**PROGRAMACIÓN ORIENTADA A OBJETOS**

**INTERFAZ**

**2019-02**

**Laboratorio 5/6**

**Eduard Arias – Juan Diaz**

**DESARROLLO**

**Directorios**

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

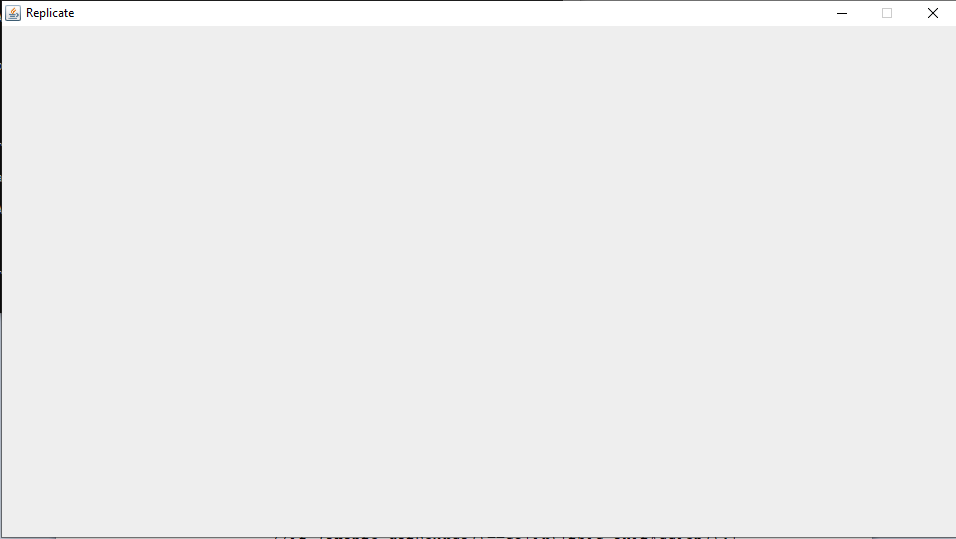
**1.** Preparen un directorio llamado Replicate con los directorios src y bin y los subdirectorios presentación y aplicación.

**Ciclo 0: Ventana vacía – Salir**

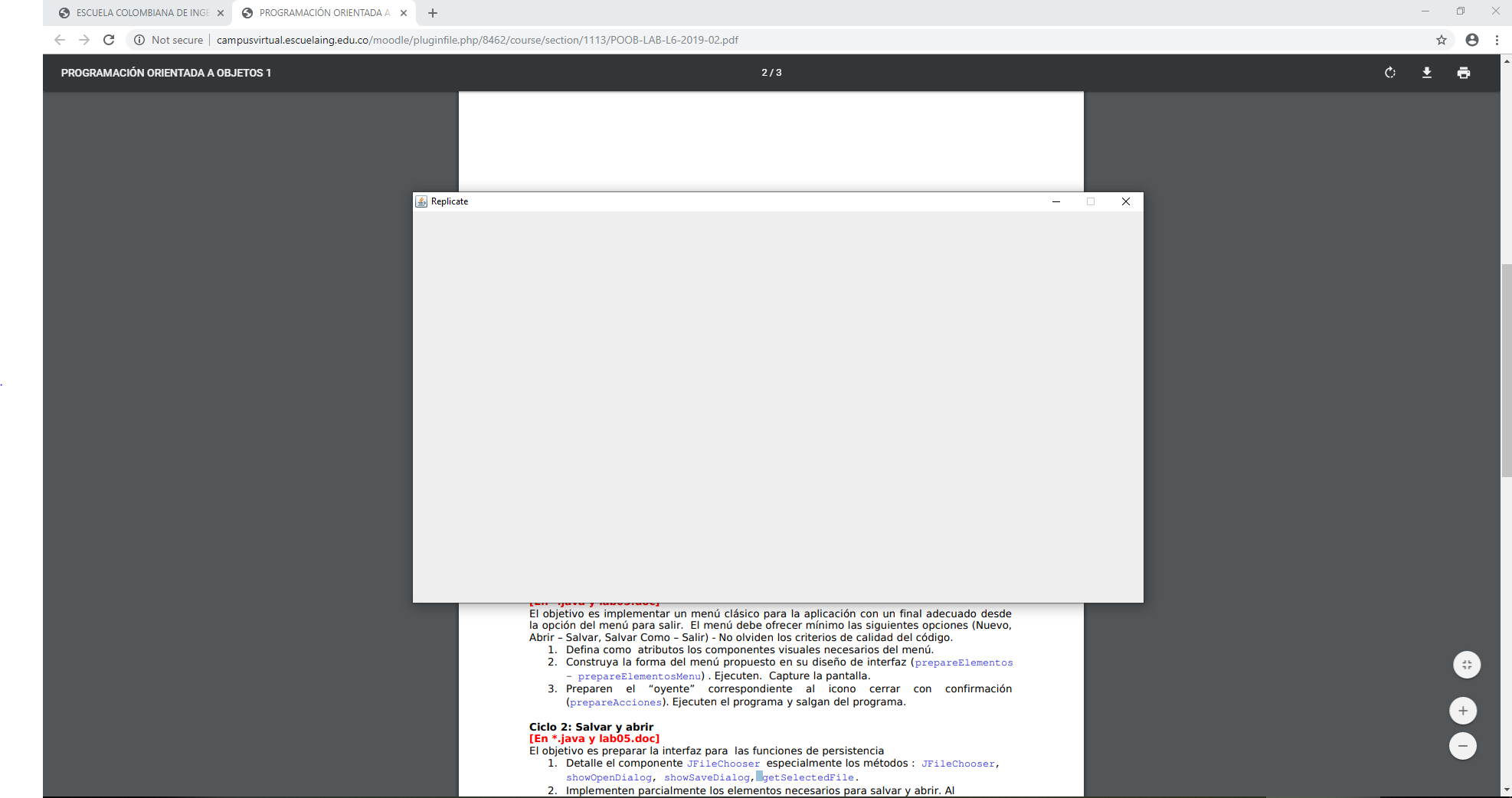
El objetivo es implementar la ventana principal de Replicate con un final adecuado a la

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)



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



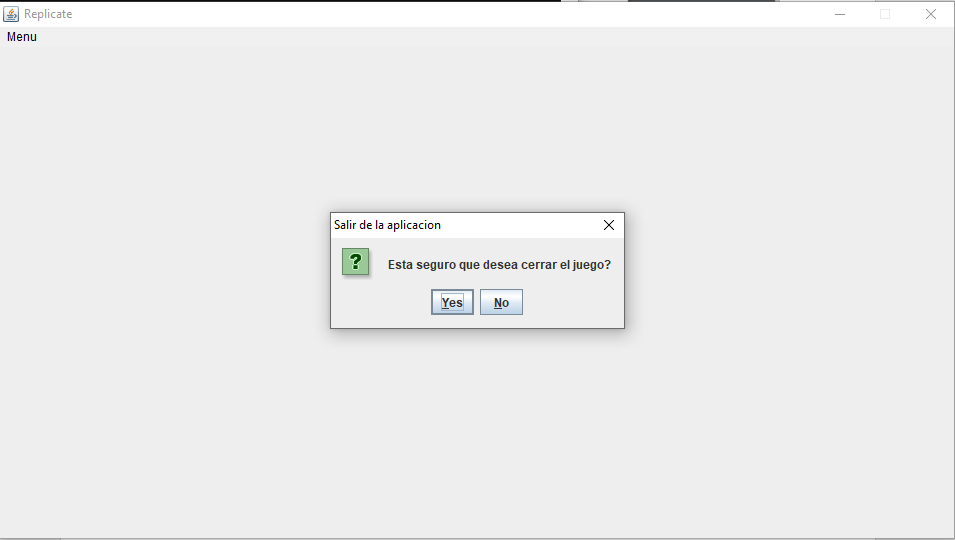
**3.** Traten de cerrar la ventana. ¿Termina la ejecución? ¿Qué deben hacer para terminar la ejecución? ¿Por qué?

Se cierra la ventana pero el proceso no termina. Hay que añadir un acción salir al hacer click en la “X” de la ventana del juego.

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

Sirve para poner qué acción debe hacer el programa al cerrar la ventana. En este caso lo usaremos para que al cerrar la ventana se cierre la ejecución del programa.

**5.** 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.



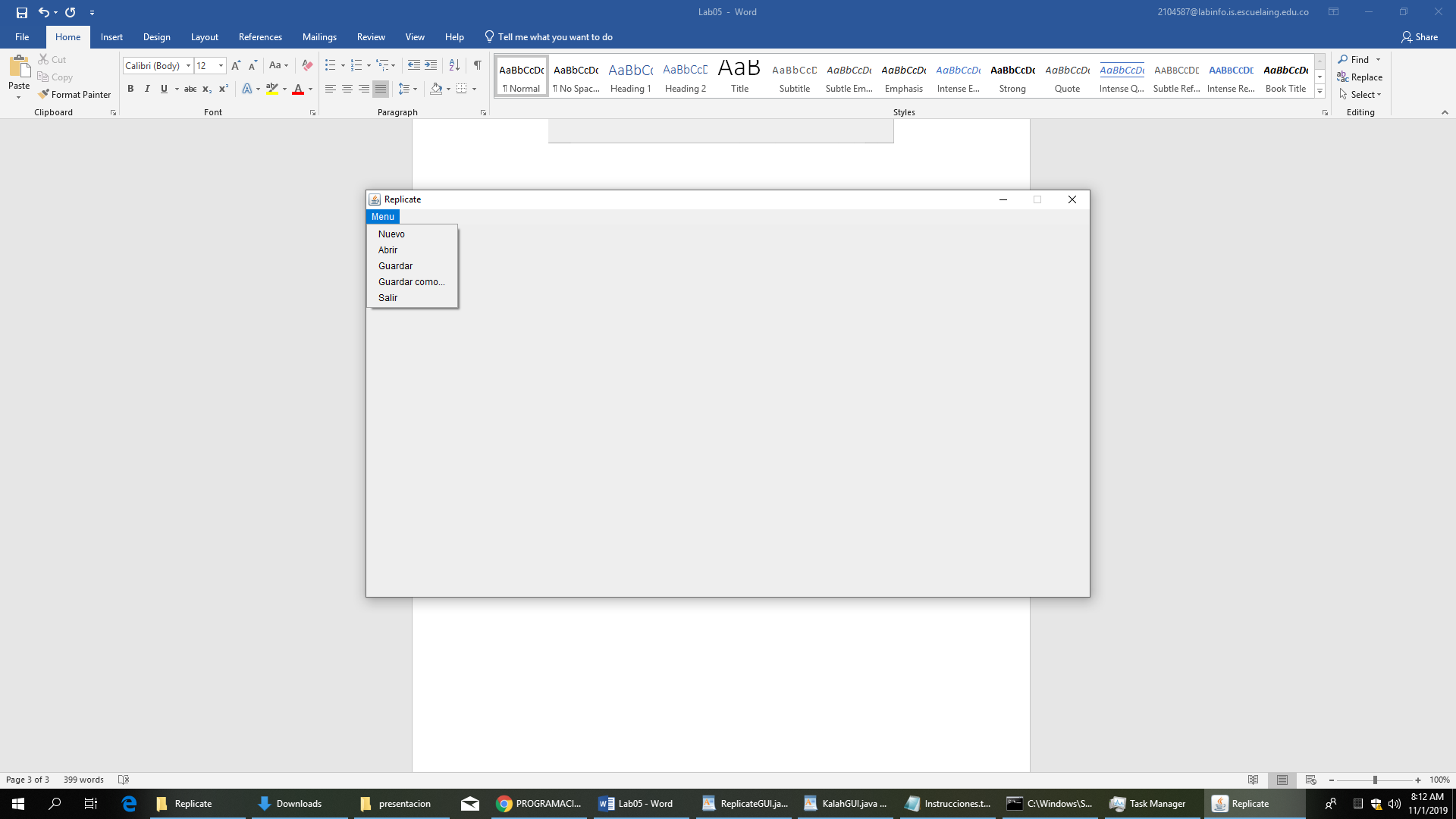
**Ciclo 1: Ventana con menú – Salir**

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.

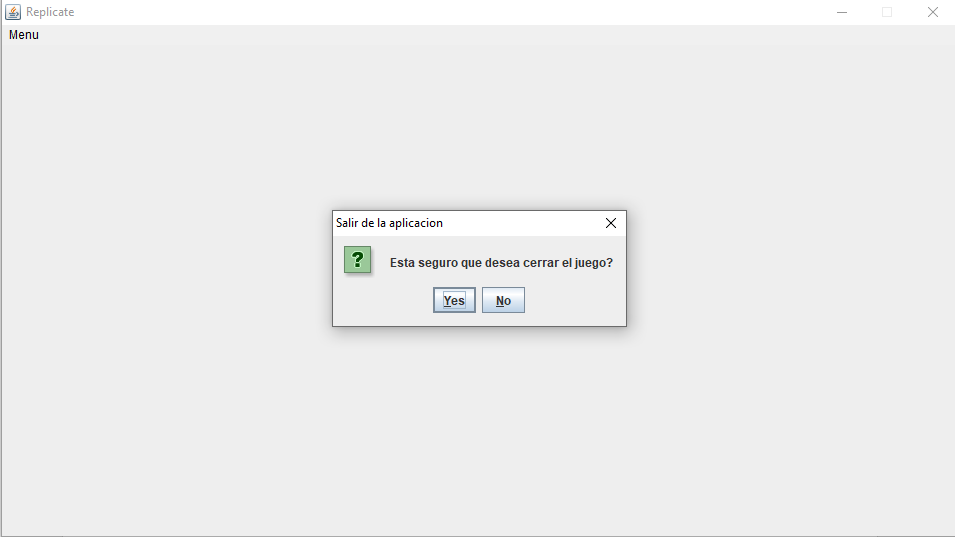
**1.** Defina como atributos los componentes visuales necesarios del menú.

Definimos un método prepareElementosMenu().

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



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



**Ciclo 2: Salvar y abrir**

El objetivo es preparar la interfaz para las funciones de persistencia

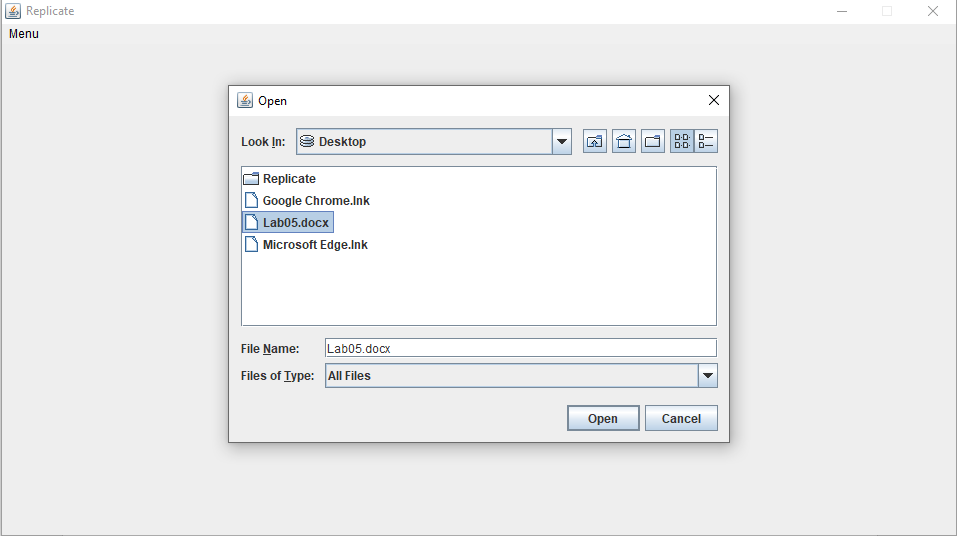
**1.** Detalle el componente JFileChooser especialmente los métodos: JFileChooser, showOpenDialog, showSaveDialog, getSelectedFile.

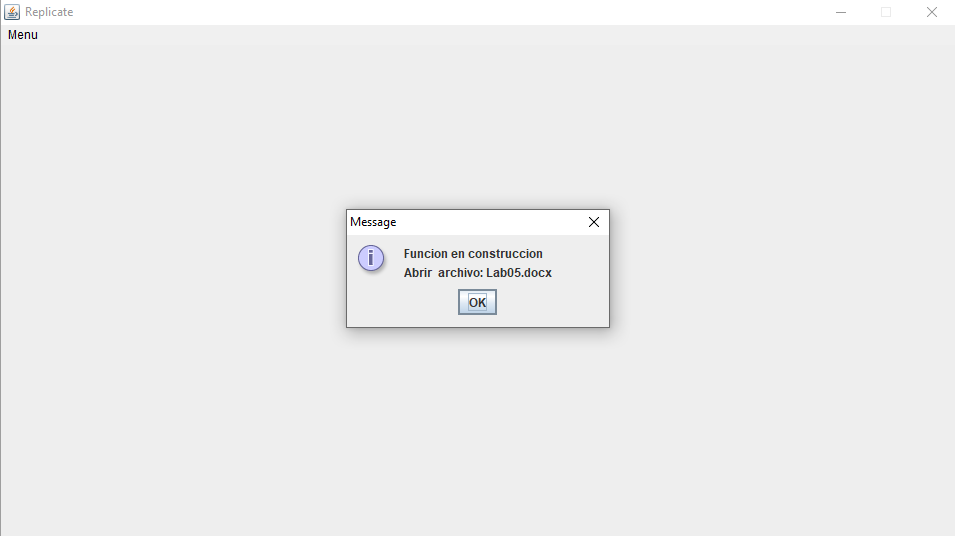
Son métodos usados para la seleccion de archivos, guardar el archivo seleecionado y diálogos de texto. Al utilizarlos, abren una pequeña pantalla de selección de archivo similar al explorador de Windows donde se puede navegar por las carpetas.

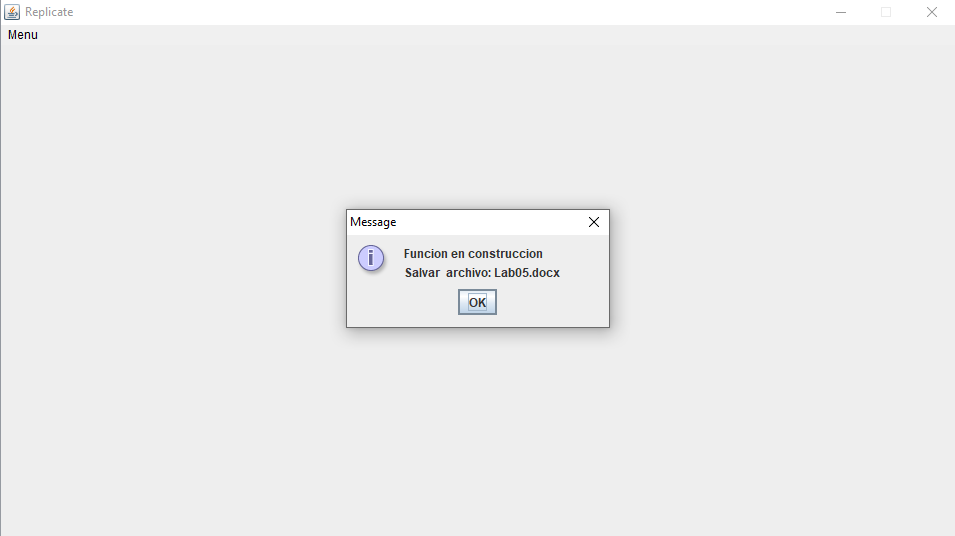
**2.** Implementen parcialmente los elementos necesarios para salvar y abrir. Al seleccionar los archivos indique que las funcionalidades están en construcción detallando la acción y el nombre del archivo seleccionado.

Se implementó acciones con estos nombres y con sus funcionalidades.

**3.** 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.

Se definieron Layouts, botones y labels para la implementación de los elementos visibles, y un método prepareElementosTablero().

**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. Este método lo vamos a implementar realmente en otros ciclos. Ejecuten y capturen esta pantalla.



**Ciclo 4: Cambiar color**

El objetivo es implementar este caso de uso.

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

**2.** Detalle el comportamiento de JColorChooser especialmente el método estático

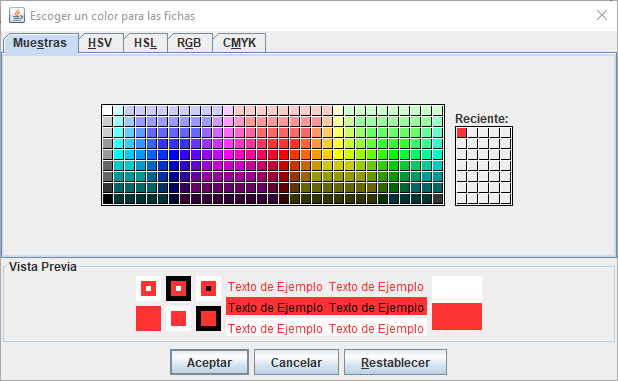
showDialog

JColorChooser desplega un panel con una paleta de colores para seleccionar, y gracias al showDialog podemos comunicar nuestra selección con el atributo que deseamos cambiar, en este caso el color del tablero.

**3.** Implementen los componentes necesarios para cambiar el color

**4.** Ejecuten el caso de uso y capture las pantallas más significativas.







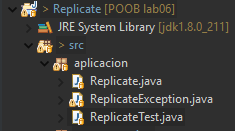
**Ciclo 5: Modelo Replicate**

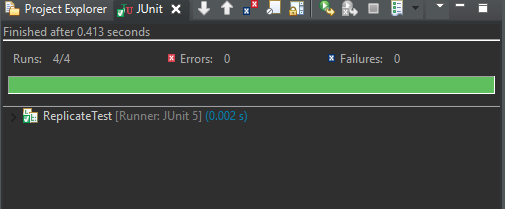
El objetivo es implementar la capa de aplicación para **Replicate** .

**1**. Construya los métodos básicos del juego **(No olvide MDD y TDD)**

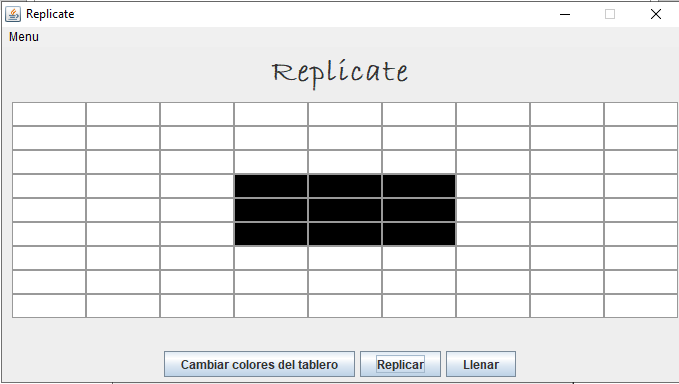
Se construyeron los métodos para replicar un estado utilizando el modelo De aplicación “Replicate”. Además, se construyeron las clases de manejo de excepciones y las pruebas correspondientes.

**2**. Ejecuten las pruebas y capturen el resultado.





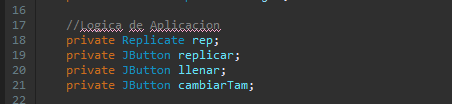




**Ciclo 6: Jugar**

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.

Se modificó para que utilizara la información que obtuviera de la lógica de “rep”.

**3**. Expliquen los elementos necesarios para implementar este caso de uso.

Lo único necesario fue que cada vez que se necesitara un servicio del caso de uso, se utilizara el homólogo de “aplicación”, para que después la capa de presentación pidiera la información e imprimirla en pantalla.

**4**. Implementen los componentes necesarios para jugar .

**5**. Ejecuten el caso de uso y capture las pantallas más significativas.







Después de replicar:



**Ciclo 7: Reiniciar**

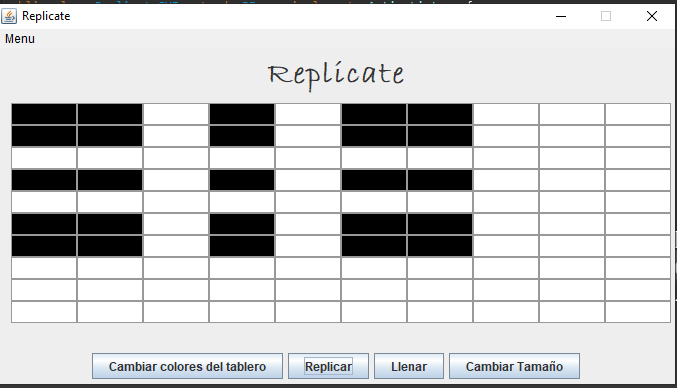
El objetivo es implementar este caso de uso.

**1**. Expliquen los elementos a usar para implementar este caso de uso.

Para lograr esta implementación se pedirá a “aplicacion” que nos dé un juego completamente nuevo, para luego colocarlo en pantalla. Dejamos encargado a la parte del menú: “nuevo” que resolviera este caso de uso.

**2**. Implementen los elementos necesarios para reiniciar

**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)

* Eduard: 6 horas
* Juan: 8 horas

**2**. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

El simulador puede resolver cualquier caso de uso con excepción del último, debido a que no encontramos una manera eficiente de cambiar la lógica y la presentación en esta parte (nos generaba una excepción: NullPointerException).

**3**. Considerando la práctica XP del laboratorio ¿por qué consideran que es importante?

Consideramos que la más importante fue **Testing** [tests](http://www.extremeprogramming.org/rules/bugs.html) When [a bug is found](http://www.extremeprogramming.org/rules/bugs.html) tests are created. , debido a que en la parte de aplicación se generaban nuevos errores que se debieron corregir eficazmente.

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

problema? ¿Qué hicieron para resolverlo?

Nuestro mayor logro fue crear una pequeña aplicación con interfaz de manera exitosa, ya que esta experiencia nos servirá para construir cosas mas complejas en el futuro. La mayor dificultad que tuvimos fue aprender a usar todos los componentes (necesarios) de una interfaz gráfica en java.

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

resultados?

No solamente pudimos dividir la arquitectura del software en la requerida (MVC), sino que pudimos “dividirnos” a nosotros también. Es decir, pudimos ser eficientes en ambos roles backend y frontend con gran éxito.