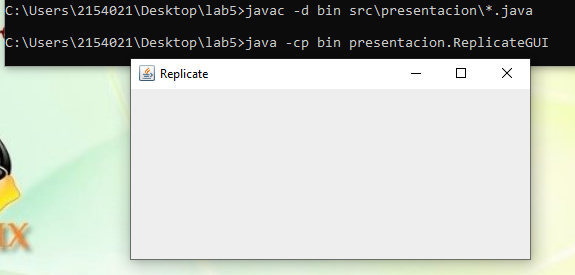
**PROGRAMACIÓN ORIENTADA A OBJETOS**   
**INTERFAZ**   
**2019-02**   
**Laboratorio 5/6**   
  
**CONTEXTO:** El objetivo es implementar el juego Replicate en versión usuario-computador.   
El trabajo se debe hacer desde CONSOLA. Para la capa de presentación NO deben hacer pruebas de unidad ni diagramas de secuencia

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

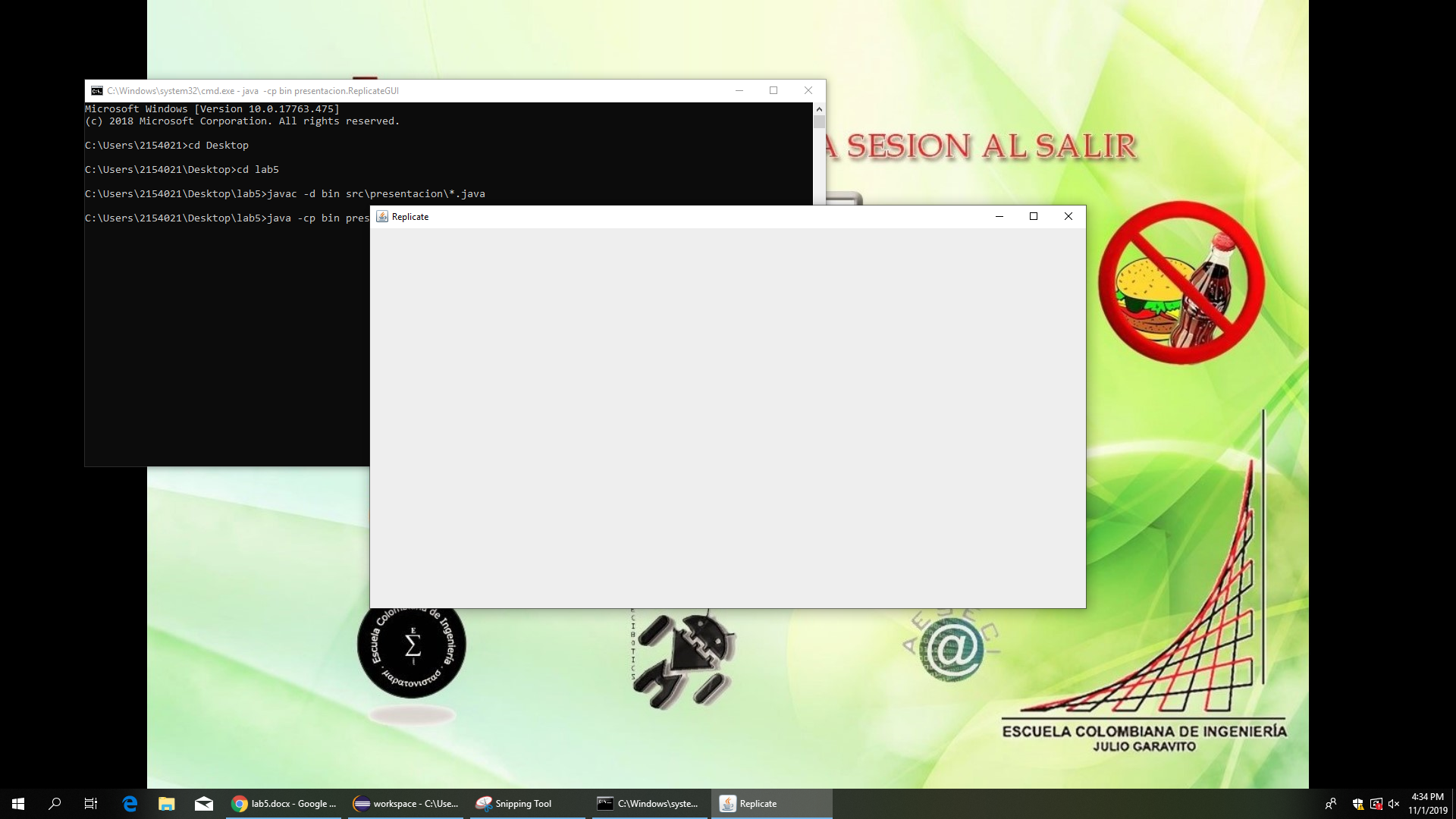
**DESARROLLO**   
Directorios: El objetivo de este punto es construir un primer esquema para el juego **Replicate**. 1. Preparen un directorio llamado **Replicate** con los directorios src y bin y los subdirectorios presentación y aplicación.

**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 prepareElementos prepareAcciones.

**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?** No se termina la ejecución, se debe usar Ctrl+C para que se cierre a la fuerza porque no está programado el cierre por el medio “normal” de cerrar la ventana.

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

**public void setDefaultCloseOperation (operación int)**: Establece la operación que se realizará de forma predeterminada cuando el usuario inicie un "cierre" en este marco. Debe especificar una de las siguientes opciones:

**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.

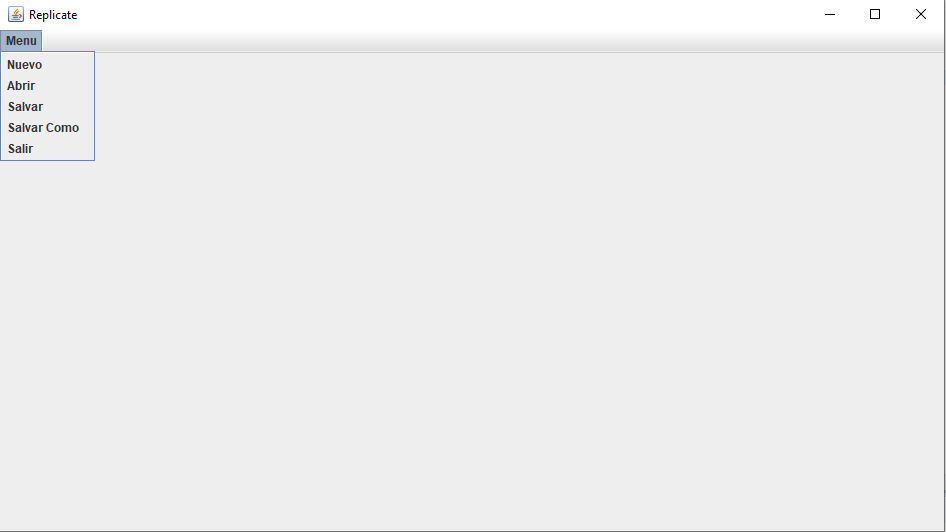
Usaríamos la última opción dado que es la más cercana a el requerimiento que se necesita

**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**   
**[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.**

**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**   
**[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.

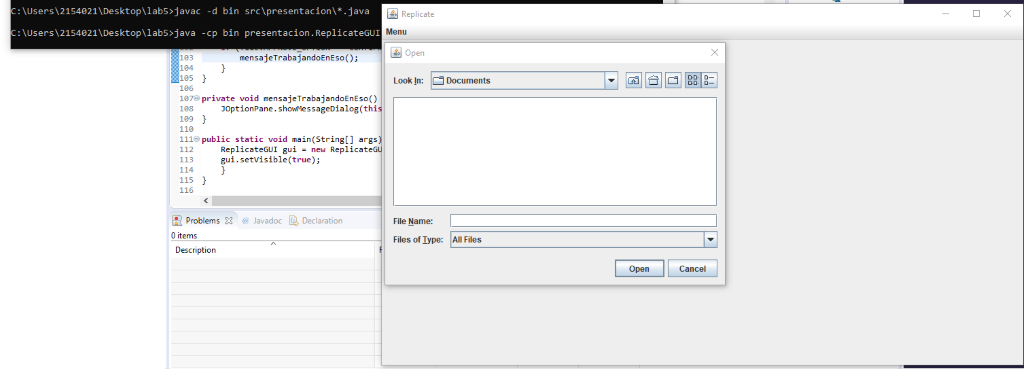
**showOpenDialog(Component parent) throws HeadlessException** Abre un diálogo de selección de archivos "Abrir archivo". Tenga en cuenta que el siguiente texto que aparece en el botón de aprobación está determinado por la L&F

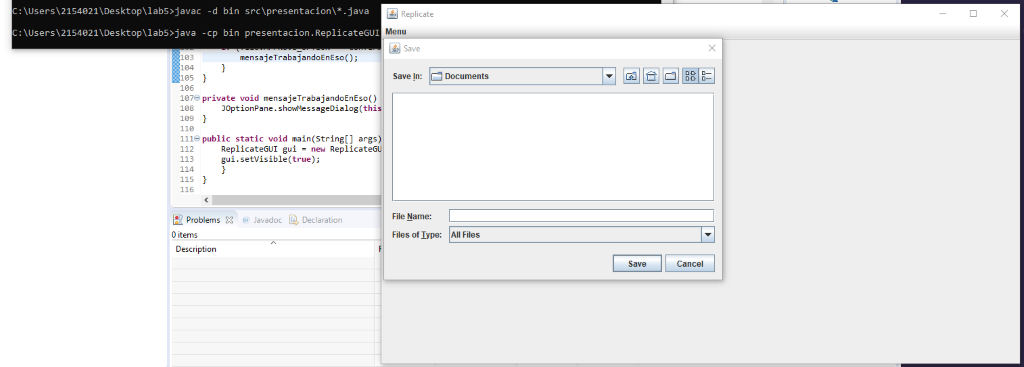
**showSaveDialog(Component parent) throws HeadlessException** Aparece un diálogo de selección de archivos "Guardar archivo". Tenga en cuenta que el texto que aparece en el botón de aprobación está determinado por la L&F

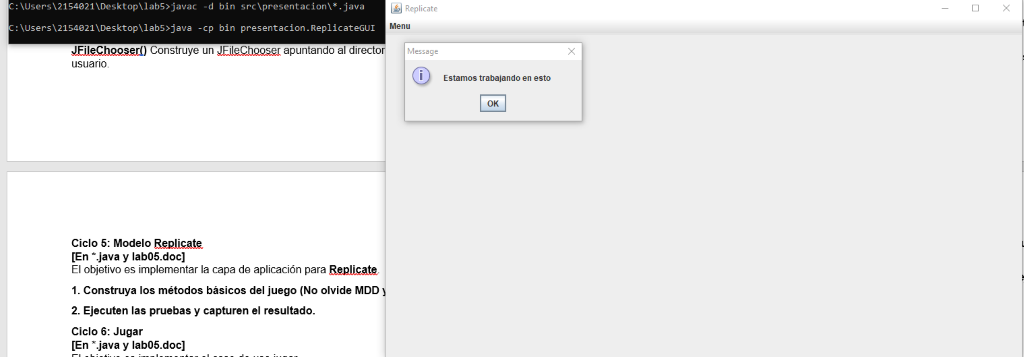
**getSelectedFile ()** Devuelve el archivo seleccionado. Esto lo puede configurar el programador a través de setSelectedFile o mediante una acción del usuario, como escribir el nombre de archivo en la interfaz de usuario o seleccionar el archivo de una lista en la UI.

**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.**

**3. Ejecuten las dos alternativas y capture las pantallas más significativas.**





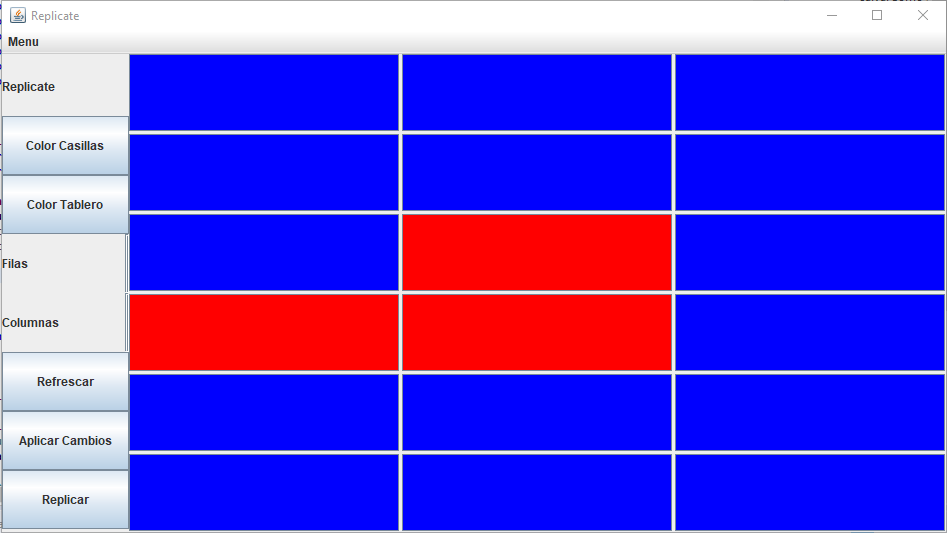


Para las tres opciones que tenemos funciona: salvar, salvar como y abrir, como se solicitan hasta el momento.

**Ciclo 3: Forma de la ventana principal**   
**[En \*.java y lab05.doc]**   
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 (azul/usuario – rojo/computador, 6 casas x 3 semillas) 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 necesarios para implementar este caso de uso.**

**Necesitamos de lo que consta JColorChooser**: Un método de conveniencia estático que muestra un diálogo modal de selección de color y devuelve el color seleccionado por el usuario, otro método de conveniencia estático para crear un diálogo de selección de color en el que se pueden especificar ActionListeners para que se invoquen cuando el usuario presiona uno de los botones de diálogo y la capacidad de crear instancias de paneles JColorChooser directamente (dentro de cualquier contenedor). También usar una variable temporal que se encargue de comparar si el color cambia, para que las demás celdas también cambien y no queden diversos colores a los dos seleccioandos.

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

**JColorChooser:** Este componente nos ayuda a seleccionar un color de una forma más atractiva para el usuario. JColorChooser proporciona un panel de controles diseñado para permitir que un usuario manipule y seleccione un color.

|  |  |
| --- | --- |
| static [**Color**](https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html) | [**showDialog**](https://docs.oracle.com/javase/7/docs/api/javax/swing/JColorChooser.html#showDialog(java.awt.Component,%20java.lang.String,%20java.awt.Color))**(**[**Component**](https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html) **component,** [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) **title,** [**Color**](https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html) **initialColor)**  Muestra un diálogo modal de selección de color y bloquea hasta que el  diálogo esté oculto. |

**3. Implementen los componentes necesarios para cambiar el color de las casas y almacenes.**

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

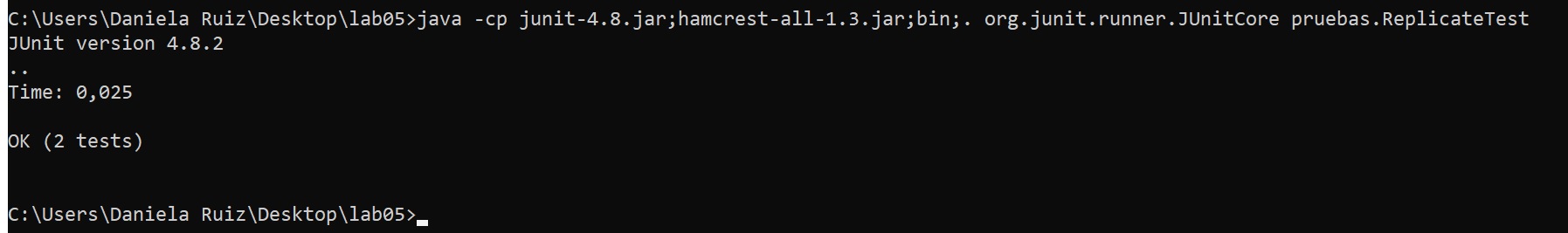


**Ciclo 5: Modelo Replicate**   
**[En \*.java y lab05.doc]**   
El objetivo es implementar la capa de aplicación para **Replicate**.

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

Pruebas: No debe permitir cambiar la configuración del tablero a menor que 0 y tiene que probarse que el replicar funcione de manera correcta en la aplicación.

**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.**

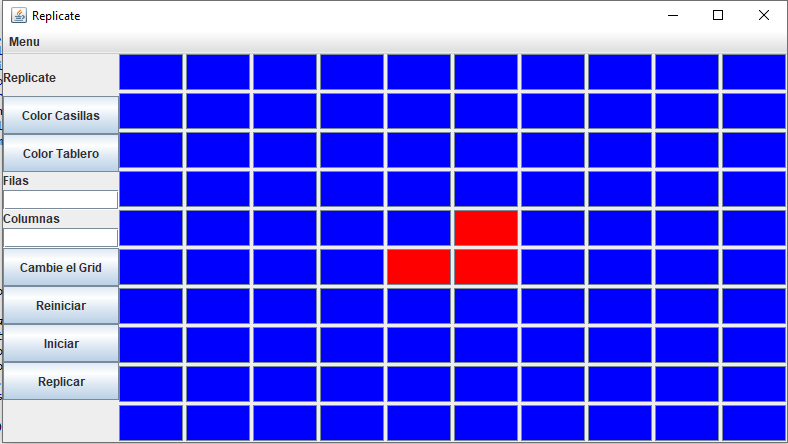
**2. Perfeccionen el método refresque() considerando la información del modelo de dominio.**

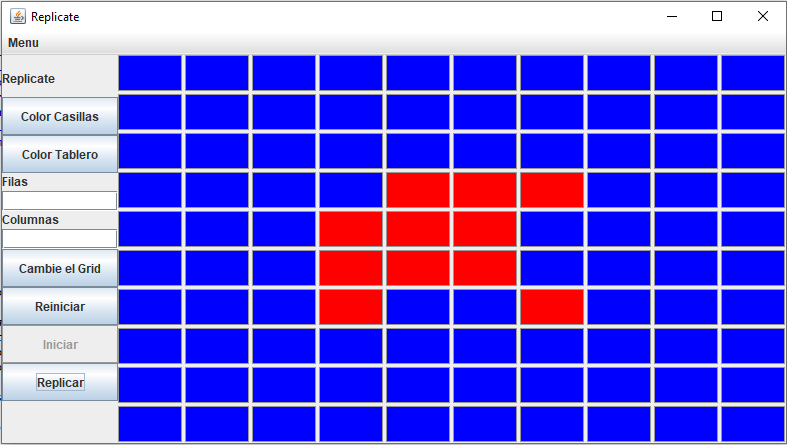
**3. Expliquen los elementos necesarios para implementar este caso de uso.**

Para jugar necesitamos dos matrices la inicial sin modificaciones, sobre esta el jugador selecciona las celdas a prender para jugar y usamos una segunda matriz que la actualiza. El iniciar solo se puede volver a activar si se usa el reiniciar, sino solo deja replicar.

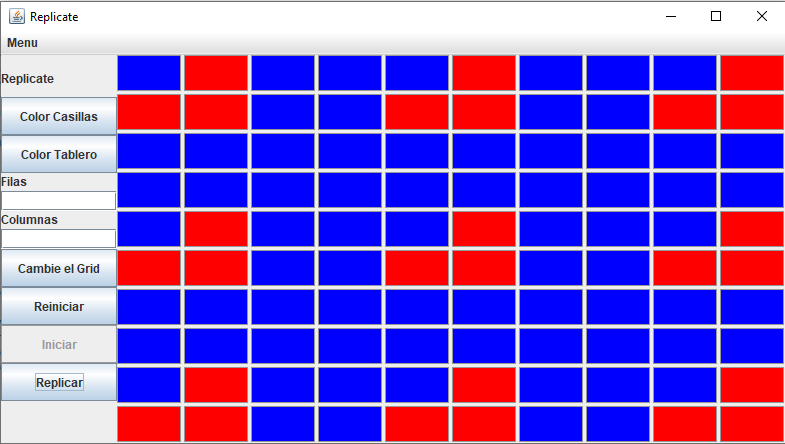
**4. Implementen los componentes necesarios para jugar.**

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





Luego de replicar 2 veces más:



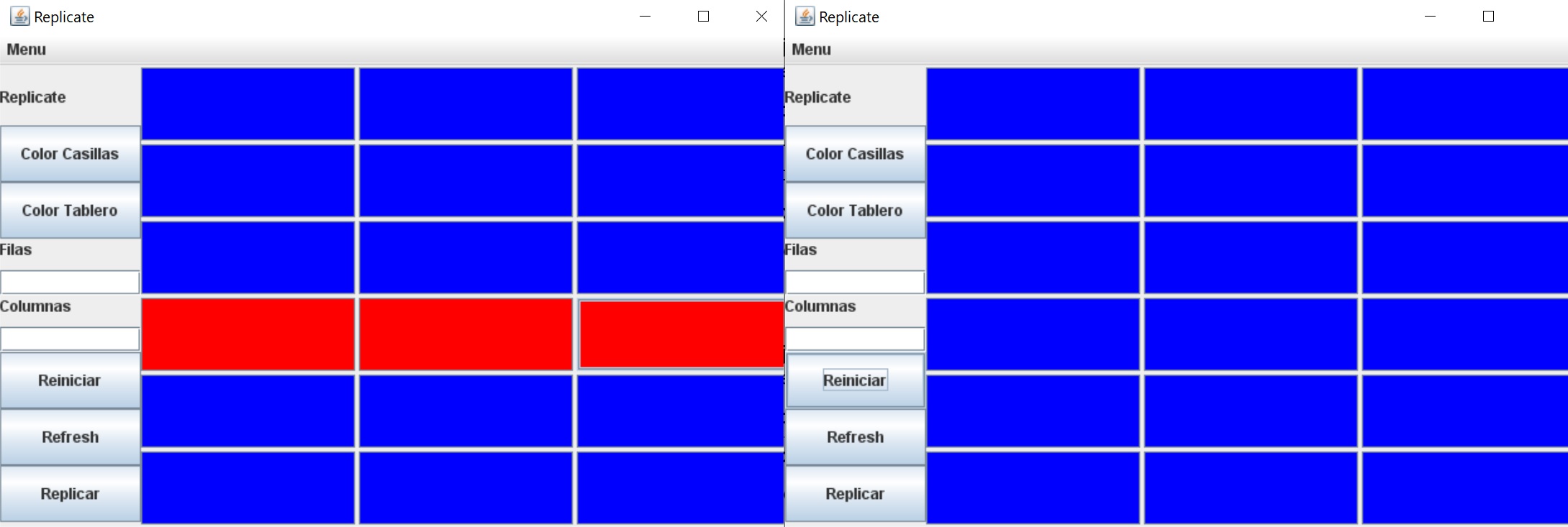
**Ciclo 7: Reiniciar**  
**[En \*.java y lab05.doc]**   
El objetivo es implementar este caso de uso.

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

Para reiniciar volveremos al estado default, usando un ActionListener para que cuando se le dé clic al botón reiniciar el tablero vuelva al estado inicial por defecto.

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

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



El botón Refresh no debía estar porque en sí era un método de respaldo para el botón replicar. Y pusimos el botón Replicar para el juego para que vaya actualizando la matriz, para poder jugar. Solo que en el pantallazo no alcanzó a quedar, pero en la versión que entregamos sí. El tablero default seleccionado por motivos de aplicación fue de 10x10 como en la guía del juego entregada.

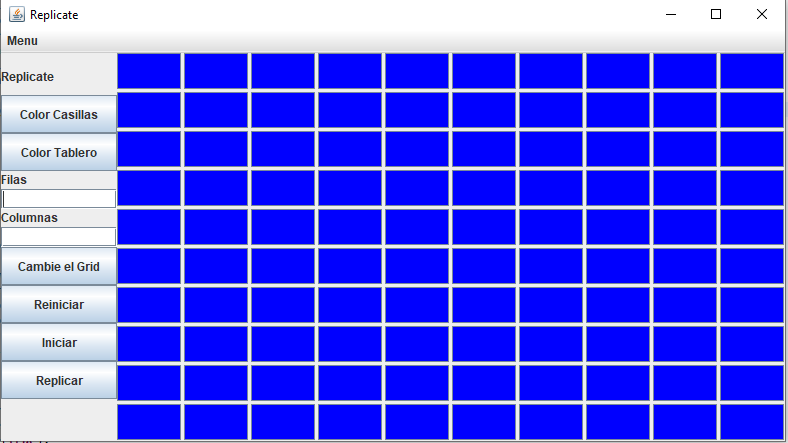
**Ciclo 8: Cambiar el tamaño del juego: casas y semillas**   
**[En \*.java y lab05.doc]**   
El objetivo es implementar este caso de uso.

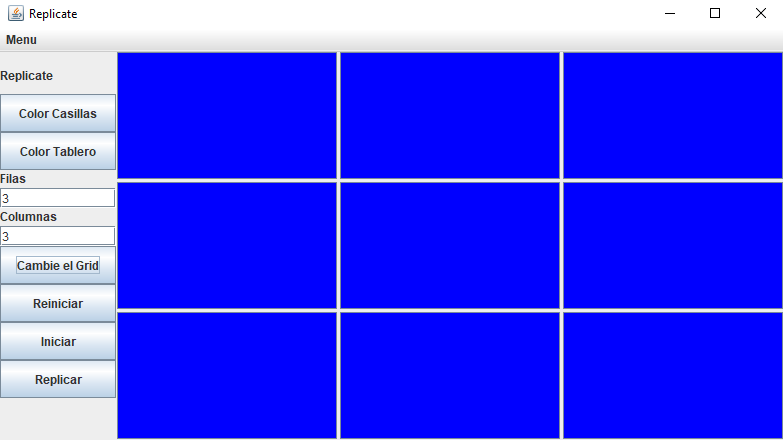
**1. Expliquen los elementos a usar para implementar este caso de uso**

Para cambiar el tamaño del juego le damos dos JText que reciben la información del tamaño a cambiar, e internamente se lo asignamos a la matriz para que se modifique.

**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?**

20 horas Ernesto y 20 horas Daniela.

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

Completo.

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

Las pruebas de aceptación se ejecutan con frecuencia, son de gran importancia para ir probando comportamientos.

Cuando se encuentra un error, se crean pruebas, esto es importante a la hora de probar la eficiencia del código

**4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?**

Aprender a usar todo lo nuevo de java.awt y java.swing, concordar en la idea del diseño y la implementación. Lo solucionamos investigando por cuenta propia más a fondo lo nuevo de GUI, probando y errando, además trabajando sobre el modelo y diseño con el que concordamos ambos.

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

Trabajar juntos en el laboratorio aprendiendo juntos, resolviendo dudas y evitando cometer tantos errores que se cometen cuando se trabaja cada uno un pedazo y luego se junta. Nos comprometemos a seguir dedicándole el espacio y tiempo necesario a los laboratorios para poder usar lo aprendido a la hora de desarrollar el proyecto final.