orden? Capture la pantalla inicial y la final.

**Ciclo 2. Incluyendo a los actores necios**

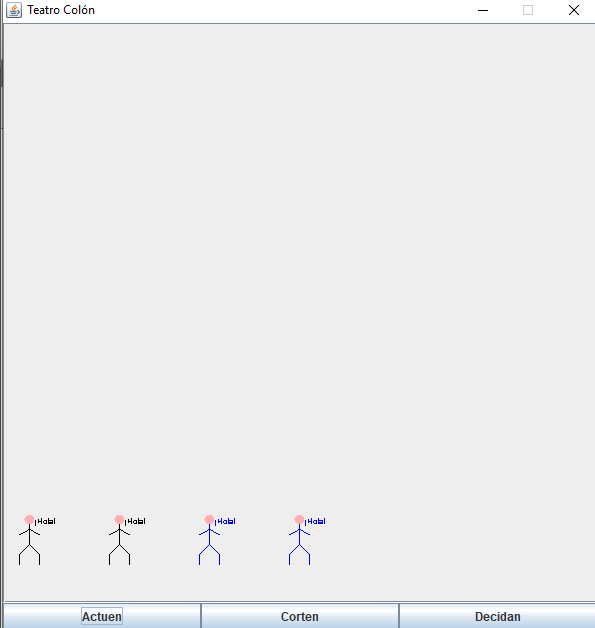
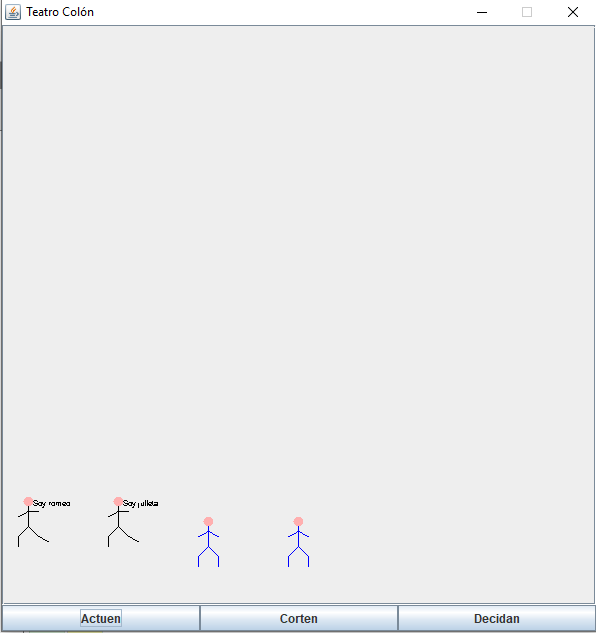
El objetivo de este punto es permitir recibir en el salón actores necios.

**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 debe hacer override en los métodos actue, corte y decida.

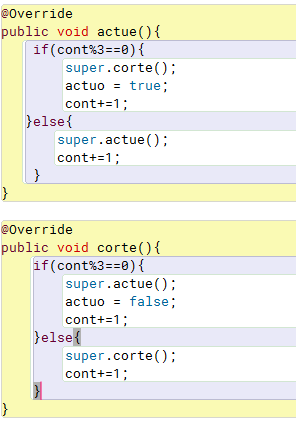


**2.** Adicione una pareja de actores necios, llámelos homer y bard, ejecute el programa y pídales a todos que actúen y que descansen. Capture la pantalla. ¿Qué pasa?

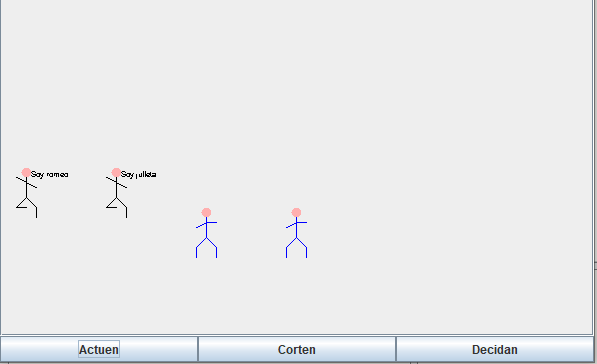
**3.** Ahora, los necios quieren sorprender con su necedad; es decir, sólo son necios cada tres veces. ¿Qué modificaría para lograr este comportamiento? ¡Hágalo!

Se debe poner un contador en los métodos del ActorNecio para que cuando sea 3, sea necio y haga el efecto contrario.



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

Se comportan normalmente hasta que sea 3 veces



**Ciclo 3. Adicionando luces**

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.

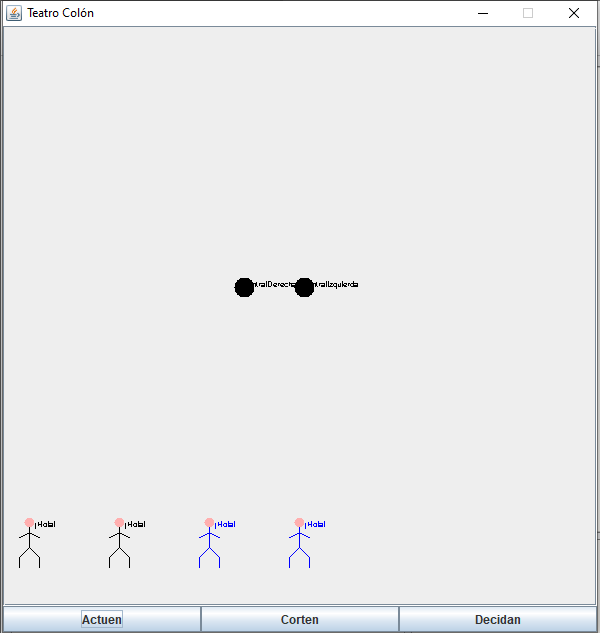
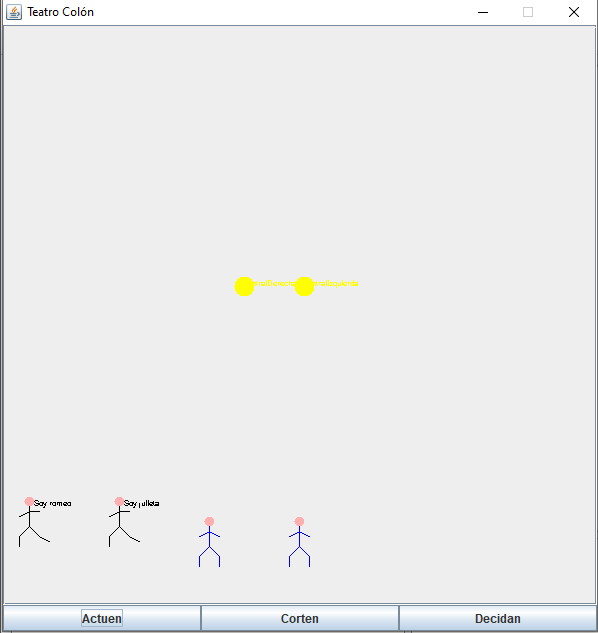
**5.** Construya la clase Luz para poder adicionarla en el Teatro. ¿qué cambios incluyó?

Se creó una clase Elemento que define a un elemento no persona dentro del teatro. La clase Luz hereda de Elemento e implementa la interfaz EnEscena para poder funcionar correctamente dentro del teatro.

**6.** Para aceptar este elemento, ¿debe cambiar en el código del Teatro en algo? ¿por qué?

No se debe cambiar nada, ya que la lista de Teatro es de EnEscena y la clase Luz implementa también esta interfaz, así que se puede añadir fácilmente dentro del teatro. Para poder dibujarlo se necesita hacer Override a un método de la interfaz, el método forma().

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



**Ciclo 4. Creando un nuevo actor: el perezoso**

El objetivo de este punto es incluir el actor perezoso, considerando que:

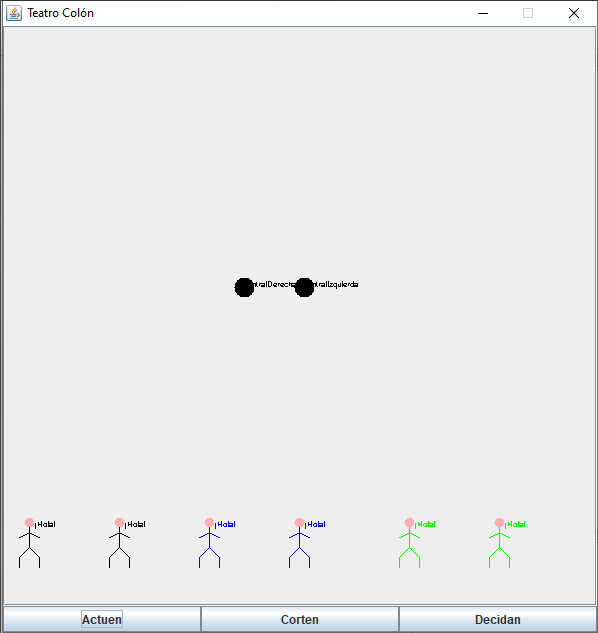
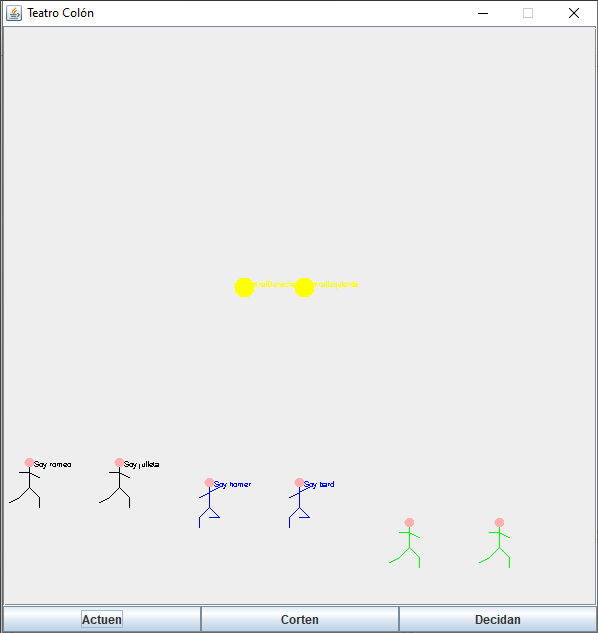
* Está vestido con color verde
* En lugar de avanzar, como romeo y julieta, sólo mueve brazos y piernas (primero los sube y luego los baja)
* Descansa sentandose
* Siempre decide descansar.
* Es un actor muy silencioso cuando actua. Cuando descansa llama a sus compañeros: “¡Aquí perezosos!”

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

Se crea la clase ActorPerezoso que hereda de Actor y hace Override a los métodos de actue() y corte() para cumplir los requisitos. No se debe cambiar nada en los diagramas de secuencia anteriores.

**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.



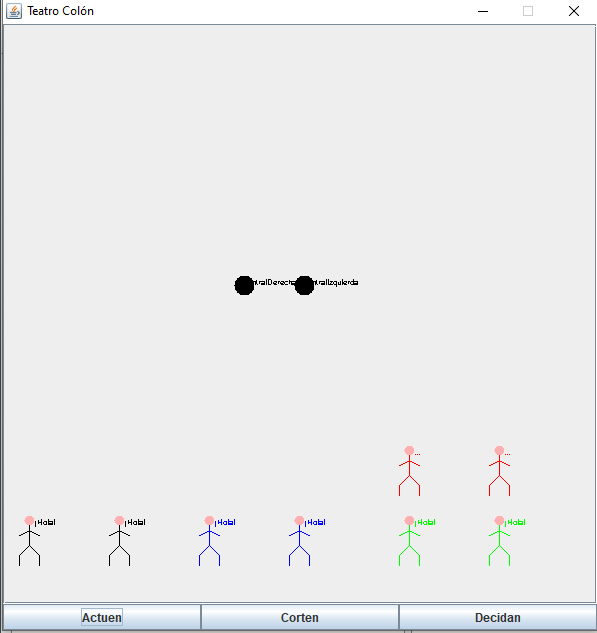
**Ciclo 5. Nuevo actor: Proponiendo y diseñando**

El objetivo de este punto es permitir recibir en un nuevo tipo de actor.

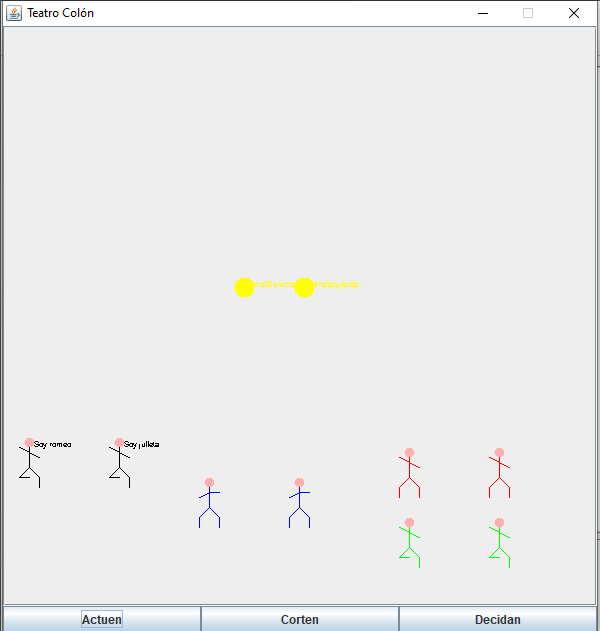
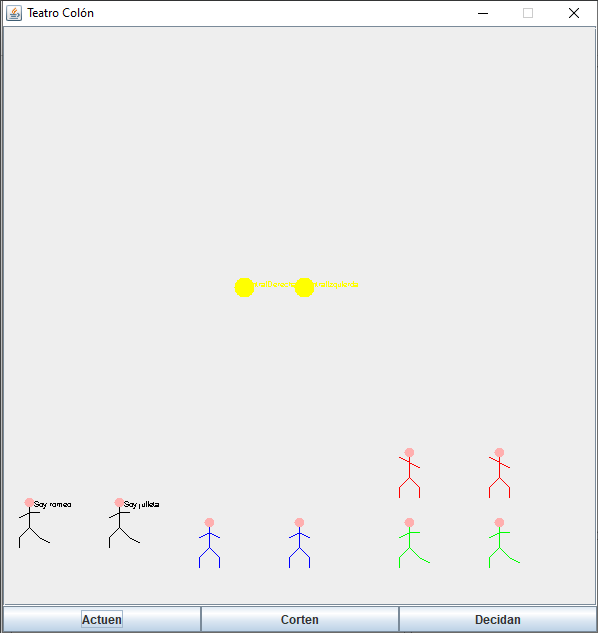
**1.** Propongan, describan e implementen un nuevo tipo de actor.

ActorTimido: Solamente mueve un brazo hacia arriba cuando actúa, no se desplaza y tampoco habla. Tiene traje de color rojo. Las palabras iniciales son tres puntos suspensivos.

**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.



Caso actuar: Solo mueve el brazo. No habla

Caso actuar luego de 3 veces al botón: No se desplaza, no habla y solamente ha movido el brazo.

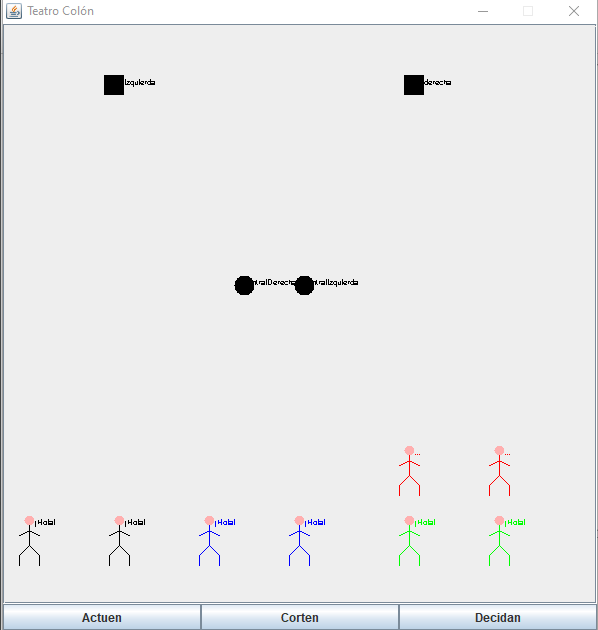
**Ciclo 6. Nuevo elemento: Proponiendo y diseñando**

El objetivo de este punto es permitir recibir en un nuevo elemento en el escenario

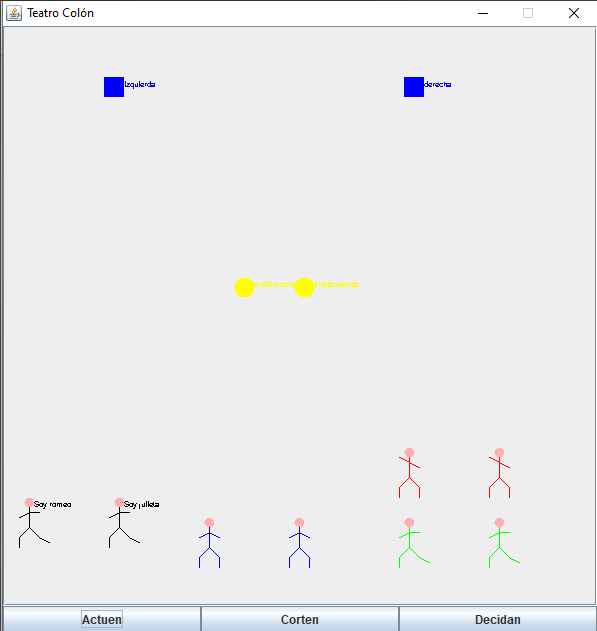
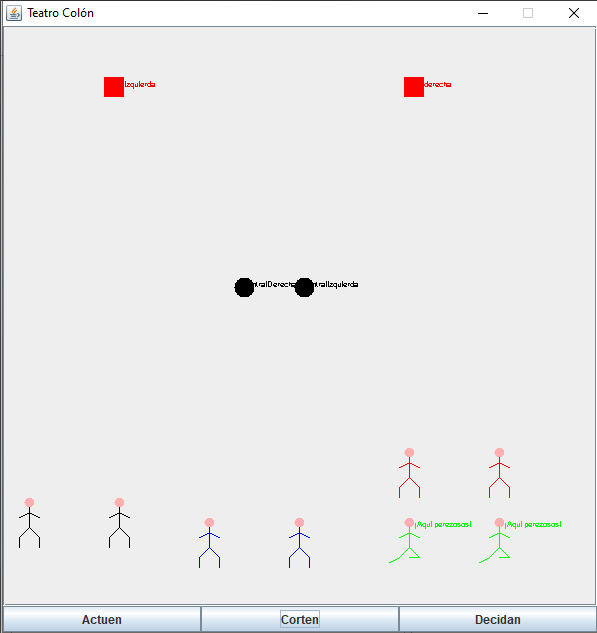
**1.** Propongan, describan e implementen un nuevo tipo de elemento

Pantalla: Esta ubicado en la parte superior del teatro. Inicia apagado(negro), cuando actúan, cambia a azul, al cortar cambia a rojo y al decidir cambia a verde.

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



Caso actuar: Cambia de color a azul

Caso corte: Cambia de color a rojo

**Empaquetando la versión final para el usuario.**

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

TeatroColon.jar

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

Se abre directamente el programa sin pasar por las fuentes.

**3.** ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

Solamente se necesita un archivo para ejecutar el programa, simplemente es ejecutar el archivo .jar y ya se tiene acceso al programa, y no se necesita estar accediendo a los paquetes para ejecutar el archivo. Por otra parte, hace portable el proyecto, y el manejo de JAR está dentro de un estándar en el API de Java.

**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**

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.

* 1. crear para archivos y para carpetas:
     1. MD nombre-de-carpeta.
     2. echo > archivo.txt
  2. borrar:
     1. rmdir [nombre carpeta] – Eliminamos un directorio (solo si esta vacio)
     2. rd/s nombre de la carpeta – Eliminar.
     3. del [nombre archivo] – Eliminamos un archivo
  3. listar su contenido:
     1. TYPE [nombreDelArchivo] archivo.txt para archivos en texto plano.
     2. Dir para ver contenido de las carpetas.
  4. copiar:
     1. copy fuente destino (dirección)
     2. Xcopy

**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

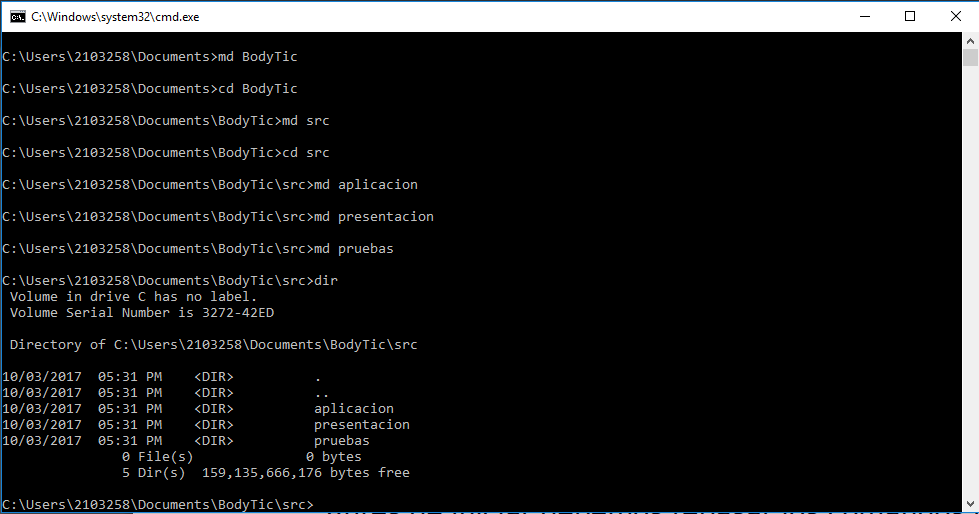
teatroColon

src

aplicacion

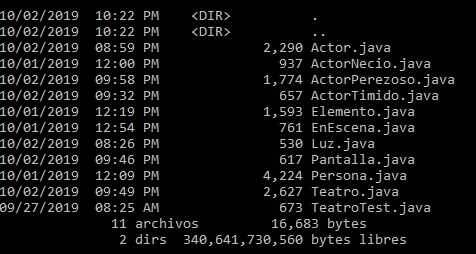
presentacion

pruebas



**3.** En el directorio copien únicamente los archivos \*.java del paquete de aplicación.

Consulte y capture el contenido de src/aplicación



**Estructura de proyectos java**

En java los proyectos se estructuran considerando tres directorios básicos.

teatroColon

src

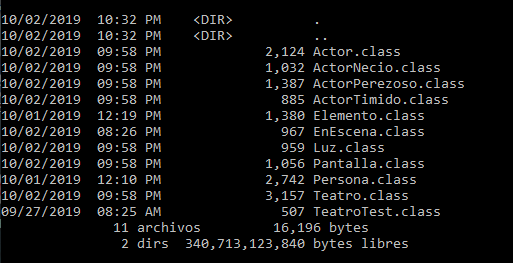
bin

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 los códigos fuente en lenguaje java (\*.java).
* en bin el bytecode o código máquina para la máquina virtual de java los archivos \*.class.
* en docs la documentación del codigo.

**2.** ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.Los archivos de compilación .class ya que es lo que la máquina entiende para poder ejecutar el proyecto.

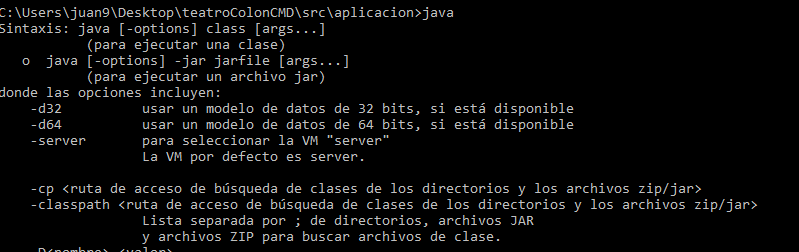


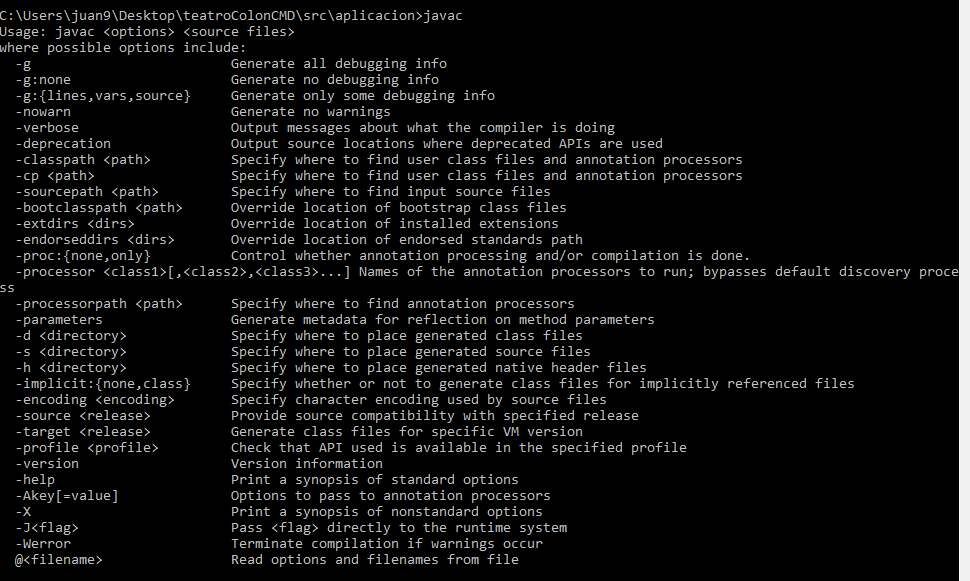
**Comandos de java**

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

* javac: llama al compilador y complia los archivos con código fuente Java (.java) en byte-codes Java (.class). Por defecto javac, genera los archivos .class en el mismo directorio del archivo fuente .java
* java: Para ejecutar el código compilado escribimos java \*.class.
* javadoc: genera la documentación de las clases en la carpeta especificada
* jar: el comando jar crea un archivo de tipo jar

**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.

* para java :-version y para javac: -version

**Compilando**

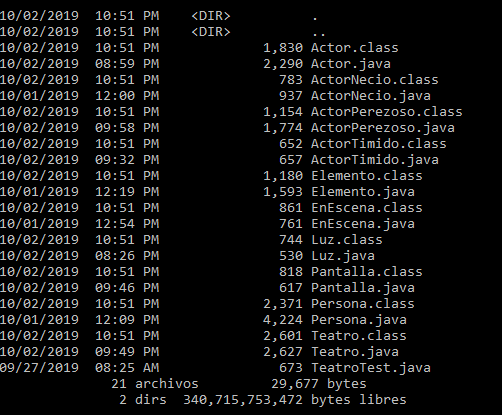
**1.** Utilizando el comando javac, desde el directorio raiz (desde teatroColon con una sóla instrucción), compile el proyecto. ¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto? Tenga presente que se pide un único comando y que los archivos compilados deben quedar en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

* javac -cp src src\presentacion\TeatroColonGUI.java

**2.** Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles

nuevos archivos aparecen ahora y dónde se ubican?

Los archivos .class aparecen en la carpeta de src



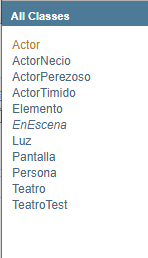
**Documentando**

**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?

Desde la carpeta src podemos usar: javadoc -d ../docs aplicacion

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

Abrimos el archivo allclasses-frame.html y empezamos a navegar por la clase que deseamos



**Ejecutando**

**1.** Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo utilizó este comando?

* javac -d bin -sourcepath src src\presentacion\TeatroColonGUI.java && java -classpath bin presentacion.TeatroColonGUI

**RETROSPECTIVA**

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

Hombre)

Eduard Arias: 10 horas

Juan Diaz: 10 horas

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

Únicamente falta lo último en pruebas desde consola. Todos los demás ciclos están completos.

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

importante?

--

**4**. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor

problema? ¿Qué hicieron para resolverlo?

Terminar todos los ciclos completos del proyecto, poder pasarlo a consola y usar los comandos para generar compilación y documentación.

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

resultados?

Trabajamos bien en conjunto, compartimos ideas y nos funcionó para terminar todos los ciclos correctamente, junto con toda la investigación necesaria.