**PROGRAMACIÓN ORIENTADA A OBJETOS**

**INTERFAZ**

**2019-02**

**Laboratorio 5/6 CONTEXTO**

El objetivo es implementar el juego **Replicate** en versión usuario-computador.

# El trabajo se debe hacer desde **CONSOLA**

|  |  |
| --- | --- |
| El propuesto por ustedes ReplicateGUI | El acordado en laboratorio Replicate |
| **Vista - Controlador** | **Modelo** |

**Para la capa de presentación NO deben hacer pruebas de unidad ni diagramas de secuencia**

## DESARROLLO

### Directorios

El objetivo de este punto es construir un primer esquema para el juego  **Replicate** .

1. Preparen un directorio llamado **Replicate**  con los directorios src y bin y los subdirectorios presentación y aplicación.



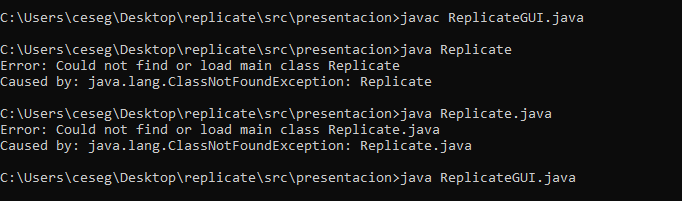
**Ciclo 0: Ventana vacía – Salir**

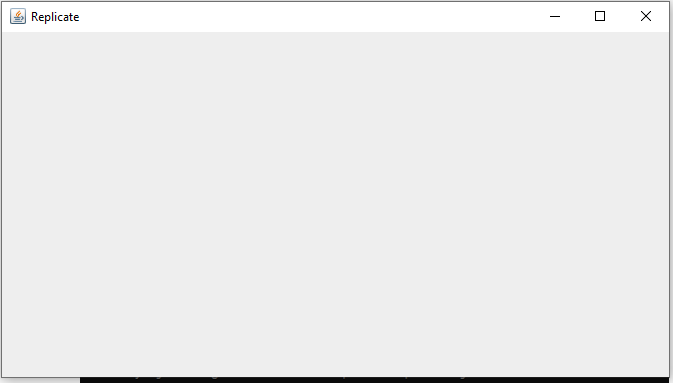
### [En \*.java y lab05.doc]

El objetivo es implementar la ventana principal de  **Replicate** con un final adecuado a la aplicación desde el icono de cerrar. Utilizar el esquema de prepareElementosprepareAcciones.

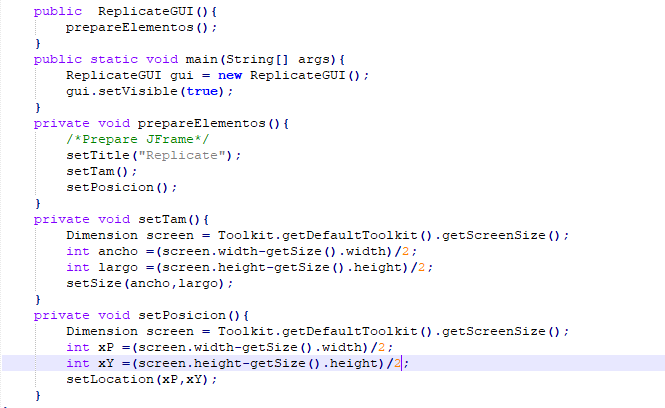
1. Construyan el primer esquema de la ventana de  **Replicate**  únicamente con el título “ **Replicate**”. Para esto cree la clase  **ReplicateGUI** como un JFrame con su creador, que sólo coloca el título, y el método main que crea un objeto **ReplicateGUI** y lo hace visible. Ejecútenlo. Capturen la pantalla.

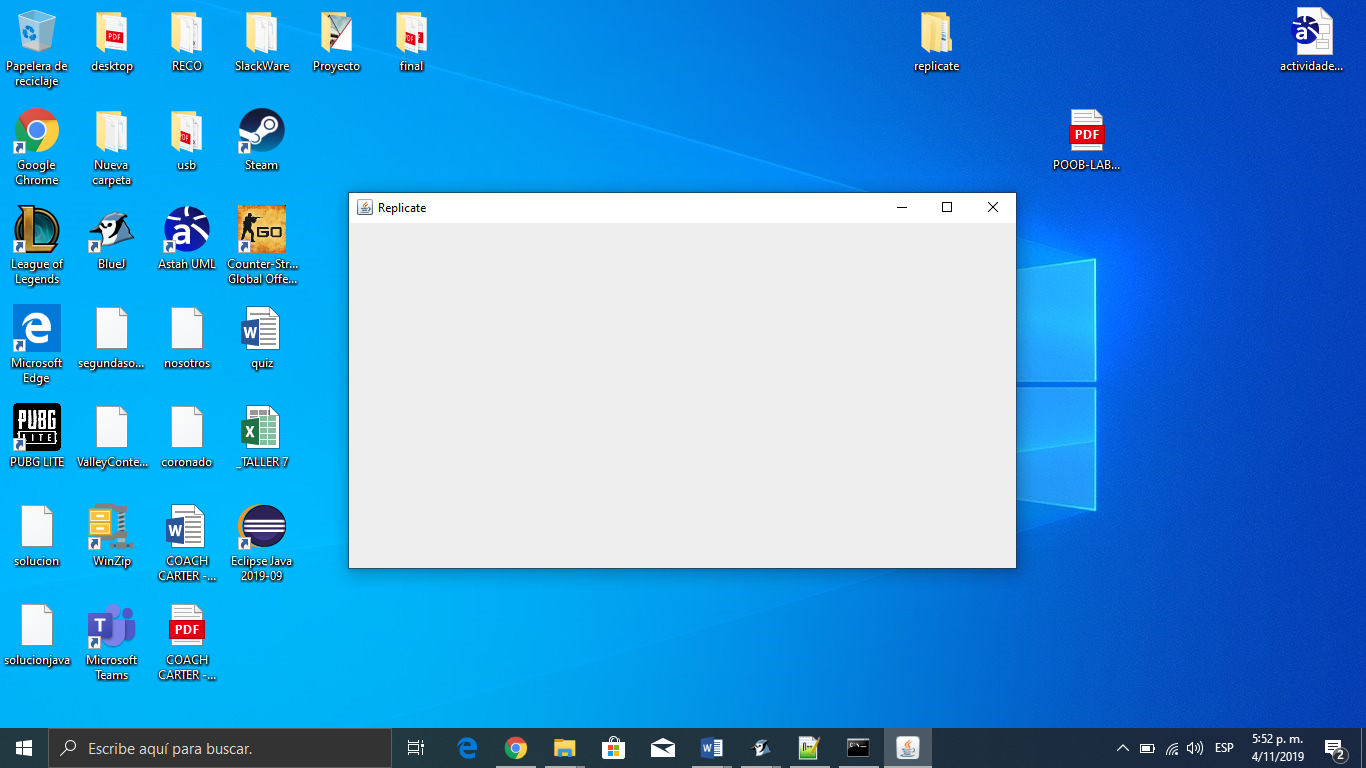
(Si la ventana principal no es la inicial en su diseño, después moverá el main al componente visual correspondiente)





1. Modifiquen el tamaño de la ventana para que ocupe un cuarto de la pantalla y ubíquenla en el centro (prepareElementos). Capturen esa pantalla.





1. Traten de cerrar la ventana. ¿Termina la ejecución? ¿Qué deben hacer para terminar la ejecución? ¿Por qué?

No termina porque se sigue ejecutando en consola a pesar de que no se vea gráficamente.

Desde consola, oprimir las teclas **ctrl+c** para cerrar la ejecución y comenzar una nueva ya que no permitiría hacer nada más desde consola.

1. Estudien en JFrame el método setDefaultCloseOperation. ¿Para qué sirve? ¿Cómo lo usarían en este caso?

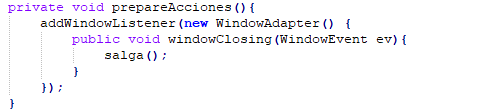
Sirve para ofrecer distintas alternativas a la hora de cerrar una aplicación dándole en el botón “x” desde no hacer nada, hasta cerrarla por completo y parar la ejecución. La usaría para cerrar la aplicación directamente una vez se oprima el botón “x”.

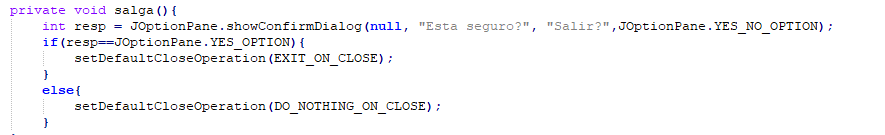
• DO\_NOTHING\_ON\_CLOSE (definido en WindowConstants): No hagas nada; requiere que el programa maneje la operación en el windowClosing método de un WindowListener objeto registrado . • HIDE\_ON\_CLOSE (definido en WindowConstants): Oculta automáticamente el marco después de invocar cualquier WindowListener objeto registrado .

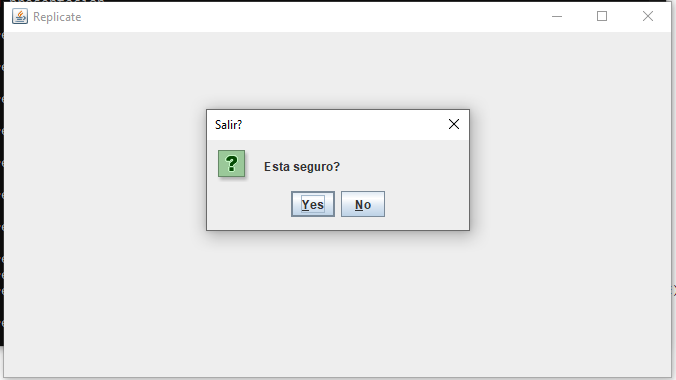
• DISPOSE\_ON\_CLOSE (definido en WindowConstants): Ocultar y eliminar automáticamente el marco después de invocar cualquier WindowListener objeto registrado .

• EXIT\_ON\_CLOSE (definido en JFrame): Salga de la aplicación usando el System exit método. Use esto solo en aplicaciones.

1. Preparen el “oyente” correspondiente al icono cerrar que le pida al usuario que confirme su selección. Para esto Implementen parcialmente el método prepareAcciones y el método asociado a la acción (salga). Ejecuten el programa y salgan del programa.





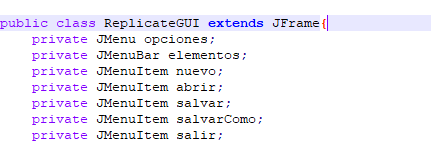


**Ciclo 1: Ventana con menú – Salir**

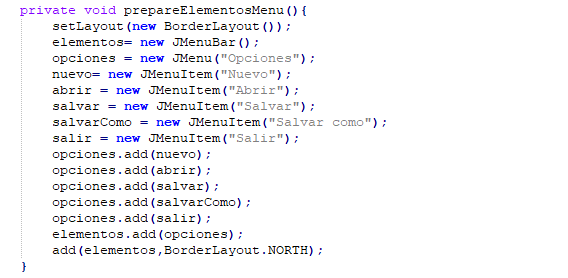
### [En \*.java y lab05.doc]

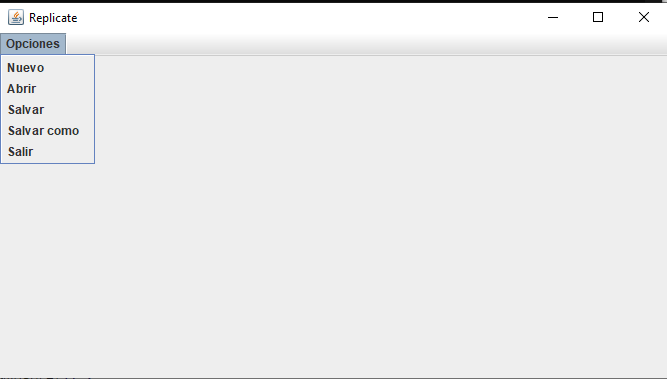
El objetivo es implementar un menú clásico para la aplicación con un final adecuado desde la opción del menú para salir. El menú debe ofrecer mínimo las siguientes opciones (Nuevo, Abrir – Salvar, Salvar Como – Salir) - No olviden los criterios de calidad del código.

1. Defina como atributos los componentes visuales necesarios del menú.

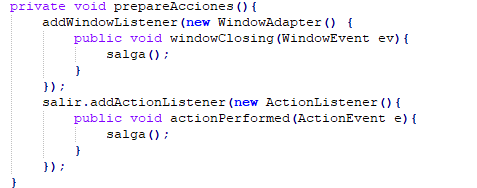


1. Construya la forma del menú propuesto en su diseño de interfaz (prepareElementos - prepareElementosMenu) . Ejecuten. Capture la pantalla.





1. Preparen el “oyente” correspondiente al icono cerrar con confirmación (prepareAcciones). Ejecuten el programa y salgan del programa.



**Ciclo 2: Salvar y abrir**

### [En \*.java y lab05.doc]

El objetivo es preparar la interfaz para las funciones de persistencia

1. Detalle el componente JFileChooser especialmente los métodos : JFileChooser, showOpenDialog, showSaveDialog, getSelectedFile.

**JFileChooser** proporciona un mecanismo simple para que el usuario elija un archivo

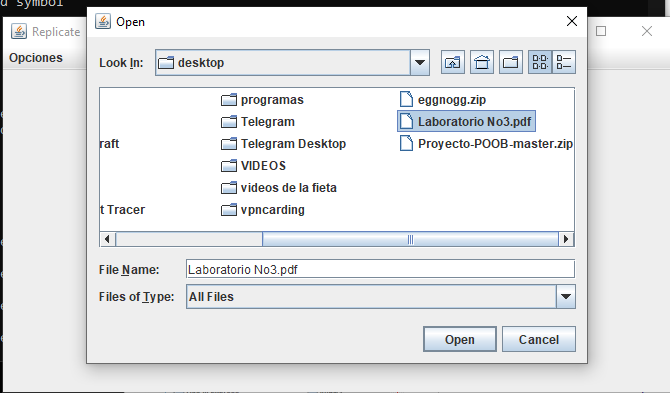
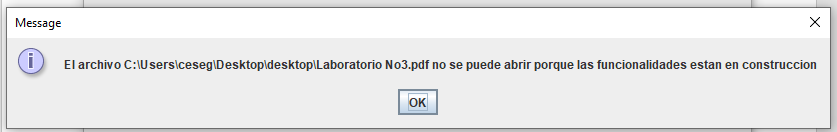
**JFileChooser()** Construye un JFileChooser apuntando al directorio predeterminado del usuario.

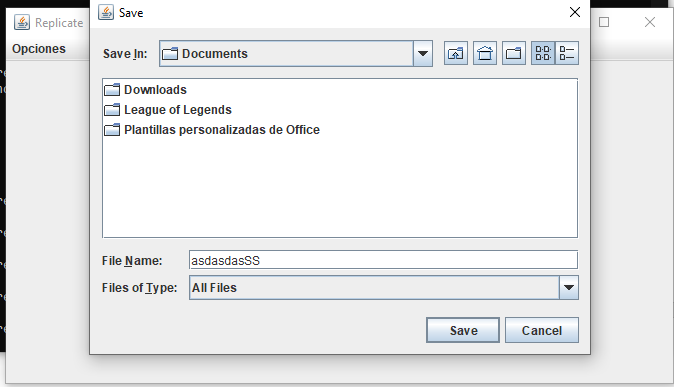
**showOpenDialog:** abre una ventana de dialogo donde tenemos la opción de abrir un archivo.

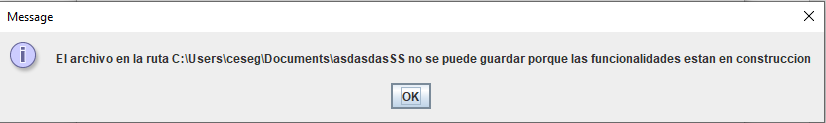
**showSaveDialog:** abre una ventana de dialogo donde tenemos la opción para guardar.

**getSelectedFile:** es para obtener el archivo seleccionado.

1. Implementen parcialmente los elementos necesarios para salvar y abrir. Al seleccionar los archivos indique que las funcionalidades están en construcción detallando la acción y el nombre del archivo seleccionado.
2. Ejecuten las dos alternativas y capture las pantallas más significativas.



****

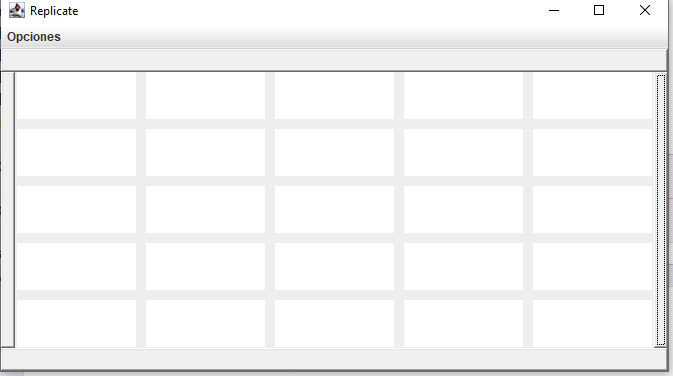
****

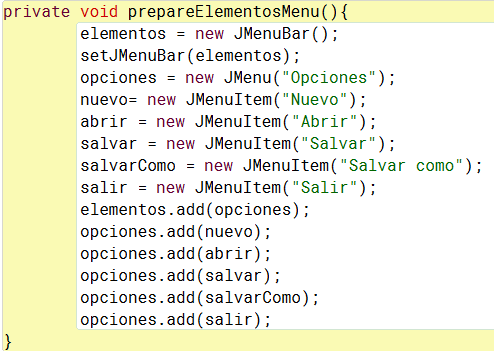
**Ciclo 3: Forma de la ventana principal**

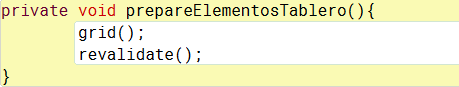
El objetivo es codificar el diseño de la ventana principal (todos los elementos de primer nivel)

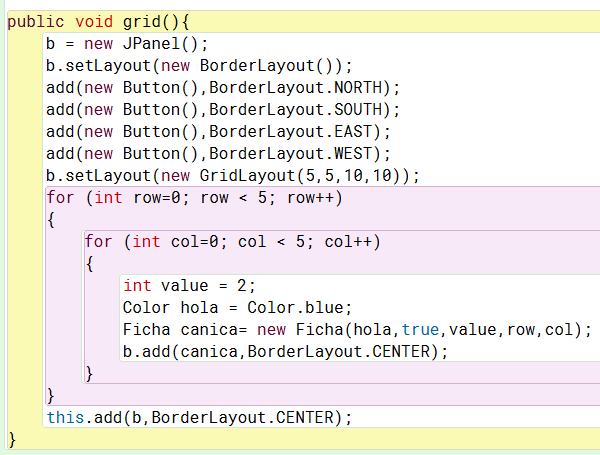
1. Definan como atributos privados todos los componentes visuales necesarios.
2. Continúe con la implementación del método prepareElementos().

Para la zona del tablero defina un método prepareElementosTablero y un método refresque() que actualiza la vista del tablero considerando, por ahora, un tablero inicial por omisión (el ejemplo del trabajo en clase) Este método lo vamos a implementar realmente en otros ciclos. Ejecuten y capturen esta pantalla.









**Ciclo 4: Cambiar color**

### [En \*.java y lab05.doc]

El objetivo es implementar este caso de uso.

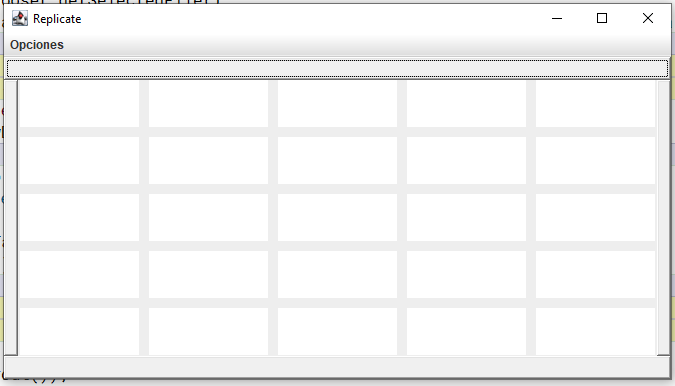
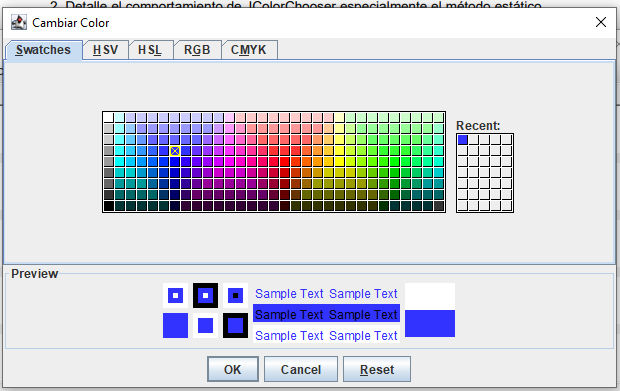
1. Expliquen los elementos necesario para implementar este caso de uso.

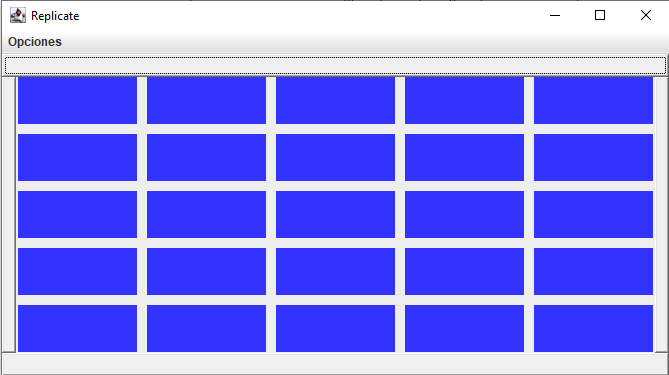
Necesitamos un Menú al cual llamaremos configuración para poder desplegar los iconos que permitan cambiar el color. Se prepara el “oyente” correspondiente al icono “Change CPU Color” y “Change Player Color” que le pida al usuario que elija el color que desea Para esto se añadirá cada oyente al método prepareAcciones() y los métodos asociados a las acciones de cambiar el color.

1. Detalle el comportamiento de JColorChooser especialmente el método estático showDialog

JColorChooser proporciona un panel de controles diseñados para permitir a un usuario manipular y seleccionar un color public static Color showDialog(Component, String title, Color initialColor) throws HeadlessException Muestra un diálogo y un selector de color modal hasta que se oculta el diálogo. Si el usuario presiona el botón "Aceptar", este método oculta / elimina el cuadro de diálogo y devuelve el color seleccionado. Si el usuario presiona el botón "Cancelar" o cierra el cuadro de diálogo sin presionar "Aceptar", este método oculta / elimina el cuadro de diálogo y devuelve nulo.

1. Implementen los componentes necesarios para cambiar el color de las casas y almacenes.
2. Ejecuten el caso de uso y capture las pantallas más significativas.





**Ciclo 5: Modelo Replicate**

### [En \*.java y lab05.doc]

El objetivo es implementar la capa de aplicación para **Replicate** .

1. Construya los métodos básicos del juego (**No olvide MDD y TDD)**
2. Ejecuten las pruebas y capturen el resultado.

**Ciclo 6: Jugar**

### [En \*.java y lab05.doc]

El objetivo es implementar el caso de uso jugar.

1. Adicione a la capa de presentación el atributo correspondiente al modelo.
2. Perfeccionen el método refresque() considerando la información del modelo de dominio.
3. Expliquen los elementos necesarios para implementar este caso de uso.
4. Implementen los componentes necesarios para jugar .
5. Ejecuten el caso de uso y capture las pantallas más significativas.

**Ciclo 7: Reiniciar**

### [En \*.java y lab05.doc]

El objetivo es implementar este caso de uso.

1. Expliquen los elementos a usar para implementar este caso de uso.
2. Implementen los elementos necesarios para reiniciar
3. Ejecuten el caso de uso y capture las pantallas más significativas.

**Ciclo 8: Cambiar el tamaño del juego: casas y semillas**

### [En \*.java y lab05.doc]

El objetivo es implementar este caso de uso.

1. Expliquen los elementos a usar para implementar este caso de uso
2. Implementen los elementos necesarios para cambiar el tamaño del juego
3. Ejecuten el caso de uso y capture las pantallas más significativas.

**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
3. Considerando la práctica XP del laboratorio ¿por qué consideran que es 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?