# PROGRAMACIÓN ORIENTADA A OBJETOS Herencia e interfaces y Java desde consola 2019-02 Laboratorio 3/6 Integrantes: Brayan Burgos y Daniel Alfonso Conociendo

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?  
  
Tiene dos paquetes, seis clases en total, la clase ejecutiva es teatroColonGUI ya que en esta está el programa principal (main) además desde este usa las demás clases que están en el paquete presentación.

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

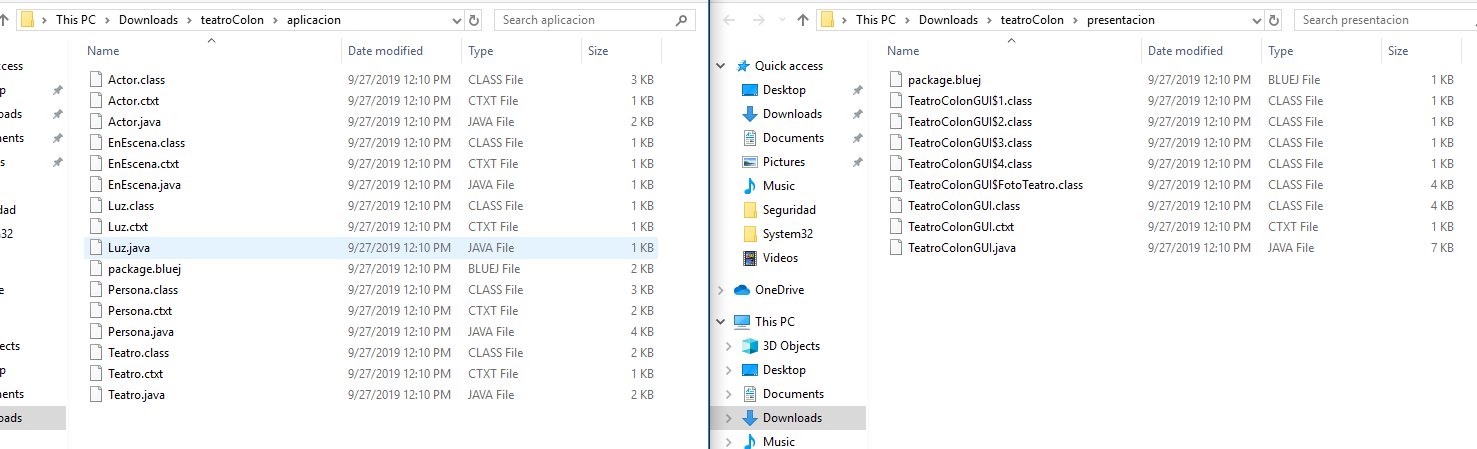
El programa ofrece una interfaz gráfica con tres acciones, Actúen, Corten y Decidan, pero no hace nada actualmente porque ninguno de estos métodos esta implementado  
  
Arquitectura general  
  
1. Consulte el significado de las palabras package e import de java. ¿Qué es un  
paquete? ¿Para qué sirve?

* ¿import en Java?  
    
  Importar es una palabra clave. La palabra clave import se usa para importar paquetes integrados y definidos por el usuario en su archivo fuente de Java para que su clase pueda referirse a una clase que está en otro paquete usando directamente su nombre.
* ¿Qué es un paquete y para qué sirve?  
  Los paquetes son una forma de organizar grupos de clases. Un paquete contiene un conjunto de clases relacionadas bien por finalidad, por ámbito o por herencia.

Los paquetes resuelven el problema del conflicto entre los nombres de las clases. Al crecer el número de clases crece la probabilidad de designar con el mismo nombre a dos clases diferentes.

Las clases tienen ciertos privilegios de acceso a los miembros dato y a las funciones miembro de otras clases dentro de un mismo paquete.

2. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su  
contenido. ¿Qué coincidencia hay entre paquetes y directorios?  
En el directorio se encuentra el archivo blueJ y el README que trae la descripción del proyecto teatroColon y en los subdirectorios encontramos los archivos .java de cada clase con el código correspondiente, los archivos .class de cada clase con el bytecode y los. ctxt



# 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.

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. Actúan 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?  
Usa un ArrayList de tipo EnEscena, puede recibir actores ya que la clase actores implementa la interface 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” como palabra

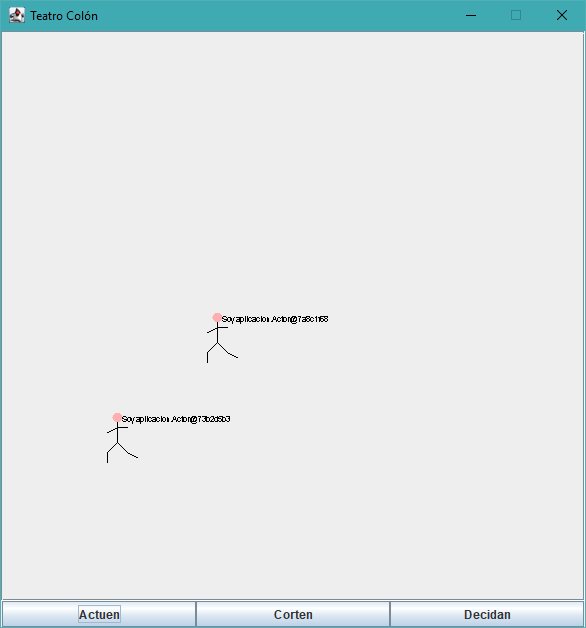
3. En el método algunosEnEscena de la clase Teatro cree dos actores en diferentes posiciones y acondiciónelos al Teatro llámelos romeo y julieta. Ejecute el programa y capture la pantalla. ¿Qué pasa ahora? ¿Pídales que entren en acción? ¿Qué pasa? ¿Por qué?

Creamos los dos actores pero en el momento que damos en acción o corten no hacen nada ya que los métodos acción corten y decidan de la clase Teatro no están implementados

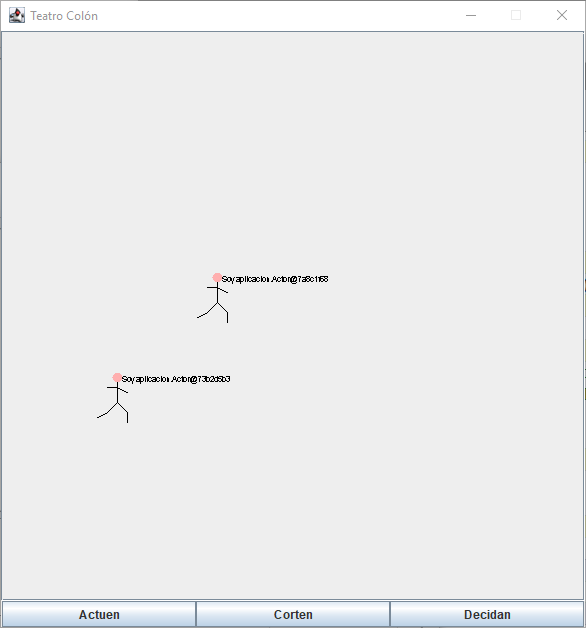
4. En este punto vamos a construir (diseño y código) el método que atiende el click del botón Accion de la interfaz: el método llamado accion () de la clase Teatro. Ejecute el programa y haga tres click en el botón Accion. ¿Cómo actúan romeo y Julieta? Capture la pantalla inicial y la final.

Cuando implementamos el método acción en Teatro y damos 3 clicks los actores van moviendo sus brazos, sus piernas y desplazándose, se adjunta pantalla inicial y final

Inicial



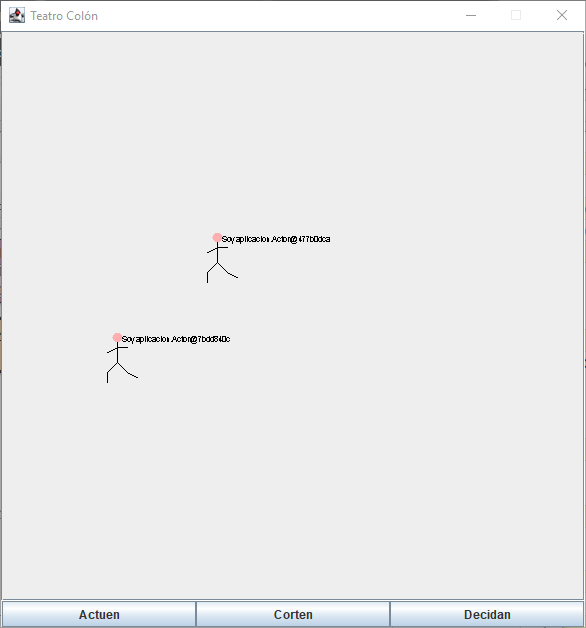
Final



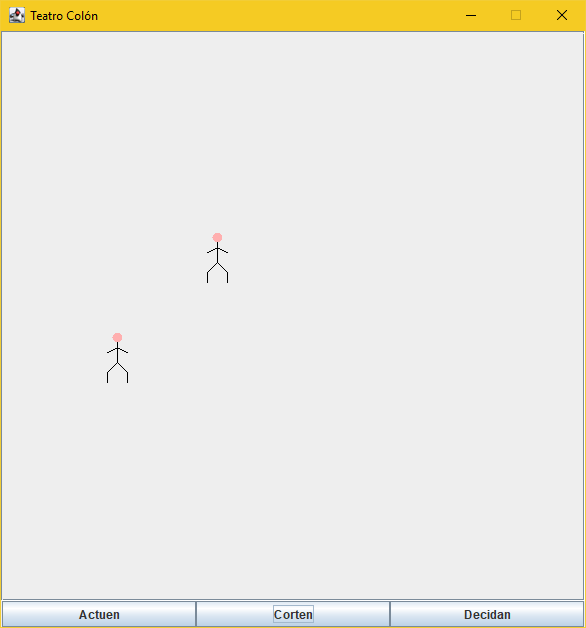
5. En este punto vamos a construir (diseño y código) el método que atiende el click del botón corten de la interfaz: el método llamado corten () de la clase Teatro. Construya el método, ejecute el programa y haga click en el botón Corten. ¿Cómo quedan todos los actores después de esta orden? Capture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

Cuando implementamos el método corte y cuando damos en corten los actores bajan sus brazos y enderezan sus piernas, y desaparece el mensaje que tienen.

Inicial



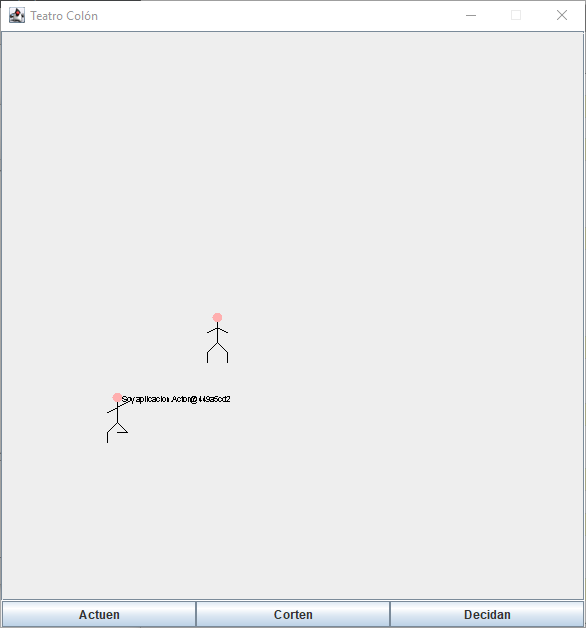
Final



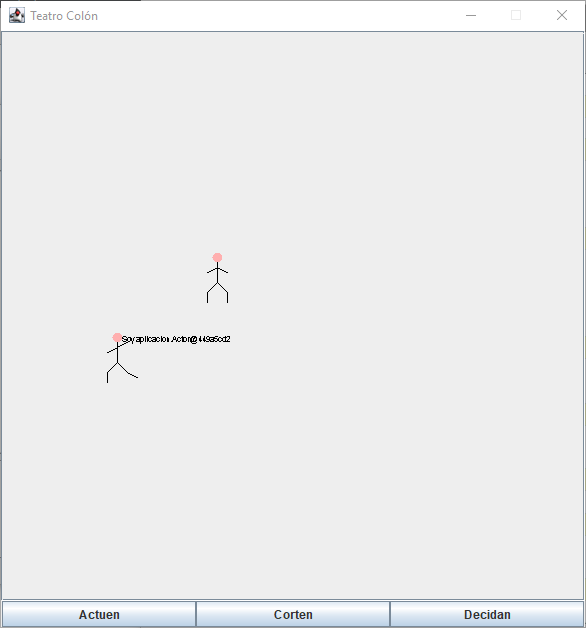
6. En este punto vamos a construir (diseño y código) el método que atiende el click del botón decidan de la interfaz: el método llamado corten () de la clase Teatro. Construya el método, ejecute el programa y haga click en el botón Decidan. ¿Cómo quedan todos los actores después de esta orden? Capture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

Cuando implementamos el método decidan los actores aleatoriamente deciden si cortar o actuar entonces tendríamos lo siguiente

Inicial



Final



# Ciclo 2. Incluyendo a los actores necios

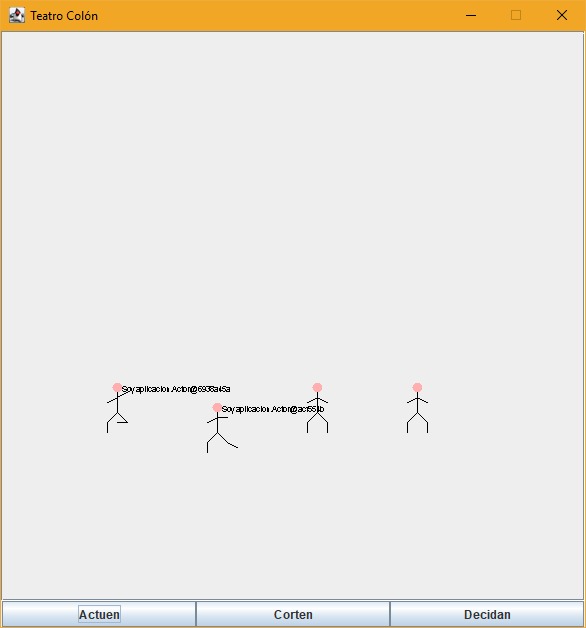
El objetivo de este punto es permitir recibir en el salón actores necios.

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)?

Al crear la clase actorNecio, como este es herencia de Actor necesitamos sobre-escribir los métodos de Actor y EnEscena los cuales son corte(), actue() y decida()

1. 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?

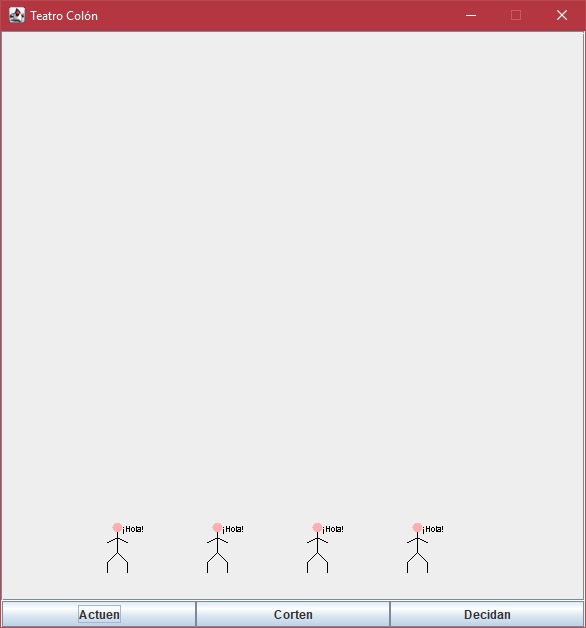
Cuando adicionamos los actoresNecios bart y homero junto con romeo y Julieta y damos actuar. Romeo y Julieta(Izquierda) mueven sus brazos y suben mientras que bart y homero(Derecha) no se mueven(cortan) como se observa



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

Cambiamos los comportamientos de actor necio, adicionamos un contador donde llevaremos las veces entonces cuando hagamos acción y el contador sea igual a 3 tiene que ser necio e igualar el contador a cero. De lo contrario tiene que llamar al método super actue de la clase actor para actue() y sumar uno al contador. Asi mismo con los métodos corte() y decida()

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



Cuando les pedimos actuar, los 4 actores lo harán, igualmente si le pedimos cortar, pero si volvemos a pedir actuar una vez más serian necios y harian lo contrario a romeo y Julieta.

# 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

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

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 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.

2. Implemente al actor Perezoso

3. Ahora sí, 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.

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

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

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.

# 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.

2. Consulte el comando java para ejecutar un archivo jar. Ejecútelo ¿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

Aplicación

Presentación

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 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

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

1. Utilizando el comando javac, desde el directorio raíz (desde automata con una   
sola 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

1. Utilizando el comando javadoc, desde el directorio raíz, genere la documentación (API) en formato HTML, en este directorio. ¿Cuál es el comando completo para generar esta documentación?

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

# Ejecutando

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

# Probando

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

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?