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

## OBJETIVOS

Desarrollar competencias básicas para:

1. Aprovechar los mecanismos de la herencia y el uso de interfaces.
2. Organizar las fuentes en paquetes.
3. Usar la utilidad jar de java para entregar una aplicación.
4. Extender una aplicación cumpliendo especificaciones de diseño, estándares y verificando su corrección.
5. Vivenciar las prácticas XP : The project is divided into iterations.
6. Utilizar los programas básicos de java (javac, java, javadoc, jar), desde la consola.

## 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.
- ➔ En el foro de entrega deben indicar el estado de avance de su laboratorio y los problemas pendientes por resolver.
- ➔ Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

## DESARROLLO

### 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? b) ¿Cuántas clases tiene en total? ¿Cuántas tienen fuentes? c) ¿Cuál es la clase ejecutiva? ¿Por qué?
2. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

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

**Ciclo 1. Actuan 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?
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?
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?
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 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.

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

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

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

**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.

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

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

1. Consulte para qué sirven cada uno de los siguientes comandos:
   `javac`
   `java`
   `javadoc`
   `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.

**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.
2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?

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

**Ejecutando** [En lab03.doc]

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

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

**Empaquetando** [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` ?
2. ¿Cómo se ejecuta el proyecto empaquetado?

**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)
2. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)
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?
5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?