

# Informe de Laboratorio 24

## Tema: Proyecto final-lab24

Nota

Estudiante	Escuela	Asignatura
Ryan Fabian Valdivia Segovia, Dylan Antonio Zuñiga Huarca rvaldiviase@unsa.edu.pe, dzunigahu@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
24	Proyecto final-lab24	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 24 de Enero 2024	Al 29 de Enero 2024

### 1. Tarea

- Cree una versión del videojuego de estrategia usando componentes básicos GUI: Etiquetas, botones, cuadros de texto, JOptionPane, Color.
- Además, utilizar componentes avanzados GUI: Layouts, JPanel, áreas de texto, checkbox, botones de radio y combobox.
- Considerar nivel estratégico y táctico.
- Considerar hasta las unidades especiales de los reinos.
- Hacerlo iterativo.

### 2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Home Single Language 64 bits 22H2.2283
- VIM 9.0.
- Visual Studio Code 64 bits 1.82.2
- OpenJDK 64-Bits 21.0.2
- Git 2.41.0.windows.1
- IntelliJ IDEA 2023.3 Runtime version: 17.0.9+7-b1087.7 amd64
- Cuenta en GitHub con el correo institucional.

### 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/RyanValdivia/fp2-23b>
- URL para el laboratorio 24 en el Repositorio GitHub.
- <https://github.com/RyanValdivia/fp2-23b/tree/main/fase03/lab22>

### 4. Actividades

- Este laboratorio consistía en elaborar una interfaz gráfica de usuario, portando todo nuestro videojuego de laboratorios anteriores.
- Para esto, reutilicé el código de laboratorios anteriores, usando la misma lógica. Entonces reciclé las clases de laboratorios anteriores

Listing 1: Superclase Soldado

```
package com.example.lab22;
import java.util.*;

public class Soldier {
    protected String name;
    protected String alias;
    protected int HP;
    protected int attack;
    protected int defense;
    protected int x;
    protected int y;
    protected boolean status;
    private String color;
    private int armyId;

    public Soldier(String name){
        this.name = name;
        this.status = true;
    }
    public Soldier(){
        this.name = "";
        this.alias = "";
        this.color = "#cdcdcd";
        this.status = false;
    }
    public void action(){};
    public static Soldier winner(Soldier s1, Soldier s2){
        Random random = new Random();
        double h1 = s1.getHP();
        double h2 = s2.getHP();
        double total = h1 + h2;
        h1 /= total;
        h2 /= total;
        double ans = random.nextDouble();
        if(h2 < h1){
            if(ans <= h2){
```

```
        return s2;
    }else{
        return s1;
    }
}
}else{
    if(ans <= h1){
        return s1;
    }else{
        return s2;
    }
}
}

@Override
public String toString(){
    return name + ", " + alias + ", " + HP + ", " + attack + ", " + defense + ", " +
        armyId + ", " + x + ", " + y + ", " + status + ", " + color;
}

public static Soldier parseSoldier(String data){
    String[] parts = data.split(", ");
    String name = parts[0];
    String alias = parts[1];
    int hp = Integer.parseInt(parts[2]);
    int attack = Integer.parseInt(parts[3]);
    int defense = Integer.parseInt(parts[4]);
    int armyId = Integer.parseInt(parts[5]);
    int x = Integer.parseInt(parts[6]);
    int y = Integer.parseInt(parts[7]);
    boolean status = Boolean.parseBoolean(parts[8]);
    String color = parts[9];

    Soldier soldier = new Soldier(name);
    soldier.setX(x);
    soldier.setY(y);
    soldier.setAlias(alias);
    soldier.setHP(hp);
    soldier.setAttack(attack);
    soldier.setDefense(defense);
    soldier.setColor(color);
    soldier.setArmyId(armyId);
    soldier.setStatus(status);

    return soldier;
}

public void destroy(){
    this.name = "";
    this.alias = "";
    this.HP = 0;
    this.attack = 0;
    this.defense = 0;
    this.status = false;
    this.color = "#cdcdcd";
    this.armyId = 0;
}
```

```
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
public String getAlias() {  
    return alias;  
}  
  
public void setAlias(String alias) {  
    this.alias = alias;  
}  
  
public int getHP() {  
    return HP;  
}  
  
public void setHP(int HP) {  
    this.HP = HP;  
}  
  
public int getAttack() {  
    return attack;  
}  
  
public void setAttack(int attack) {  
    this.attack = attack;  
}  
  
public int getDefense() {  
    return defense;  
}  
  
public void setDefense(int defense) {  
    this.defense = defense;  
}  
  
public int getX() {  
    return x;  
}  
  
public String getColor() {  
    return color;  
}  
  
public void setColor(String color) {  
    this.color = color;  
}  
  
public int getArmyId() {  
    return armyId;  
}
```

```
    public void setArmyId(int armyId) {
        this.armyId = armyId;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public boolean getStatus() {
        return status;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }
}

class Archer extends Soldier{
    private int arrows;
    Random random = new Random();

    public Archer(String name) {
        super(name);
        this.arrows = random.nextInt(10);
        this.setAttack(7);
        this.setDefense(3);
        this.setHP(random.nextInt(3) + 3);
    }

    public int getArrows() {
        return arrows;
    }

    public void setArrows(int arrows) {
        this.arrows = arrows;
    }

    //Shoots one arrow to an enemy
    @Override
    public void action(){
        this.arrows--;
    }
}

class Knight extends Soldier{
    private String weapon;
    private boolean isMounted;
    Random random = new Random();

    public Knight(String name) {
```

```
        super(name);
        this.setAttack(13);
        this.setDefense(7);
        this.setHP(random.nextInt(3) + 10);
    }

    public String getWeapon() {
        return weapon;
    }

    public void setWeapon(String weapon) {
        this.weapon = weapon;
    }

    public boolean isMounted() {
        return isMounted;
    }

    public void setMounted(boolean mounted) {
        isMounted = mounted;
    }

    //If it's unmounted, it will mount and its weapon will change to a Sword, if not,
    // it will unmount and its weapon will change to a Spear
    @Override
    public void action(){
        this.isMounted = !this.isMounted;

        if(this.isMounted){
            this.weapon = "Sword";
        }else{
            this.weapon = "Spear";
        }
    }
}

class Swordsman extends Soldier{
    private final int swordLong;
    Random random = new Random();

    public Swordsman(String name) {
        super(name);
        this.swordLong = random.nextInt(10);
        this.setAttack(10);
        this.setDefense(8);
        this.setHP(random.nextInt(3) + 8);
    }

    public int getSwordLong() {
        return swordLong;
    }

    //This will set a wall of shields, and increase its defense by 1
    @Override
    public void action(){
        this.setDefense(this.getDefense() + 1);
    }
}
```

```
}  
class Spearman extends Soldier{  
    private final int spearLong;  
    Random random = new Random();  
  
    public Spearman(String name) {  
        super(name);  
        this.spearLong = random.nextInt(10);  
        this.setAttack(5);  
        this.setDefense(8);  
        this.setHP(random.nextInt(3) + 8);  
    }  
  
    public int getSpearLong() {  
        return spearLong;  
    }  
  
    //This will make a schiltrom and will increase its defense by 1  
    @Override  
    public void action(){  
        this.setDefense(this.getDefense() + 1);  
    }  
}
```

Listing 2: Clase Board

```
284 package com.example.lab22;  
285 import java.util.*;  
286  
287 public class Board {  
288     private Soldier[][] table = new Soldier[10][10];  
289  
290     private String[] territories = new String[] {  
291         "Bosque", "Campo Abierto", "Montana", "Desierto", "Playa" };  
292     private String territory;  
293     private Map<String, String> perks = new HashMap<>();  
294     Random random = new Random();  
295     public Board() {  
296         this.territory = territories[random.nextInt(territories.length)];  
297     }  
298     public void initializeTable(){  
299         for(int i = 0; i < table.length; i++){  
300             for(int j = 0; j < table[i].length; j++){  
301                 table[i][j] = new Soldier();  
302             }  
303         }  
304     }  
305     public void initializeArmy(Army a){  
306         for(Soldier s: a.getSoldiers()){  
307             int x = 0;  
308             int y = 0;  
309             do{  
310                 x = random.nextInt(9);  
311                 y = random.nextInt(9);  
312             }while(table[y][x].getStatus());  
313             s.setY(y);
```

```

314         s.setX(x);
315         table[y][x] = s;
316     }
317 }
318 public void deployArmy(Army a){
319     for(Soldier s: a.getSoldiers()){
320         int x = s.getX();
321         int y = s.getY();
322         table[y][x] = s;
323     }
324 }
325
326 public Soldier[][] getTable() {
327     return table;
328 }
329
330 public void setTable(Soldier[][] table) {
331     this.table = table;
332 }
333
334 public String[] getTerritories() {
335     return territories;
336 }
337
338 public void setTerritories(String[] territories) {
339     this.territories = territories;
340 }
341
342 public String getTerritory() {
343     return territory;
344 }
345
346 public void setTerritory(String territory) {
347     this.territory = territory;
348 }
349
350 public Map<String, String> getPerks() {
351     return perks;
352 }
353
354 public void setPerks(Map<String, String> perks) {
355     this.perks = perks;
356 }
357 }

```

- Esta clase implementa la interfaz gráfica en base a un fichero fxmL, en el cual desarrollo lo que se va a mostrar al usuario.
- Para esta interfaz, decidí implementar 100 botones que representen un tablero de 10x10 usando el contenedor GridPane, para poder llevar una cuenta de las coordenadas de cada botón, además de una barra arriba del tablero para cambiar de turno, minimizar, cerrar o maximizar.

Listing 3: Archivo FXML

```

361 <?xml version="1.0" encoding="UTF-8"?>

```



```

362
363 <?import javafx.geometry.*?>
364 <?import javafx.scene.control.*?>
365 <?import javafx.scene.layout.*?>
366
367 <VBox fx:id="rootVBox" prefHeight="400.0" prefWidth="666.0" spacing="20.0" styleClass="vbox"
    stylesheets="@css/game.css" xmlns="http://javafx.com/javafx/17.0.2-ea"
    xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="com.example.lab22.VideoGameController">
368     <padding>
369         <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
370     </padding>
371     <children>
372         <Pane prefHeight="200.0" prefWidth="200.0" styleClass="toolBar">
373             <children>
374                 <Pane fx:id="closeButton" layoutX="580.0" layoutY="-12.0"
                    onMouseClicked="#handleCloseButton" />
375                 <Pane fx:id="minimizeButton" layoutX="520.0" layoutY="-12.0"
                    onMouseClicked="#handleMinimizeButton" />
376                 <Pane fx:id="expandButton" layoutX="550.0" layoutY="-12.0"
                    onMouseClicked="#handleExpandButton" />
377                 <Pane fx:id="nextButton" layoutX="15.0" layoutY="-12.0"
                    onMouseClicked="#changeTurn" />
378                 <Pane fx:id="saveButton" layoutX="45.0" layoutY="-12.0"
                    onMouseClicked="#saveGame" />
379                 <Pane fx:id="loadButton" layoutX="75.0" layoutY="-12.0"
                    onMouseClicked="#loadGame" />
380                 <Pane fx:id="infoButton" layoutX="105.0" layoutY="-12.0"
                    onMouseClicked="#showRulesDialog" />
381             </children>
382         </Pane>
383
384
385         <GridPane id="board" fx:id="gridPane" minWidth="0.0" prefHeight="330.0"
            prefWidth="352.0">
386             <columnConstraints>
387                 <ColumnConstraints percentWidth="10" />
388                 <ColumnConstraints percentWidth="10" />
389                 <ColumnConstraints percentWidth="10" />
390                 <ColumnConstraints percentWidth="10" />
391                 <ColumnConstraints percentWidth="10" />
392                 <ColumnConstraints percentWidth="10" />
393                 <ColumnConstraints percentWidth="10" />
394                 <ColumnConstraints percentWidth="10" />
395                 <ColumnConstraints percentWidth="10" />
396                 <ColumnConstraints percentWidth="10" />
397             </columnConstraints>
398             <rowConstraints>
399                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
400                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
401                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
402                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
403                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
404                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
405                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
406                 <RowConstraints percentHeight="10" vgrow="SOMETIMES" />

```

```
407         <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
408         <RowConstraints percentHeight="10" vgrow="SOMETIMES" />
409     </rowConstraints>
410     <children>
411
412         <!-- Row 0 -->
413         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
414             styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="0" />
415         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
416             styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="0" />
417         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
418             styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="0" />
419         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
420             styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="0" />
421         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
422             styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="0" />
423         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
424             styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="0" />
425         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
426             styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="0" />
427         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
428             styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="0" />
429         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
430             styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="0" />
431         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
432             styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="0" />
433
434         <!-- Row 1 -->
435         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
436             styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="1" />
437         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
438             styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="1" />
439         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
440             styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="1" />
441         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
442             styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="1" />
443         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
444             styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="1" />
445         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
446             styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="1" />
447         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
448             styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="1" />
449         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
450             styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="1" />
451         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
452             styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="1" />
453         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
454             styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="1" />
455
456         <!-- Row 2 -->
457         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
458             styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="2" />
459         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
460             styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="2" />
461         <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
462             styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="2" />
```

```
440 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="2" />
441 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="2" />
442 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="2" />
443 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="2" />
444 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="2" />
445 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="2" />
446 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="2" />
447
448 <!-- Row 3 -->
449 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="3" />
450 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="3" />
451 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="3" />
452 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="3" />
453 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="3" />
454 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="3" />
455 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="3" />
456 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="3" />
457 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="3" />
458 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="3" />
459
460 <!-- Row 4 -->
461 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="4" />
462 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="4" />
463 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="4" />
464 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="4" />
465 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="4" />
466 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="4" />
467 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="4" />
468 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="4" />
469 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
      styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="4" />
```

```

470     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="4" />
471
472     <!-- Row 5 -->
473     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="5" />
474     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="5" />
475     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="5" />
476     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="5" />
477     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="5" />
478     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="5" />
479     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="5" />
480     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="5" />
481     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="5" />
482     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="5" />
483
484     <!-- Row 6 -->
485     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="6" />
486     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="6" />
487     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="6" />
488     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="6" />
489     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="6" />
490     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="6" />
491     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="6" />
492     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="6" />
493     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="6" />
494     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="6" />
495
496     <!-- Row 7 -->
497     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="7" />
498     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="7" />
499     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="7" />
500     <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
         styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="7" />

```



```
501 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="7" />
502 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="7" />
503 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="7" />
504 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="7" />
505 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="7" />
506 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="7" />
507
508 <!-- Row 8 -->
509 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="8" />
510 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="8" />
511 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="8" />
512 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="8" />
513 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="8" />
514 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="8" />
515 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="8" />
516 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="8" />
517 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="8" />
518 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="8" />
519
520 <!-- Row 9 -->
521 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="0" GridPane.rowIndex="9" />
522 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="1" GridPane.rowIndex="9" />
523 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="2" GridPane.rowIndex="9" />
524 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="3" GridPane.rowIndex="9" />
525 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="4" GridPane.rowIndex="9" />
526 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="5" GridPane.rowIndex="9" />
527 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="6" GridPane.rowIndex="9" />
528 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="7" GridPane.rowIndex="9" />
529 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="8" GridPane.rowIndex="9" />
530 <Button mnemonicParsing="false" onMouseClicked="#onButtonClick"
    styleClass="button" text="" GridPane.columnIndex="9" GridPane.rowIndex="9" />
```

```
531
532
533     </children>
534   </GridPane>
535 </children>
536 </VBox>
```

- Esto con el fin de asignar una función llamada `onButtonClick`, que implementaré mas adelante, a cada botón, para
- Por último, programé el resumen final, donde se dice cuál de los dos ejércitos es el ganador, basado en probabilidad según la cantidad de vida total por ejército. Eso es todo. Al final, solo me quedaba armar todo en la clase principal, sin embargo, se iba a ver algo feo, así que apliqué una hoja de estilos css para que se vea mucho más agradable a la vista.

Listing 4: Game.css

```
538 /* Estilos para el VBox (contenedor principal) */
539 .vbox {
540     -fx-spacing: 20px;
541     -fx-background-color: #000;
542     -fx-alignment: center;
543     -fx-halignment: center;
544 }
545 .toolBar{
546     -fx-max-width: 626px;
547 }
548 /* Estilos para el GridPane */
549 #board {
550     -fx-max-width: 626px;
551     -fx-max-height: 624px;
552     -fx-padding: 10px;
553     -fx-border-color: #000;
554     -fx-border-width: 2px;
555 }
556 }
557
558 /* Estilos para los botones */
559 .button {
560     -fx-min-width: 60px;
561     -fx-min-height: 60px;
562     -fx-max-width: 60px;
563     -fx-max-height: 60px;
564     -fx-background-color: #fff; /* Color de fondo blanco por defecto */
565     -fx-font-size: 14px;
566     -fx-font-weight: bold;
567     -fx-border-color: #000;
568     -fx-border-width: 2px;
569 }
570 #closeButton{
571     -fx-min-width: 30px;
572     -fx-min-height: 30px;
573     -fx-background-color: transparent;
574     -fx-background-image: url('../img/x-circle-regular-24.png');
575     -fx-background-repeat: no-repeat;
```

```
576     -fx-background-size: cover;
577 }
578 #minimizeButton{
579     -fx-min-width: 30px;
580     -fx-min-height: 30px;
581     -fx-background-color: transparent;
582     -fx-background-image: url('../img/minus-regular-24.png');
583     -fx-background-repeat: no-repeat;
584     -fx-background-size: cover;
585 }
586 #expandButton{
587     -fx-min-width: 30px;
588     -fx-min-height: 30px;
589     -fx-background-color: transparent;
590     -fx-background-image: url('../img/expand-alt-regular-24.png');
591     -fx-background-repeat: no-repeat;
592     -fx-background-size: cover;
593 }
594 #nextButton{
595     -fx-min-width: 30px;
596     -fx-min-height: 30px;
597     -fx-background-color: transparent;
598     -fx-background-image: url('../img/skip-next-circle-regular-24.png');
599     -fx-background-repeat: no-repeat;
600     -fx-background-size: cover;
601 }
602 #loadButton{
603     -fx-min-width: 30px;
604     -fx-min-height: 30px;
605     -fx-background-color: transparent;
606     -fx-background-image: url('../img/folder-open-regular-24.png');
607     -fx-background-repeat: no-repeat;
608     -fx-background-size: cover;
609 }
610 #saveButton{
611     -fx-min-width: 30px;
612     -fx-min-height: 30px;
613     -fx-background-color: transparent;
614     -fx-background-image: url('../img/save-regular-24.png');
615     -fx-background-repeat: no-repeat;
616     -fx-background-size: cover;
617 }
618 #infoButton{
619     -fx-min-width: 30px;
620     -fx-min-height: 30px;
621     -fx-background-color: transparent;
622     -fx-background-image: url('../img/info-circle-regular-24.png');
623     -fx-background-repeat: no-repeat;
624     -fx-background-size: cover;
625 }
```

- Con esto, solo me falta implementar la clase principal controladora para manejar el videojuego.

Listing 5: Clase controladora

```
629 package com.example.lab22;
630
631 import javafx.animation.KeyFrame;
632 import javafx.animation.Timeline;
633 import javafx.application.Platform;
634 import javafx.fxml.FXML;
635 import javafx.fxml.FXMLLoader;
636 import javafx.scene.Parent;
637 import javafx.scene.Scene;
638 import javafx.scene.control.*;
639 import javafx.scene.input.KeyCode;
640 import javafx.scene.input.MouseEvent;
641 import javafx.scene.layout.GridPane;
642 import javafx.scene.layout.VBox;
643 import javafx.stage.FileChooser;
644 import javafx.stage.Modality;
645 import javafx.stage.Stage;
646 import javafx.util.Duration;
647
648 import java.io.*;
649 import java.util.concurrent.atomic.AtomicBoolean;
650
651 public class VideoGameController {
652     @FXML
653     private GridPane gridPane = new GridPane();
654
655     private Army a1 = new Army(1);
656     private Army a2 = new Army(2);
657     private Board map = new Board();
658     private static int actualTurn = 1;
659     private int nEj1 = a1.getSoldiers().size();
660     private int nEj2 = a2.getSoldiers().size();
661     private boolean gameOver = false;
662
663
664     @FXML
665     private VBox rootVBox;
666
667
668     @FXML
669     public void initialize() {
670         initializeBoard();
671         runGame();
672     }
673
674     public void runGame() {
675         final boolean[] isAlertShown = {false};
676         Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1), event -> {
677             if (!gameOver) {
678                 gameOver();
679                 placeSoldiers();
680             } else if (!isAlertShown[0]){
681                 Platform.runLater(() -> {
682                     Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
683                     alert.setTitle("Juego Terminado");
684                     alert.setHeaderText(null);
```



```
685         alert.setContentText("El ganador es el Ejercito " + (nEj1 == 0 ? 2 : 1));
686         alert.showAndWait();
687         // Cerrar la aplicacion despues de mostrar el mensaje de juego terminado
688         Platform.exit();
689     });
690     isAlertShown[0] = true;
691 }
692 }));
693 timeline.setCycleCount(Timeline.INDEFINITE);
694 timeline.play();
695
696 }
697 private void initializeBoard(){
698     map.initializeTable();
699     map.initializeArmy(a1);
700     map.initializeArmy(a2);
701     nEj1 = a1.getSoldiers().size();
702     nEj2 = a2.getSoldiers().size();
703 }
704
705 private void placeSoldiers() {
706
707     int rowCount = 1;
708     int colCount = 1;
709
710     for (javafx.scene.Node node : gridPane.getChildren()) {
711         Soldier s = map.getTable()[rowCount - 1][colCount - 1];
712         String color = s.getColor();
713         String n = s.getAlias();
714         if (node instanceof Button) {
715             Button button = (Button) node;
716             String buttonName = n;
717             button.setText(buttonName);
718             button.setStyle("-fx-background-color: " + color);
719             colCount++;
720
721             if (colCount > 10) {
722                 colCount = 1;
723                 rowCount++;
724             }
725         }
726     }
727 }
728 public void onButtonClick(MouseEvent event){
729     if (event.getSource() instanceof Button) {
730         Button clickedButton = (Button) event.getSource();
731         int columnIndex = GridPane.getColumnIndex(clickedButton);
732         int rowIndex = GridPane.getRowIndex(clickedButton);
733
734         Soldier s = map.getTable()[rowIndex][columnIndex];
735
736         if(s.getStatus()){
737             if(s.getArmyId() == actualTurn){
738                 getDirection(s);
739             }else{
740                 showAlert("Elige un soldado de tu propio Ejercito", "Error");
741             }
742         }
743     }
744 }
```

```
741     }
742     }
743
744     }
745 }
746 private static void showAlert(String contentText, String title) {
747     Alert alert = new Alert(Alert.AlertType.WARNING);
748     alert.setTitle(title);
749     alert.setHeaderText(null);
750     alert.setContentText(contentText);
751     alert.showAndWait();
752 }
753 private void battle(Soldier s1, Soldier s2){
754     Soldier winner = Soldier.winner(s1, s2);
755
756
757     if(winner.getArmyId() == 1){
758         nEj2--;
759     }else{
760         nEj1--;
761     }
762
763     Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
764     alert.setTitle("Batalla");
765     alert.setHeaderText(null);
766     alert.setContentText("Los dos soldados batallaron, el ganador fue: " +
767         winner.getName());
768
769     alert.showAndWait();
770
771     int x1 = s1.getX();
772     int y1 = s1.getY();
773     int x2 = s2.getX();
774     int y2 = s2.getY();
775     int x = winner.getX();
776     int y = winner.getY();
777
778     if(x1 == x && y1 == y){
779         Soldier temp = map.getTable()[y2][x2];
780         map.getTable()[y1][x1] = temp;
781         map.getTable()[y2][x2] = winner;
782         map.getTable()[y1][x1].destroy();
783     }else{
784         map.getTable()[y1][x1].destroy();
785     }
786 }
787 private void move(Soldier s, int x, int y){
788     Soldier s2 = map.getTable()[y][x];
789     int oldX = s.getX();
790     int oldY = s.getY();
791     if(s2.getStatus()){
792         if(s2.getArmyId() == s.getArmyId()){
793             showAlert("No se permite el fuego aliado", "Error");
794         }else{
795             battle(s, s2);
796             actualTurn = (actualTurn == 1) ? 2 : 1;
```

```
796     }
797   }else{
798     s.setX(x);
799     s.setY(y);
800     s2.setX(oldX);
801     s2.setY(oldY);
802
803     Soldier temp = map.getTable()[y][x];
804     map.getTable()[y][x] = s;
805     map.getTable()[oldY][oldX] = temp;
806     actualTurn = (actualTurn == 1) ? 2 : 1;
807
808   }
809 }
810 public void gameOver(){
811     if (nEj1 == 0 || nEj2 == 0) {
812         gameOver = true;
813     }
814 }
815 public void changeTurn(MouseEvent event) {
816     Alert alerta = new Alert(Alert.AlertType.CONFIRMATION);
817     alerta.setTitle("Confirmacion");
818     alerta.setHeaderText("Quieres terminar tu turno?");
819     alerta.setContentText("Por favor, elige una opcion.");
820
821     // Personalizar los botones de la alerta
822     ButtonType botonSi = new ButtonType("Si");
823     ButtonType botonNo = new ButtonType("No");
824     alerta.getButtonTypes().setAll(botonSi, botonNo);
825
826     // Mostrar la alerta y esperar a que el usuario elija una opcion
827     alerta.showAndWait().ifPresent(resultado -> {
828         if (resultado == botonSi) {
829             actualTurn = (actualTurn == 1) ? 2 : 1;
830         }
831     });
832
833 }
834
835 }
836
837 // Metodo para cerrar la ventana
838 @FXML
839 private void handleCloseButton(MouseEvent event) {
840     Stage stage = (Stage) rootVBox.getScene().getWindow();
841     stage.close();
842 }
843
844 // Metodo para minimizar la ventana
845 @FXML
846 private void handleMinimizeButton(MouseEvent event) {
847     Stage stage = (Stage) rootVBox.getScene().getWindow();
848     stage.setIconified(true);
849 }
850
851 // Metodo para expandir/ restaurar la ventana (puedes personalizar segun tus necesidades)
```

```
852 @FXML
853 private void handleExpandButton(MouseEvent event) {
854     Stage stage = (Stage) rootVBox.getScene().getWindow();
855     if (stage.isMaximized()) {
856         stage.setMaximized(false);
857     } else {
858         stage.setMaximized(true);
859     }
860 }
861 @FXML
862 public void showRulesDialog(MouseEvent event) {
863     try {
864         FXMLLoader loader = new FXMLLoader(getClass().getResource("Rules.fxml"));
865         Parent rulesRoot = loader.load();
866
867         Stage rulesStage = new Stage();
868         rulesStage.initModality(Modality.APPLICATION_MODAL);
869         rulesStage.setTitle("Reglas del Juego");
870
871         Scene rulesScene = new Scene(rulesRoot, 400, 300);
872         rulesStage.setScene(rulesScene);
873         rulesStage.showAndWait();
874     } catch (IOException e) {
875         e.printStackTrace();
876     }
877 }
878
879 private final AtomicBoolean isMoveAlertShown = new AtomicBoolean(false);
880 public void getDirection(Soldier s) {
881     final boolean[] result = {false};
882     String text = s.getName() + " " + s.getHP() + " HP";
883     Alert alerta = new Alert(Alert.AlertType.CONFIRMATION);
884     alerta.setTitle("Confirmacion");
885     alerta.setHeaderText(text + " seleccionado");
886     alerta.setContentText("Por favor, elige una opcion.");
887
888     ButtonType botonSi = new ButtonType("Mover Soldado");
889     ButtonType botonNo = new ButtonType("Cancelar");
890     alerta.getButtonTypes().setAll(botonSi, botonNo);
891
892     alerta.showAndWait().ifPresent(resultado -> {
893         if (resultado == botonSi) {
894             Platform.runLater(() -> {
895                 Alert moveAlert = new Alert(Alert.AlertType.INFORMATION);
896                 moveAlert.setTitle("Mueve tu Soldado");
897                 moveAlert.setHeaderText("Usa las teclas para mover tu soldado.");
898                 moveAlert.setContentText("AWXD (vertical-horizontal), QEZC (diagonal) o S  
(no moverse)");
899
900                 Scene moveScene = moveAlert.getDialogPane().getScene();
901                 moveScene.setOnKeyPressed(event -> {
902                     KeyCode pressedKey = event.getCode();
903
904                     if (!isMoveAlertShown.get()) {
905                         moveSoldado(pressedKey, s);
906                         isMoveAlertShown.set(true);
```

```
907         moveAlert.close();
908     }
909     });
910
911     moveAlert.showAndWait();
912     isMoveAlertShown.set(false);
913     });
914 }
915 });
916
917
918 }
919 private void moveSoldado(KeyCode key, Soldier s) {
920     int x = 0, y = 0;
921
922     switch (key) {
923         case W:
924             y = -1;
925             break;
926         case A:
927             x = -1;
928             break;
929         case D:
930             x = 1;
931             break;
932         case S:
933             // No moverse en la posicion actual
934             break;
935         case Q:
936             x = -1;
937             y = -1;
938             break;
939         case E:
940             x = 1;
941             y = -1;
942             break;
943         case Z:
944             x = -1;
945             y = 1;
946             break;
947         case C:
948             x = 1;
949             y = 1;
950             break;
951         case X:
952             y = 1;
953             break;
954         default:
955             break;
956     }
957
958     // Calcular la nueva posicion
959     int newY = s.getY() + y;
960     int newX = s.getX() + x;
961
962     // Verificar si la nueva posicion es valida
```

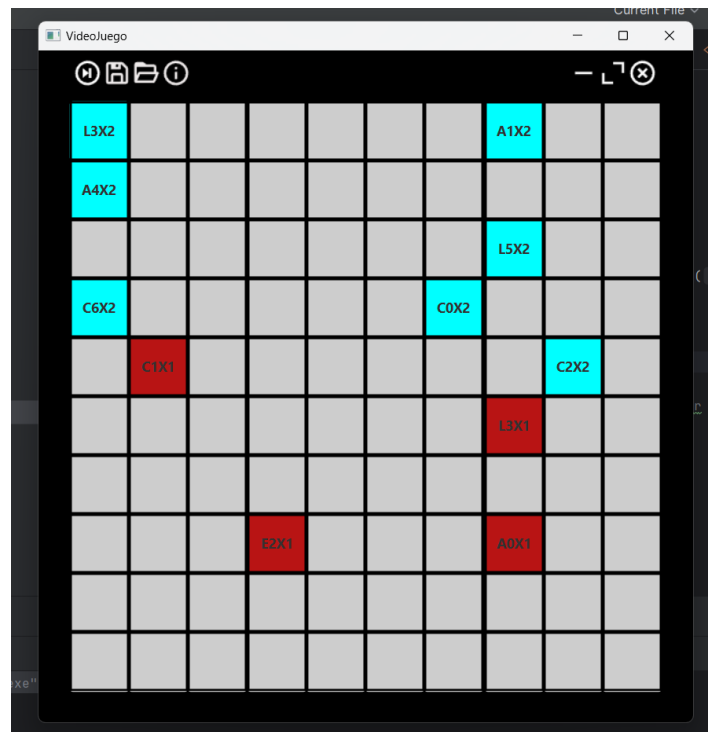
```
963     if (isValidMove(newX, newY)) {
964         // Mover el soldado
965         move(s, newX, newY);
966         // Actualizar la posicion del soldado en la matriz map.getTable()
967
968     } else {
969         showAlert("Casilla invalida (Solo puedes moverte una casilla en cada direccion)",
970             "Error");
971     }
972 }
973
974 // Metodo para verificar si la posicion es valida
975 private boolean isValidMove(int x, int y) {
976     return x >= 0 && x < 10 && y >= 0 && y < 10;
977 }
978 @FXML
979 public void saveGame(MouseEvent event){
980     // Crear un objeto FileChooser
981     FileChooser fileChooser = new FileChooser();
982     fileChooser.setTitle("Guardar Archivo");
983
984     // Agregar una extension al FileChooser (opcional)
985     FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Archivos de
986         Texto (*.txt)", "*.txt");
987     fileChooser.getExtensionFilters().add(extFilter);
988
989     Stage stage = (Stage) rootVBox.getScene().getWindow();
990
991     // Mostrar el cuadro de dialogo para guardar archivo
992     File file = fileChooser.showSaveDialog(stage);
993
994     if (file != null) {
995         // El usuario ha seleccionado un archivo, ahora puedes escribir en el
996         writeToFile(file);
997     }
998 }
999 private void writeToFile(File file) {
1000     try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
1001         // Escribir contenido en el archivo
1002         writer.write("#" + a1.getId());
1003         writer.newLine();
1004         for(Soldier s: a1.getSoldiers()){
1005             writer.write(s.toString());
1006             writer.newLine();
1007         }
1008         writer.newLine();
1009         writer.write("#" + a2.getId());
1010         writer.newLine();
1011         for(Soldier s: a2.getSoldiers()){
1012             writer.write(s.toString());
1013             writer.newLine();
1014         }
1015     } catch (IOException e) {
1016         e.printStackTrace();
1017     }
```

```
1017 }
1018 @FXML
1019 public void loadGame(MouseEvent event){
1020     FileChooser fileChooser = new FileChooser();
1021     fileChooser.setTitle("Cargar Archivo");
1022
1023     // Agregar una extension al FileChooser (opcional)
1024     FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Archivos de
        Texto (*.txt)", "*.txt");
1025     fileChooser.getExtensionFilters().add(extFilter);
1026
1027     Stage stage = (Stage) rootVBox.getScene().getWindow();
1028
1029     // Mostrar el cuadro de dialogo para cargar archivo
1030     File file = fileChooser.showOpenDialog(stage);
1031
1032     if (file != null) {
1033         // El usuario ha seleccionado un archivo, ahora puedes leer de el
1034         readFromFile(file);
1035     }
1036 }
1037 public void readFromFile(File file) {
1038     try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
1039         // Limpiar ejercitos existentes antes de cargar nuevos datos
1040         a1.clearSoldiers();
1041         a2.clearSoldiers();
1042
1043         String line;
1044         while ((line = reader.readLine()) != null) {
1045             if (line.startsWith("#")) {
1046                 // Identificar el ejercito por el ID
1047                 int armyId = Integer.parseInt(line.substring(1));
1048                 Army currentArmy = (armyId == a1.getId()) ? a1 : a2;
1049
1050                 // Leer las siguientes lineas correspondientes a los soldados
1051                 while ((line = reader.readLine()) != null && (!line.startsWith("#") &&
                    !line.trim().isEmpty())) {
1052                     // Crear un nuevo soldado a partir de la linea y agregarlo al ejercito
1053                     Soldier soldier = Soldier.parseSoldier(line);
1054                     currentArmy.addSoldier(soldier);
1055                 }
1056             }
1057         }
1058         // Por ejemplo, volver a colocar los soldados en la interfaz grafica
1059         map.initializeTable();
1060         map.deployArmy(a1);
1061         map.deployArmy(a2);
1062
1063         placeSoldiers();
1064     } catch (IOException e) {
1065         e.printStackTrace();
1066     }
1067 }
1068 }
```

- En este código, tengo el método `onButtonClick`, que lo que hace es primero buscar si hay un

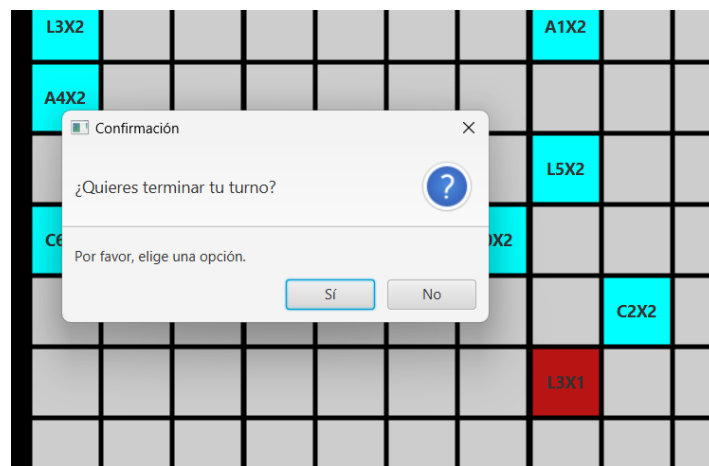
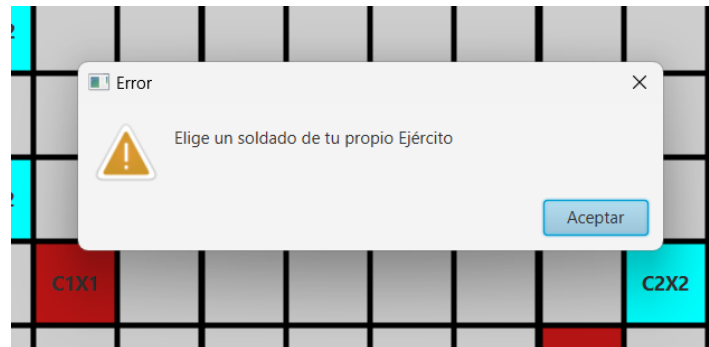
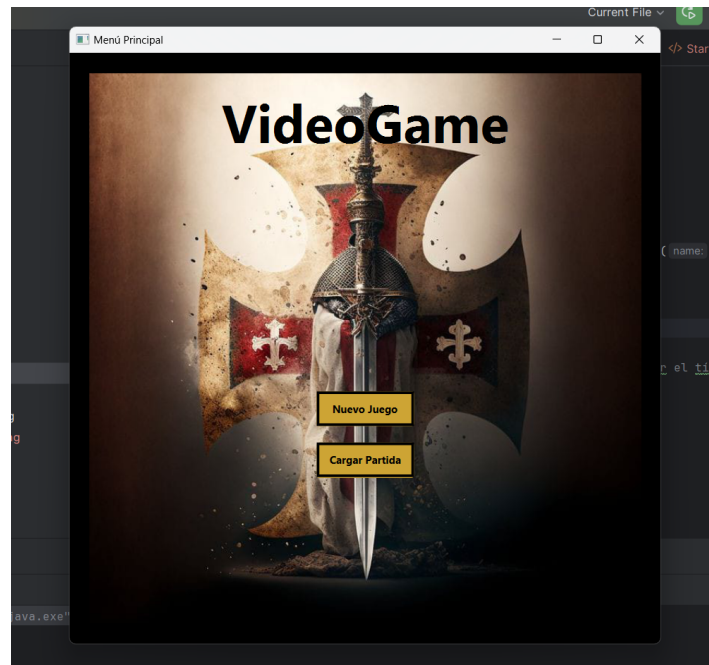
soldado en esa posición, (usando de referencia el mapa) una vez que detecta un soldado, revisa si es del ejército respectivo al turno actual (si es turno del ejército 1 o 2) para que solo puedas mover soldados cuando te toca a ti.

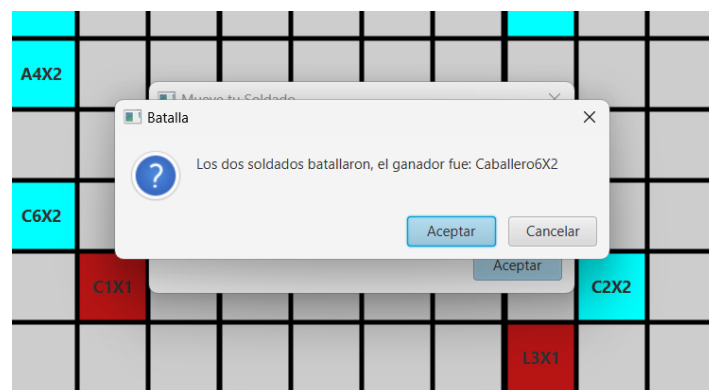
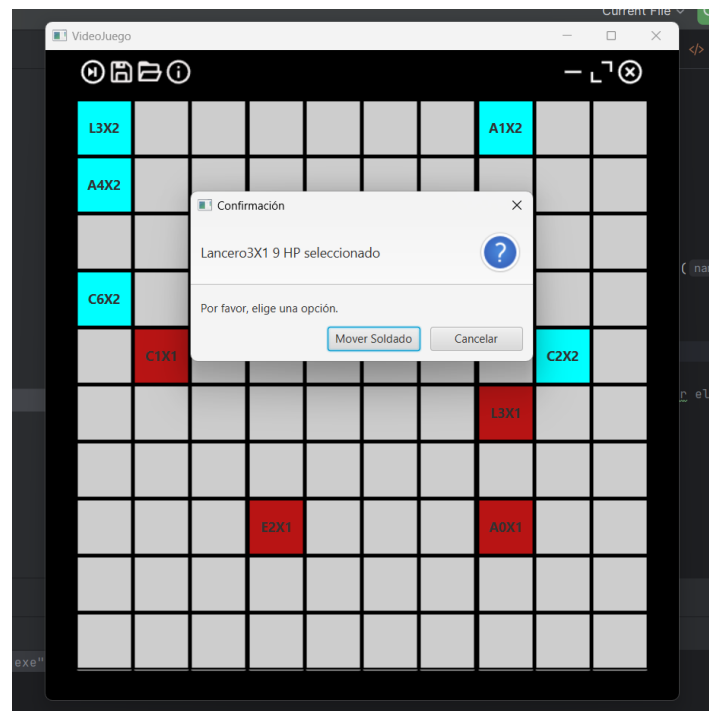
- Luego de eso, pregunto al usuario a que coordenadas lo quiere mover, revisando que sea una coordenada a 1 casilla de distancia, usando las teclas QWEADZXC.
- Luego de eso ve si hay un soldado en la casilla de destino y si lo hay, revisa que no sea del mismo ejército. Si no lo es realiza una batalla y coloca ahí al ganador, para terminar.
- Esa es la lógica principal, también está el método runGame que trabaja en un hilo, constantemente refrescando los gráficos. Y el resto de métodos son simplemente para que la interfaz funcione, como los métodos para minimizar o cerrar usando los botones de la barra de tareas.
- También está el método para determinar si el juego se ha terminado o no y se cierra la pestaña.
- Al final el tablero queda de esta forma al iniciar el juego.

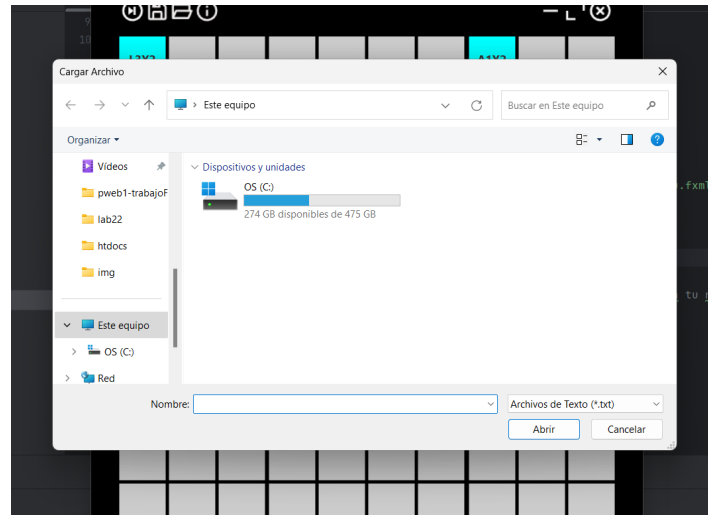


- Y con algunas capturas del funcionamiento del videojuego.









- Estos cambios fueron subidos al repositorio, a través de commits.

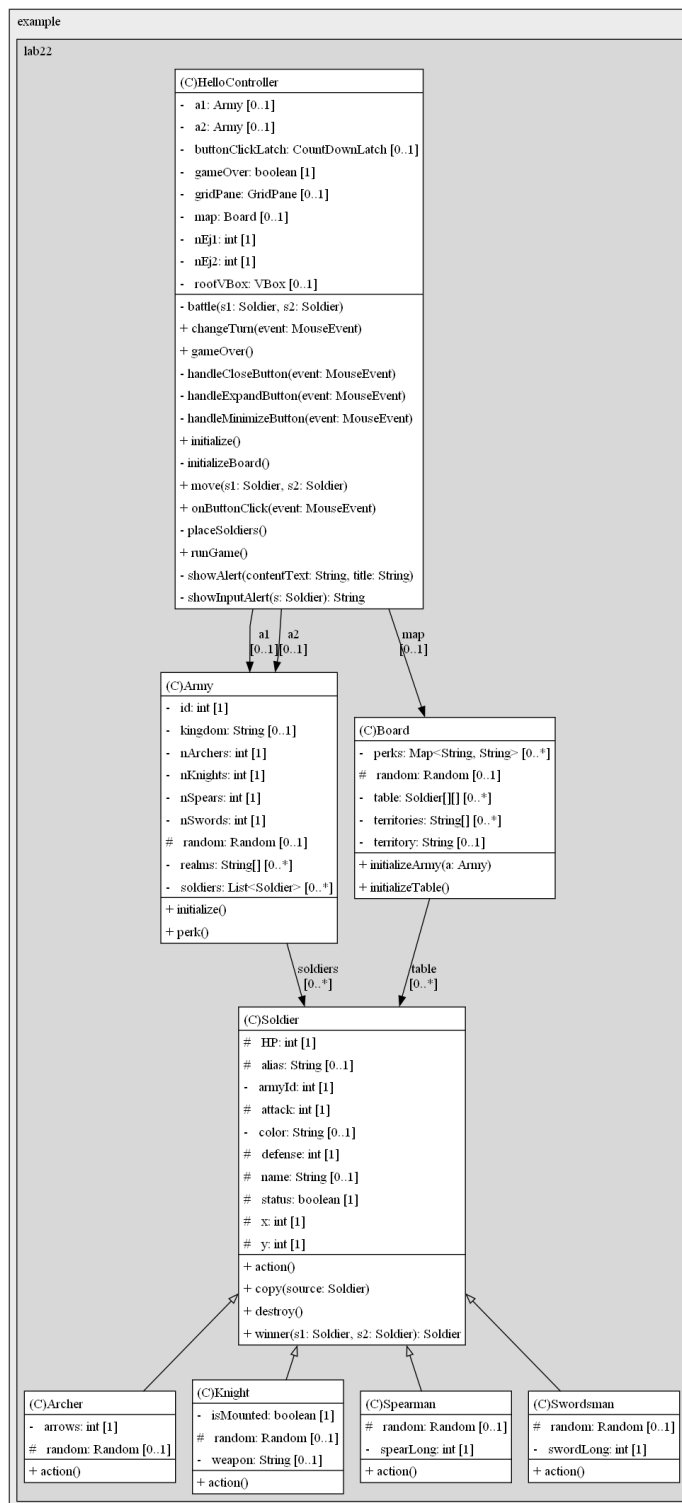
Listing 6: Commit

```
1069 $commit feec5bf606898afbbaaadd1a80b1f0c0c356a83f (HEAD -> main, origin/main)
1070 Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
1071 Date: Sat Jan 20 11:48:31 2024 -0500
1072
1073     Estableciendo las clases necesarias y realizando pruebas de JavaFx, usando IntelliJ Idea
```

Listing 7: Commit

```
1074 $commit 44f6eaa51af046b665943d901d3ac65b04040216 (HEAD -> main, origin/main)
1075 Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
1076 Date: Mon Jan 22 13:38:31 2024 -0500
1077
1078     Trabajo terminado, el juego es 100% funcional
```

- Además, aquí está el diagrama UML de todo el proyecto.



## 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			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	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	2	
<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	4	
<b>Total</b>		20		18	

## 6. Referencias

- Fundamentos de la programación 2 - Tópicos de la programación Orientada a Objetos (Marco Aedo)