

# Informe de Laboratorio 22

## Tema: Interfaz Gráfica de Usuario

**Nota**

Estudiante	Escuela	Asignatura
Julio Rubén Chura Acabana jchuraaca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	F. de Programación 2 Semestre: I Código: 20230472

Laboratorio	Tema	Duración
22	Interfaz Gráfica de Usuario	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 10 de Enero 2023	Al 15 de Enero 2023

### 1. Tarea

- Cree una versión del videojuego de estrategia usando componentes básicos GUI: Etiquetas, botones, cuadros de texto, JOptionPane, Color.

### 2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- Visual Studio Code 1.85.1
- OpenJDK 64-Bits 20.0.2.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- Herencia y Polimorfismo

### 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/JulioChura/fp2-23b.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- <https://github.com/JulioChura/fp2-23b/tree/main/fase02/lab22>

## 4. Actividades con el repositorio GitHub

### 4.1. Preparación del espacio de trabajo

Listing 1: Se crea la carpeta de laboratorio 20 y se copian los archivos del lab012 al lab20

```
mkdir lab22
xcopy C:\Users\USUARIO\Desktop\fp2-23b\fase02\lab20
      C:\Users\USUARIO\Desktop\fp2-23b\fase02\lab22 /s /e
cd ..\lab22
code .
```

### 4.2. Evolución del código

Listing 2: Commit: d4ded5435b38f7e1e79f718e82fcf34e773e4dd9

```
git add Menu.java
git commit -m "Se genera un menu con netbeans"
git push -u origin main
```

- En este commit se decide crear un menu para el juego, se hace uso de netbeans, sin embargo, la idea se descarta y se decide que se usará para el trabajo final

Listing 3: Commit: 7108905874788aafe3a0774d8a976766b0b840fd

```
git add Opciones.java
git commit -m "4be9d3c230b3adf4550996cd2bb299064bfc27b5"
git push -u origin main
```

- En esta parte, se creó una ventana que tendrá las opciones para iniciar el juego, esta se encargaría de recoger el nombre de los ejércitos y la arena, solo faltaría implementar los listeners, sin embargo esta idea se descarta y se decide guardarla para el trabajo final, más adelante se mostrará la forma propuesta en la que se registrará el nombre de los ejércitos y la arena de juego.

Listing 4: Commit: 0d9b85edbec5ba2b5bc7e220d8d7db654ee05d02

```
code Tablero.java
git add Tablero.java
git commit -m "Se crea una clase Tablero para hacer test sobre como se imprimiria una
matiz de botones"
git push -u origin main
```

- En esta parte del trabajo, más antes se había copiado la clase Board del lab20 al lab22, sin embargo, más me terminó enredando por lo que decido eliminar varios métodos, además como primer avance, pretendo generar una matriz de botones.

Listing 5: Commit: e9567435759726fa3977d549e7b5e96619c935ae

```
code Tablero.java
git add Tablero.java
```

```
git commit -m "Se muestra el tablero con las posiciones de cada soldado que estan  
representados con caracteres"  
git push -u origin main
```

- Para hacer una prueba para poner en marcha una idea para asignar valores en los botones de acuerdo a la posición de cada soldado, se decide imprimir caracteres.

Listing 6: Commit: fde297f82a2db07850d0ebfb414669aee7afef58

```
code Tablero.java  
git add Tablero.java  
git commit -m "Encima de los botones se coloca las imagenes de los tipos de guerreos  
correspondiente"  
git push -u origin main
```

- Como funcionó la idea de los caracteres, ahora se decide que en lugar de caracteres se ponga íconos. Para ello se hace una serie de estructuras condicionales y el método `instanceOf` para determinar a qué tipo de `Soldier` pertenece el soldado de la actual iteración, por ello se recurre a un método de la clase `Army` el cual es `typeSoldier`, sin embargo, dicho método fue creado sin registrarlo en github puesto que se estaba trabajando con `Tablero` y no con la clase `Army`.

Listing 7: Commit: 238928df88e329446fe7864bed270d1889ed721f

```
code Tablero.java  
git add Tablero.java  
git commit -m "Se trata de realizar la logica de turnos, pero no dio resultados lo  
propuesto, se debe recurrir a otro metodo"  
git push -u origin main
```

- En este punto, se hicieron leves modificaciones a las demás clase que fueron copiadas del lab20 al 22, sin embargo, en este push recién se subirían, menciono ello para evitar confusiones. En cuanto a lo que realmente trata el commit, ya se implementa los `addListener`, pero para evitar que el código sea largo, se decide crearlo en otra clase, sin embargo, al probarlo, no salen los resultados esperados. Para este punto se hace una autoevaluación y se decide hacer el código de forma ordenada, ya que habían métodos que estaban en clases a las cuales no les correspondía de forma directa, además de que faltan métodos para la clase `Army`, por lo que se decide usar como referencia los métodos del lab12.

Listing 8: Commit: 205507f3ac3fa3180df2e8ffeee4bcf425f8e622

```
code Army.java  
git add Army.java  
git commit -m "Se crea el metodo searchSoldier el cual es estatico"  
git push -u origin main
```

- Esta es la primera versión del método, en su versión final se omite usar un ciclo `for` para recorrer el array, pero a pesar del cambio que hubo, la idea es que se pueda retornar un objeto `soldier` en una posición especificada y si no hay nada, retornar un `null`.

Listing 9: Commit: 3ee4852354ebaefb01e0ee7863610337747023cf

```
code Army.java
git add Army.java
git commit -m "Se crea un metodo estatico para poder validar si la posicion de destino
es la correcta"
git push -u origin main
```

- Este método, en su primera versión, se encarga de verificar si un soldado ubicado en dos posiciones es correcta, es decir, la de su origen y otra la de su futura ubicación no deben generar errores, es por esa razón que se creó el método `searchSoldier`.

Listing 10: Commit: d22eb15ed7ad2ea792ee5f1c28bfe93268a154d3

```
code Army.java
git add Army.java
git commit -m "Se crea el metodo moveSoldier, pero para un caso se necesita usar un
metodo que simule una batalla"
git push -u origin main
```

- Este método tiene dos casos, el primero es cuando el soldado seleccionado se moverá a una posición en la cual no hay soldados aliados ni enemigos, en estas circunstancias solo se actualizará la ubicación de este soldado dentro del `ArrayList` bidimensional. El segundo es cuando la posición de destino ya está ocupada por un soldado enemigo y así ahí donde se produciría una batalla, por lo cual requerimos de un método que simule una batalla.

Listing 11: Commit: 5c8804ab36be558c9b6c7450436559b8da1fe8da

```
code Army.java
git add Army.java
git commit -m "Se crea el metodo battle, pero para simplificar el metodo, se requiere
uno que determine al ganador"
git push -u origin main
```

- Este método a pesar de llevar el nombre de `battle`, no es el que realmente gestiona los combates, más que nada se usa para actualizar las nuevas posiciones de los soldados tras el combate

Listing 12: Commit: 2b03d7c1ed558e54ee31c51f78d89cfa4864b212

```
code Army.java
git add Army.java
git commit -m "Se crea el metodo winner"
git push -u origin main
```

- Este método es el que calcula al ganador en base a los puntos de vida y el uso de las probabilidades, según el resultado de la batalla este método muestra un cuadro de diálogo que muestra las estadísticas de la batalla y retorna valores que serán usados por el método `battle` para actualizar las nuevas posiciones.

Listing 13: Commit: 2f268b1e4b76f0adde845e61dc72a38a5b7107b3

```
code .
git add .
git commit -m "Se corrigen los indices de las columnas y filas"
git push -u origin main
```

- En esta parte se estaba probando el ActionListener pero no daba resultados. Por ello se decide hacer una corrección de las columnas y los filas porque supuse que el error podría ser eso, además de que para este punto he creado una clase test para ir probando los cambios.

Listing 14: Commit: 05b7ac04e84b96874123f30d29b229f4ea125ad6

```
code .
git add .
git commit -m "El metodo validatePosition fue simplificado, se hizo un test pero a
pesar de que las coordenadas que se mandan don correctas, las fichas no se mueven"
git push -u origin main
```

- Mas antes, las piezas se podían mover, pero cuando había cruce de posiciones el combate no se daba, además de que ocurrían inconsistencias en los movimientos.

Listing 15: Commit: 660f38ab7ff35142fcd6f634f4a7c4179a6d63bb

```
code Army.java
git add Army
git commit -m "Se soluciono el problema que daba cuando se trataba de obtener los
puntos de vida"
git push -u origin main
```

- Más antes, el problema por el cual las fichas no se movían era porque se había hecho una mala asignación de los valores en el parámetro, sin embargo se corrigió ello en la clase test, pero el problema era ahora de que al momento de obtener la vida, salía error de que no se podía aplicar el método getActualLife a null, por lo que se corrige este error ya que el método moveSoldier ya calculaba el valor de la vida sin que se haya verificado si el soldado era null o no .

Listing 16: Commit: 9220747824333dea82bfc24cd3401bdbbb246304

```
code Tablero.java
git add Tablero.java
git commit -m "Se establecen propiedades para el tablero y se pone la vida de los
soldados encima de los botones"
git push -u origin main
```

- Para que el juego sea más claro, se decide incluir un texto encima de las piezas que indica el valor de vida, para lograr el centrado perfecto del texto se hace uso de boton.setVerticalTextPosition(SwingConstants.CENTER) y boton.setHorizontalTextPosition(SwingConstants.CENTER) además de que se modifican los parámetros, ahora reciben dos objetos de tipo Army y el nombre de la arena, dichos valores se establecen en el título de la ventana.

Listing 17: Commit: 1aa785c91270143c1ff8fd449aa286693d84ed7c

```
code Army.java
git add Army.java
git commit -m "Se maneja la opcion de que la probabilidad definitiva salga igual a las
ya calculadas, se compila y sale un error en la actualizacion de las fichas"
git push -u origin main
```

- Hasta este punto, el juego mostraba inconsistencias en las batallas, por lo que el error radicaba en la clase winner, el cual no manejaba todos los casos y aquellos casos que no eran procesados retornaban el valor 3, además de que en el siguiente commit se detectó un error tipográfico cuando se actualizaban las posiciones en el método battle ya que los valores que se pasaban no estaban correctos, .

Listing 18: Commit: 08c58e08ef28150018984f2ac92a45b89c176de7

```
code Army.java
git add Army
git commit -m "Se corrige el movimiento, el problema radicaba en el metodo battle"
git push -u origin main
```

- En el paso anterior se mencionó sobre lo que causaba el error, que era un mal registro en los parámetros del método battle, así que se corrigen los valores .

Listing 19: Commit: 9b67006f46efdfb1cc92fbb419f399d028db2213

```
code .
git add .
git commit -m "Se borran lineas que no se usaran, es decir las de color amarillo,
ademas se corrigen ciertos detalles"
git push -u origin main
```

- Se borran líneas innecesarias (las que marcaban color amarillo en vs code). También se suben todos los cambios y se borran clase que no se usarán para el funcionamiento del presente código. Además de que la clase test.java pasa a llamarse Aplicación.java y recién se sube.

Listing 20: Commit: af25af9eba9738c387d5278b1dad3f96ea09b9dd

```
code .
git add .
git commit -m "Se crea una interfaz"
git push -u origin main
```

- Debido a que en laboratorios pasados que no fueron realizados se pedía el uso de interfaces mejor elaboradas, para este se decide incluirlo, pero de una manera más simple. Se incluye la interfaz SpecialUnit y como clase que la implementa es FrancoKnight que para el juego con con interfaz gráfica no se empleará

### 4.3. Métodos que generan el movimiento

```
201 public static Soldier searchSoldier(Army army, int row, int column) {
202     Soldier sol = army.getArmyInArrayListBi().get(row).get(column);
203     if (sol == null) {
204         return null;
205     } else {
206         return sol;
207     }
208 }
```

```
218 public static boolean validatePosition(Army ally, int rowOrigin, int columnOrigin,
219 int rowDestination, int columnDestination) {
220
221     if (searchSoldier(ally, rowOrigin, columnOrigin) != null
222         && searchSoldier(ally, rowDestination, columnDestination) == null) {
223         JOptionPane.showMessageDialog(parentComponent:null, message:"Coordenadas correctas!");
224         return false;
225     } else {
226         JOptionPane.showMessageDialog(parentComponent:null, message:"Coordenadas no válidas!");
227         return true;
228     }
229 }
```

```
233 public static void movesSoldier(Army destination, Army origin, int rowOrigin, int columnOrigin,
234 int rowDestination, int columnDestination) {
235     Soldier destinationSoldier = searchSoldier(destination, rowDestination, columnDestination);
236     Soldier originSoldier = searchSoldier(origin, rowOrigin, columnOrigin);
237
238     if (destinationSoldier == null) {
239         origin.getArmyInArrayListBi().get(rowOrigin).set(columnOrigin, element:null);
240         origin.getArmyInArrayListBi().get(rowDestination).set(columnDestination, originSoldier);
241     } else if (destinationSoldier != null) {
242         battle(destination, origin, rowOrigin, columnOrigin, rowDestination, columnDestination, destinationSoldier,
243             originSoldier);
244     }
245
246 }
```

```
248 public static void battle(Army destination, Army origin, int rowOrigin, int columnOrigin,
249 int rowDestination, int columnDestination, Soldier destinationSoldier, Soldier originSoldier) {
250     int lifeDestination = destinationSoldier.getActualLife();
251     int lifeOrigin = originSoldier.getActualLife();
252
253     int result = winner(lifeOrigin, lifeDestination);
254     if (1 == result) {
255         origin.getArmyInArrayListBi().get(rowOrigin).set(columnOrigin, element:null);
256         origin.getArmyInArrayListBi().get(rowDestination).set(columnDestination, originSoldier);
257         destination.getArmyInArrayListBi().get(rowDestination).set(columnDestination, element:null);
258     } else if (2 == result) {
259         origin.getArmyInArrayListBi().get(rowOrigin).set(columnOrigin, element:null);
260     } else {
261         origin.getArmyInArrayListBi().get(rowOrigin).set(columnOrigin, element:null);
262         destination.getArmyInArrayListBi().get(rowDestination).set(columnDestination, element:null);
263     }
264 }
```

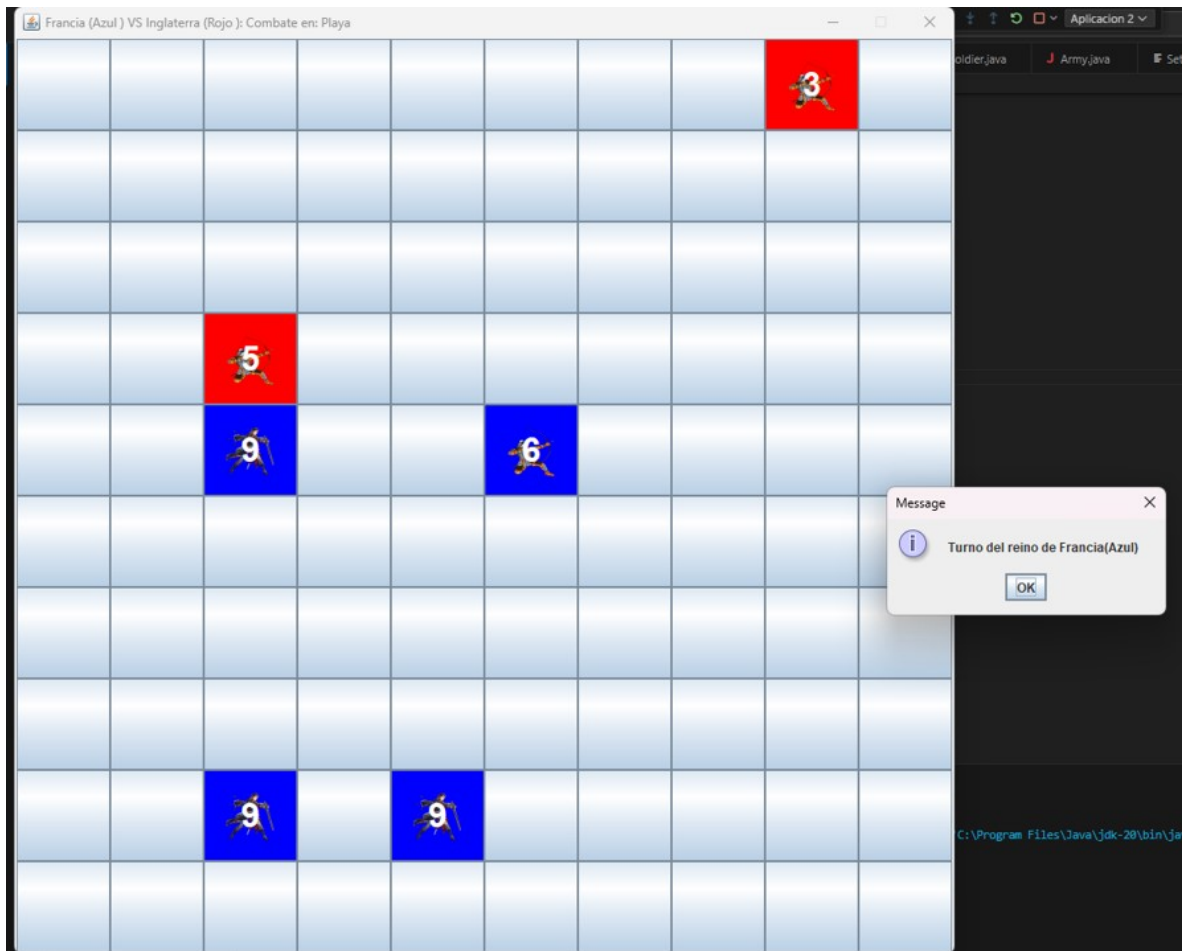
```

266 public static int winner(int life1, int life2) {
267     Random random = new Random();
268     int definitiveProbability = random.nextInt(bound:100);
269
270     double totallife = life1 + life2;
271     double probabilityAttack1 = 100 * life1 / totallife;
272     double probabilityAttack2 = 100 - probabilityAttack1;
273
274     JOptionPane.showMessageDialog(parentComponent:null, "Estadísticas de batalla:\tSoldado atacante: " + probabilityAttack1
275         + "%\t Soldado en reposo: " + probabilityAttack2 + "%\t Salio como aleatorio: "
276         + definitiveProbability + "%");
277
278     System.out.println();
279
280     if (definitiveProbability < probabilityAttack1) {
281         JOptionPane.showMessageDialog(parentComponent:null, message:"Gano el soldado atacante");
282         return 1;
283     } else if (definitiveProbability > probabilityAttack1) {
284         JOptionPane.showMessageDialog(parentComponent:null, message:"Gano el soldado en reposo");
285         return 2;
286     } else {
287         JOptionPane.showMessageDialog(parentComponent:null, message:"Empate");
288         return 3;
289     }
290 }
291
292
293 public static boolean winnerDefinitive(Army e1, Army e2) {
294     if (e2.converterToArrayUni().size() == 0) {
295         JOptionPane.showMessageDialog(parentComponent:null, "Ganó " + e2.getName());
296         return false;
297     }
298     if (e1.converterToArrayUni().size() == 0) {
299         JOptionPane.showMessageDialog(parentComponent:null, "Ganó " + e1.getName());
300         return false;
301     }
302     return true;
303 }
304
305 }

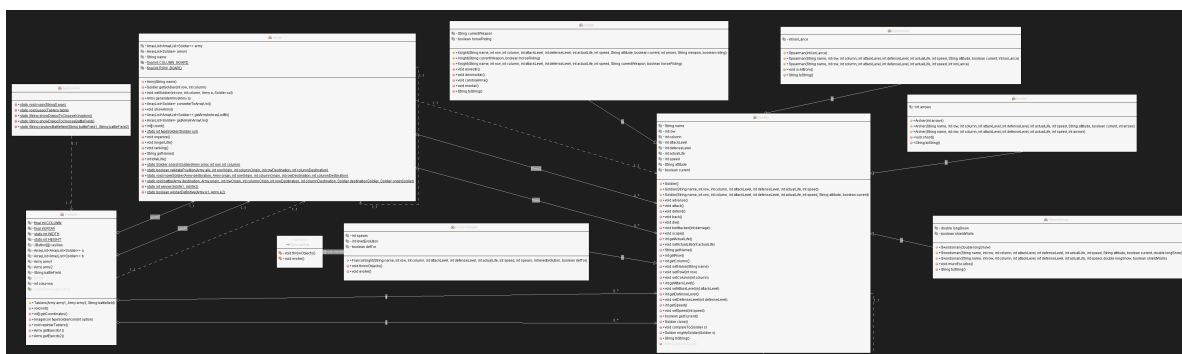
```



#### 4.4. Probando el juego



#### 4.5. Diagrama UML



- Si hay problemas en la visualización, puede ver la imagen en la carpeta img (descárgela)

#### 4.6. Estructura de laboratorio 22

- El contenido que se entrega en este laboratorio es el siguiente:

```
C:.\n|  Aplicacion.java\n|  Archer.java\n|  Army.java\n|  FrancoKnight.java\n|  Knight.java\n|  Soldier.java\n|  Spearman.java\n|  SpecialUnit.java\n|  Swordsman.java\n|  Tablero.java\n|\n|---img\n|    arquero.png\n|    caballero.png\n|    espadachin.png\n|    lancero.png\n|\n|---latex\n|    programacion_lab22_rescobedoq_v1.0.pdf\n|    programacion_lab22_rescobedoq_v1.0.tex\n|\n|---img\n|    battle.png\n|    knight.png\n|    logo_abet.png\n|    logo_episunsa.png\n|    move.jpg\n|    search.jpg\n|    test.jpg\n|    uml.png\n|    validate.jpg\n|    winner.png
```

- Debido a problemas en la codificación de latex con ciertos caracteres que genera el comando tree f, se opta por enviar la estructura de este laboratorio en formato de imagen

## 5. Rúbricas

### 5.1. Entregable Informe

Tabla 1: Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

## 5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
<b>Puntos</b>	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
<b>2. Commits</b>	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
<b>3. Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
<b>4. Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
<b>5. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
<b>6. Fechas</b>	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos.	2	X	2	
<b>8. Madurez</b>	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	2	
<b>Total</b>		20		18	

## 6. Referencias

- <https://www.geeksforgeeks.org/insertion-sort/>