

Informe de Laboratorio 22

Tema: Interfaz Gráfica de Usuario

Nota

Estudiante	Escuela	Asignatura
Christian Mestas Zegarra cmestasz@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
22	Interfaz Gráfica de Usuario	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 15 Enero 2024	Al 22 Enero 2024

1. Tarea

■ Item 1:

- 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 Microsoft Windows 10 Pro 64 bits
- Visual Studio Code 1.82.2
- Java Development Kit 17.0.1
- Git 2.41.0.windows.1
- Windows PowerShell 5.1.19041.3031
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos
- HashMap de Objetos
- ArrayList de Objetos
- Agregación y composición
- Herencia y polimorfismo
- Miembros de clase e instancia
- Interfaz gráfica de usuario

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/cmestasz/fp2-23b.git>
- URL para el laboratorio 20 en el Repositorio GitHub.
- <https://github.com/cmestasz/fp2-23b/tree/main/fase03/lab22>

4. Actividades con el repositorio GitHub

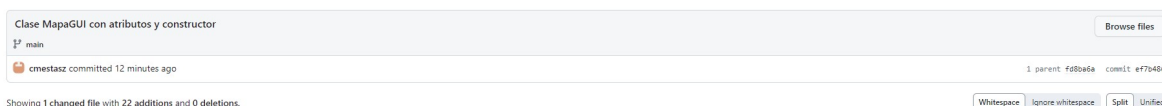
commits.bash

```
1 $ git add Arquero.java
2
3 $ git add Caballero.java
4
5 $ git add Espadachin.java
6
7 $ git add Lancero.java
8
9 $ git add Soldado.java
10
11 $ git add Mapa.java
12
13 $ git add Videojuego.java
14
15 $ git commit -m "Clases del laboratorio 20"
16 [main fd8ba6a] Clases del laboratorio 20
17 7 files changed, 426 insertions(+)
18 create mode 100644 fase03/lab22/Arquero.java
19 create mode 100644 fase03/lab22/Caballero.java
20 create mode 100644 fase03/lab22/Espadachin.java
21 create mode 100644 fase03/lab22/Lancero.java
22 create mode 100644 fase03/lab22/Mapa.java
23 create mode 100644 fase03/lab22/Soldado.java
24 create mode 100644 fase03/lab22/Videojuego.java
25
26 $ git add Mapa
27 Mapa.java          MapaSuperior.java
28 MapaGUI.java       MapaSuperiorGUI.java
29
30 $ git add MapaGUI.java
31
32 $ git commit -m "Clase MapaGUI con atributos y constructor"
33 [main ef7b486] Clase MapaGUI con atributos y constructor
34 1 file changed, 22 insertions(+)
35 create mode 100644 fase03/lab22/MapaGUI.java
36
37 $ git add MapaGUI.java
38
39 $ git commit -m "Clase MapaGUI completa"
40 [main 7be2ea7] Clase MapaGUI completa
41 1 file changed, 30 insertions(+)
42
43 $ git add Mapa.java
44
45 $ git add Soldado.java
46
47 $ git commit -m "Clase Mapa con interfaz grafica en MapaGUI"
48 [main 2979a36] Clase Mapa con interfaz grafica en MapaGUI
49 2 files changed, 42 insertions(+), 76 deletions(-)
50
51 $ git add MapaSuperior.java
52
```

```
53 $ git commit -m "Clase MapaSuperior para manejar las batallas internas"
54 [main edec0a3] Clase MapaSuperior para manejar las batallas internas
55 1 file changed, 39 insertions(+)
56 create mode 100644 fase03/lab22/MapaSuperior.java
57
58 $ git add MapaSuperior.java
59
60 $ git add MapaSuperiorGUI.java
61
62 $ git commit -m "Clase MapaSuperiorGUI con atributos y constructor"
63 [main c0b4ba8] Clase MapaSuperiorGUI con atributos y constructor
64 2 files changed, 51 insertions(+)
65 create mode 100644 fase03/lab22/MapaSuperiorGUI.java
66
67 $ git add MapaSuperiorGUI.java
68
69 $ git commit -m "Clase MapaSuperiorGUI para manejar la interfaz grafica"
70 [main 8bfb33e] Clase MapaSuperiorGUI para manejar la interfaz grafica
71 1 file changed, 34 insertions(+)
72
73 $ git add Videojuego.java
74
75 $ git commit -m "Clase Videojuego para el comportamiento iterativo"
76 [main 6377d8b] Clase Videojuego para el comportamiento iterativo
77 1 file changed, 12 insertions(+), 8 deletions(-)
78
79 $ git push
80 Enumerating objects: 59, done.
81 Counting objects: 100% (59/59), done.
82 Delta compression using up to 4 threads
83 Compressing objects: 100% (53/53), done.
84 Writing objects: 100% (54/54), 16.91 KiB | 753.00 KiB/s, done.
85 Total 54 (delta 26), reused 0 (delta 0), pack-reused 0
86 remote: Resolving deltas: 100% (26/26), completed with 4 local objects.
87 To https://github.com/cmestasz/fp2-23b.git
88 d31824d..6377d8b main -> main
```



Primer Commit.



Segundo Commit.

Clase MapaGUI completa

 main

 cimestasz committed 12 minutes ago

1 parent e779486 commit 78e2ea7

Showing 1 changed file with 30 additions and 0 deletions.

Whitespace


Ignore whitespace


Split

Unified

Tercer Commit.

Clase Mapa con interfaz grafica en MapaGUI

 main

 cimestasz committed 8 minutes ago

1 parent 78e2ea7 commit 2979a36

Showing 2 changed files with 42 additions and 76 deletions.

Whitespace


Ignore whitespace


Split

Unified

Cuarto Commit.

Clase MapaSuperior para manejar las batallas internas

 main

 cimestasz committed 5 minutes ago

1 parent 2979a36 commit edec8a3

Showing 1 changed file with 39 additions and 0 deletions.

Whitespace


Ignore whitespace


Split

Unified

Quinto Commit.

Clase MapaSuperiorGUI con atributos y constructor

 main

 cimestasz committed 5 minutes ago

1 parent edec8a3 commit c8b4ba8

Showing 2 changed files with 51 additions and 0 deletions.

Whitespace


Ignore whitespace

Split

Unified

Sexto Commit.

Clase MapaSuperiorGUI para manejar la interfaz grafica

 main

 cimestasz committed 5 minutes ago

1 parent c8b4ba8 commit 80fb33e

Showing 1 changed file with 34 additions and 0 deletions.

Whitespace


Ignore whitespace


Split

Unified

Septimo Commit.

Clase Videojuego para el comportamiento iterativo

 main

 cimestasz committed 5 minutes ago

1 parent 80fb33e commit 6377d8b

Showing 1 changed file with 12 additions and 8 deletions.

Whitespace

Ignore whitespace

Split

Unified

Octavo Commit.

5. Código desarrollado

Soldado.java

```
1 package fase03.lab22;
2
3 import javax.swing.*;
4
5 public abstract class Soldado extends JLabel {
6     private static int totalSoldados1;
7     private static int totalSoldados2;
8     private String nombre;
9     private int equipo;
10    private int vidaInicial;
11    private int vidaActual;
12    private int ataque;
13    private int defensa;
14    private String clase;
15
16    public Soldado(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
17        String clase) {
18        super(nombre.charAt(0) + nombre.substring(nombre.length() - 3),
19            SwingConstants.CENTER);
20        this.nombre = nombre;
21        this.equipo = equipo;
22        this.vidaInicial = vidaInicial;
23        this.vidaActual = vidaInicial;
24        this.ataque = ataque;
25        this.defensa = defensa;
26        this.clase = clase;
27        if (equipo == 1)
28            totalSoldados1++;
29        else
30            totalSoldados2++;
31    }
32
33    public String getNombre() {
34        return nombre;
35    }
36
37    public int getEquipo() {
38        return equipo;
39    }
40
41    public int getVidaInicial() {
42        return vidaInicial;
43    }
44
45    public int getVidaActual() {
46        return vidaActual;
47    }
48
49    public int getAtaque() {
50        return ataque;
51    }
52}
```

```
51 public int getDefensa() {
52     return defensa;
53 }
54
55 public String getClase() {
56     return clase;
57 }
58
59 public static int getTotalSoldados1() {
60     return totalSoldados1;
61 }
62
63 public static int getTotalSoldados2() {
64     return totalSoldados2;
65 }
66
67 public void aumentarVida() {
68     vidaActual++;
69 }
70
71 public void atacar(Soldado otro, int instancias) {
72     otro.herir(instancias * (Math.max(0, ataque - otro.getDefensa())));
73 }
74
75 public void herir(int vida) {
76     vidaActual -= vida;
77 }
78
79 public void defender() {
80     defensa++;
81 }
82 }
```

- Clase abstracta que guarda nombre y vida del soldado.
- Posee getters para todos los atributos.
- Posee métodos para el combate como aumentarVida, atacar, herir, defender.
- Hereda de la clase JLabel para la interfaz gráfica

Caballero.java

```
1 package fase03.lab22;
2
3 public class Caballero extends Soldado {
4     private static int totalCaballeros1;
5     private static int totalCaballeros2;
6     private String arma = "ESPADA";
7     private boolean montado = false;
8
9     public Caballero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
10         String clase) {
11         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
12         if (equipo == 1)
13             totalCaballeros1++;
14     }
15 }
```

```
13         else
14             totalCaballeros2++;
15     }
16
17     public void cambiarArma(String arma) {
18         this.arma = arma;
19     }
20
21     public void montar(Soldado otro) {
22         montado = true;
23         cambiarArma("LANZA");
24         embestir(otro);
25     }
26
27     public void desmontar() {
28         montado = false;
29         cambiarArma("ESPADA");
30         defender();
31     }
32
33     public void embestir(Soldado otro) {
34         if (montado)
35             atacar(otro, 3);
36         else
37             atacar(otro, 2);
38     }
39
40     public static int getTotalCaballeros1() {
41         return totalCaballeros1;
42     }
43
44     public static int getTotalCaballeros2() {
45         return totalCaballeros2;
46     }
47 }
```

- Clase que mantiene atributos y métodos independientes de un caballero.
- Arma, montado, cambiarArma, montar, desmontar, embestir.

Arquero.java

```
1 package fase03.lab22;
2
3 public class Arquero extends Soldado {
4     private static int totalArqueros1;
5     private static int totalArqueros2;
6     private int flechas = 10;
7
8     public Arquero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
9         String clase) {
10         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
11         if (equipo == 1)
12             totalArqueros1++;
13         else
14             totalArqueros2++;
```



```
14     }
15
16     public void disparar(Soldado otro) {
17         if (flechas > 0) {
18             atacar(otro, 1);
19             flechas--;
20         }
21     }
22
23     public static int getTotalArqueros1() {
24         return totalArqueros1;
25     }
26
27     public static int getTotalArqueros2() {
28         return totalArqueros2;
29     }
30 }
```

- Clase que mantiene atributos y métodos independientes de un arquero.
- Flechas, disparar.

Espadachin.java

```
1 package fase03.lab22;
2
3 public class Espadachin extends Soldado {
4     private static int totalEspadachines1;
5     private static int totalEspadachines2;
6     private double longitudEspada;
7
8     public Espadachin(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
9         String clase) {
10         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
11         if (equipo == 1)
12             totalEspadachines1++;
13         else
14             totalEspadachines2++;
15     }
16
17     public void generarMuroEscudos() {
18         System.out.println(getNombre() + " genera un muro de escudos!");
19     }
20
21     public static int getTotalEspadachines1() {
22         return totalEspadachines1;
23     }
24
25     public static int getTotalEspadachines2() {
26         return totalEspadachines2;
27     }
28 }
```

- Clase que mantiene atributos y métodos independientes de un espadachin.

- Longitud de espada, generarMuroEscudos.

Lancero.java

```
1 package fase03.lab22;
2
3 public class Lancero extends Soldado {
4     private static int totalLanceros1;
5     private static int totalLanceros2;
6     private int longitudLanza;
7
8     public Lancero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
9         String clase) {
10         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
11         if (equipo == 1)
12             totalLanceros1++;
13         else
14             totalLanceros2++;
15     }
16
17     public void schiltrom() {
18         defender();
19     }
20
21     public static int getTotalLanceros1() {
22         return totalLanceros1;
23     }
24
25     public static int getTotalLanceros2() {
26         return totalLanceros2;
27     }
28 }
```

- Clase que mantiene atributos y métodos independientes de un lancero.
- Longitud de lanza, schiltrom.

Mapa.java

```
1 package fase03.lab22;
2
3 import java.util.ArrayList;
4 import java.util.HashMap;
5 import java.util.Random;
6
7 public class Mapa {
8     private MapaGUI gui = new MapaGUI("Videojuego");
9
10     private HashMap<String, Soldado> soldados = new HashMap<String, Soldado>();
11     private ArrayList<Soldado> listaSoldados1 = new ArrayList<Soldado>();
12     private ArrayList<Soldado> listaSoldados2 = new ArrayList<Soldado>();
13     private String reino1;
14     private String reino2;
15     private String terreno;
16     private final String[] CLASES = { "CABALLERO", "ARQUERO", "ESPADACHIN", "LANCERO" };
```

```
17 private final Random RANDOM = new Random();
18
19 public Mapa(String reino1, String reino2, String terreno) {
20     this.reino1 = reino1;
21     this.reino2 = reino2;
22     this.terreno = terreno;
23     inicializarSoldados(soldados, listaSoldados1, 1);
24     inicializarSoldados(soldados, listaSoldados2, 2);
25     mostrarVentana();
26 }
27
28 private void mostrarVentana() {
29     String desc1 = "";
30     desc1 += "El terreno elegido es: " + terreno + "\n";
31     desc1 += "El reino 1 es: " + reino1 + "\n";
32     desc1 += "El reino 2 es: " + reino2 + "\n";
33     desc1 += verificarVentaja(reino1);
34     desc1 += verificarVentaja(reino2);
35     String desc2 = obtenerEstado();
36     String desc3 = obtenerGanador();
37     gui.mostrarVentana(soldados, desc1, desc2, desc3);
38 }
39
40 private void inicializarSoldados(HashMap<String, Soldado> mapaSoldados,
41     ArrayList<Soldado> listaSoldados,
42     int equipo) {
43     int cantidad = RANDOM.nextInt(10) + 1;
44     for (int i = 0; i < cantidad; i++) {
45         String clase = CLASES[RANDOM.nextInt(CLASES.length)];
46         int vida, ataque, defensa, fila, columna;
47         do {
48             fila = RANDOM.nextInt(10);
49             columna = RANDOM.nextInt(10);
50         } while (mapaSoldados.containsKey(generarLlave(fila, columna)));
51         String nombre = clase + i + "X" + equipo;
52         Soldado soldado = null;
53         switch (clase) {
54             case "CABALLERO":
55                 ataque = 13;
56                 defensa = 7;
57                 vida = RANDOM.nextInt(3) + 10;
58                 soldado = new Caballero(nombre, equipo, vida, ataque, defensa, clase);
59                 break;
60             case "ARQUERO":
61                 ataque = 7;
62                 defensa = 3;
63                 vida = RANDOM.nextInt(3) + 3;
64                 soldado = new Arquero(nombre, equipo, vida, ataque, defensa, clase);
65                 break;
66             case "ESPADACHIN":
67                 ataque = 10;
68                 defensa = 8;
69                 vida = RANDOM.nextInt(3) + 8;
70                 soldado = new Espadachin(nombre, equipo, vida, ataque, defensa, clase);
71                 break;
72             case "LANCERO":
```

```
72         ataque = 5;
73         defensa = 10;
74         vida = RANDOM.nextInt(4) + 5;
75         soldado = new Lancero(nombre, equipo, vida, ataque, defensa, clase);
76         break;
77     }
78     mapaSoldados.put(generarLlave(fila, columna), soldado);
79     listaSoldados.add(soldado);
80 }
81 }
82
83 private String verificarVentaja(String reino) {
84     switch (reino) {
85         case "INGLATERRA":
86             if (terreno.equals("BOSQUE"))
87                 mejorarSoldados(reino);
88             break;
89         case "FRANCIA":
90             if (terreno.equals("CAMPO ABIERTO"))
91                 mejorarSoldados(reino);
92             break;
93         case "CASTILLA-ARAGON":
94             if (terreno.equals("MONTANA"))
95                 mejorarSoldados(reino);
96             break;
97         case "MOROS":
98             if (terreno.equals("DESIERTO"))
99                 mejorarSoldados(reino);
100            break;
101         case "SACRO IMPERIO":
102             if (terreno.equals("BOSQUE") || terreno.equals("PLAYA") ||
103                 terreno.equals("CAMPO ABIERTO"))
104                 return mejorarSoldados(reino);
105             break;
106     }
107     return "";
108 }
109
110 private String mejorarSoldados(String reino) {
111     ArrayList<Soldado> soldados = reino.equals(reino1) ? listaSoldados1 : listaSoldados2;
112     for (Soldado soldado : soldados)
113         soldado.aumentarVida();
114     return reino + " tiene ventaja en el terreno!\n";
115 }
116
117 private String obtenerEstado() {
118     String descripcion = "";
119     descripcion += String.format(
120         "Ejercito 1: %s\nCantidad total de soldados: %d\nEspadachines: %d\nArqueros: %d\nCaballeros: %d\nLanceros: %d\n",
121         reino1, Soldado.getTotalSoldados1(), Espadachin.getTotalEspadachines1(),
122         Arquero.getTotalArqueros1(),
123         Caballero.getTotalCaballeros1(), Lancero.getTotalLanceros1());
124     descripcion += String.format(
125         "Ejercito 2: %s\nCantidad total de soldados: %d\nEspadachines: %d\nArqueros: %d\nCaballeros: %d\nLanceros: %d\n",
```

```
124         reino2, Soldado.getTotalSoldados2(), Espadachin.getTotalEspadachines2(),
125         Arquero.getTotalArqueros2(),
126         Caballero.getTotalCaballeros2(), Lancero.getTotalLanceros2());
127     return descripcion;
128 }
129 private String obtenerGanador() {
130     String descripcion = "";
131     int vida1 = vidaTotal(listaSoldados1);
132     int vida2 = vidaTotal(listaSoldados2);
133     double chance1 = 1.0 * vida1 / (vida1 + vida2);
134     double chance2 = 1 - chance1;
135     descripcion += String.format("%s: %d%.5f%% de probabilidad de victoria%n%n",
136     reino1, vida1, chance1);
137     descripcion += String.format("%s: %d%.5f%% de probabilidad de victoria%n%n",
138     reino2, vida2, chance2);
139     double aleatorio = RANDOM.nextDouble(1);
140     if (aleatorio < chance1)
141         descripcion += ("Gana " + reino1 + "\n");
142     else
143         descripcion += ("Gana " + reino2 + "\n");
144     descripcion += ("Aleatorio generado: " + aleatorio + "\n");
145     return descripcion;
146 }
147 private int vidaTotal(ArrayList<Soldado> soldados) {
148     int suma = 0;
149     for (Soldado soldado : soldados)
150         suma += soldado.getVidaActual();
151     return suma;
152 }
153 private String generarLlave(int fila, int columna) {
154     return fila + "," + columna;
155 }
156 public static char intToChar(int n) {
157     return (char) (n + 'A' - 1);
158 }
159 public static int charToInt(char c) {
160     return (int) (c - 'A' + 1);
161 }
162 }
163 }
164 }
```

- Ciclo de un juego contenido dentro del constructor.
- Método inicializarSoldados crea los soldados y los pone en el tablero.
- Método verificarVentaja otorga ventaja al reino aventajado por el terreno.
- Método obtenerEstado retorna el estado del tablero.
- Método obtenerGanador calcula el ganador de la batalla y lo retorna.
- Todos los datos se envían a la clase MapaGUI.

MapaGUI.java

```
1 package fase03.lab22;
2
3 import java.util.HashMap;
4 import javax.swing.*;
5 import java.awt.*;
6
7 public class MapaGUI extends JFrame {
8     private static final int WIDTH = 800;
9     private static final int HEIGHT = 1000;
10    private static final int ROWS = 10;
11    private static final int COLS = 10;
12    private JPanel map = new JPanel(new GridLayout(10, 10));
13    private JPanel descripcion = new JPanel(new GridLayout(1, 3));
14    private JLabel[][] soldadosMapa = new JLabel[ROWS][COLS];
15
16    public MapaGUI(String name) {
17        super(name);
18        setSize(WIDTH, HEIGHT);
19        setLayout(new BorderLayout());
20        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
21    }
22
23    public void mostrarVentana(HashMap<String, Soldado> soldados, String desc1, String desc2,
24        String desc3) {
25        for (int i = 0; i < ROWS; i++) {
26            for (int j = 0; j < COLS; j++) {
27                String llave = generarLlave(i, j);
28                if (soldados.containsKey(llave))
29                    soldadosMapa[i][j] = soldados.get(llave);
30                else
31                    soldadosMapa[i][j] = new JLabel("", SwingConstants.CENTER);
32                soldadosMapa[i][j].setOpaque(true);
33                soldadosMapa[i][j].setBackground(((i + j) % 2 == 0) ? Color.lightGray :
34                    Color.white);
35                soldadosMapa[i][j].setBorder(BorderFactory.createLineBorder(Color.black));
36                map.add(soldadosMapa[i][j]);
37            }
38        }
39
40        descripcion.add(new JTextArea(desc1));
41        descripcion.add(new JTextArea(desc2));
42        descripcion.add(new JTextArea(desc3));
43
44        map.setAlignmentY(CENTER_ALIGNMENT);
45        add(BorderLayout.CENTER, map);
46        add(BorderLayout.SOUTH, descripcion);
47
48        setVisible(true);
49    }
50
51    private String generarLlave(int fila, int columna) {
52        return fila + "," + columna;
53    }
54 }
```

- Clase que hereda de JFrame para la interfaz gráfica.
- Método mostrarVentana ubica todos los elementos y los muestra

MapaSuperior.java

```
1 package fase03.lab22;
2
3 import java.util.HashMap;
4 import java.util.Random;
5
6 public class MapaSuperior {
7     private MapaSuperiorGUI gui;
8
9     private Videojuego videojuego;
10    private HashMap<String, String> batallas = new HashMap<String, String>();
11    private final String[] TIPOS = { "BOSQUE", "CAMPO ABIERTO", "MONTANA", "DESIERTO",
        "PLAYA" };
12    private final String[] REINOS = { "INGLATERRA", "FRANCIA", "CASTILLA-ARAGON", "MOROS",
        "SACRO IMPERIO" };
13    private final Random RANDOM = new Random();
14
15    public MapaSuperior(Videojuego videojuego) {
16        this.videojuego = videojuego;
17        inicializarBatallas(batallas);
18        String reino1 = REINOS[RANDOM.nextInt(REINOS.length)];
19        String reino2 = REINOS[RANDOM.nextInt(REINOS.length)];
20        String terreno = TIPOS[RANDOM.nextInt(TIPOS.length)];
21        gui = new MapaSuperiorGUI("Videojuego", reino1, reino2, terreno, batallas, this);
22        gui.mostrarVentana();
23    }
24
25    private void inicializarBatallas(HashMap<String, String> batallas) {
26        int cantidad = RANDOM.nextInt(10) + 1;
27        for (int i = 0; i < cantidad; i++) {
28            int fila, columna;
29            do {
30                fila = RANDOM.nextInt(10);
31                columna = RANDOM.nextInt(10);
32            } while (batallas.containsKey(generarLlave(fila, columna)));
33            String nombre = "#" + i;
34
35            batallas.put(generarLlave(fila, columna), nombre);
36        }
37    }
38
39    public void terminarGuerra() {
40        videojuego.continuar();
41    }
42
43    private String generarLlave(int fila, int columna) {
44        return fila + "," + columna;
45    }
46 }
```

- Clase que permite acceder a las diferentes batallas dentro del mapa

- Método inicializarBatallas crea las batallas y las pone en el tablero.
- Método terminarGuerra regresa el control al juego principal y pregunta al usuario si iniciar otro juego.

MapaSuperiorGUI.java

```
1 package fase03.lab22;
2
3 import java.util.HashMap;
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8
9 public class MapaSuperiorGUI extends JFrame {
10     private static final int WIDTH = 800;
11     private static final int HEIGHT = 800;
12     private static final int ROWS = 10;
13     private static final int COLS = 10;
14
15     private MapaSuperior mapaSuperior;
16     private String reino1;
17     private String reino2;
18     private String terreno;
19     private JButton[][] batallasMapa = new JButton[ROWS][COLS];
20     private HashMap<String, String> batallas;
21
22     public MapaSuperiorGUI(String name, String reino1, String reino2, String terreno,
23         HashMap<String, String> batallas,
24         MapaSuperior mapaSuperior) {
25         super(name);
26         this.reino1 = reino1;
27         this.reino2 = reino2;
28         this.terreno = terreno;
29         this.batallas = batallas;
30         this.mapaSuperior = mapaSuperior;
31         setSize(WIDTH, HEIGHT);
32         setLayout(new GridLayout(ROWS, COLS));
33         setDefaultCloseOperation(EXIT_ON_CLOSE);
34     }
35
36     public void mostrarVentana() {
37         for (int i = 0; i < ROWS; i++) {
38             for (int j = 0; j < COLS; j++) {
39                 String llave = generarLlave(i, j);
40                 if (batallas.containsKey(llave)) {
41                     batallasMapa[i][j] = new JButton(batallas.get(llave));
42                     batallasMapa[i][j].addActionListener(new BotonListener(llave));
43                 } else {
44                     batallasMapa[i][j] = new JButton("");
45                 }
46                 batallasMapa[i][j].setOpaque(true);
47                 batallasMapa[i][j].setBackground(((i + j) % 2 == 0) ? Color.white :
48                     Color.lightGray);
49                 batallasMapa[i][j].setBorder(BorderFactory.createLineBorder(Color.black));
50                 add(batallasMapa[i][j]);
51             }
52         }
53     }
54 }
```



```
49     }
50 }
51
52     setVisible(true);
53 }
54
55 private String generarLlave(int fila, int columna) {
56     return fila + "," + columna;
57 }
58
59 private class BotonListener implements ActionListener {
60     private String llave;
61
62     public BotonListener(String llave) {
63         this.llave = llave;
64     }
65
66     public void actionPerformed(ActionEvent e) {
67         new Mapa(reino1, reino2, terreno);
68         batallas.remove(llave);
69         JButton boton = (JButton) e.getSource();
70         boton.setText("");
71         boton.removeActionListener(this);
72
73         if (batallas.size() == 0) {
74             mapaSuperior.terminarGuerra();
75         }
76     }
77 }
78 }
```

- Clase que hereda de JFrame para la interfaz gráfica.
- Método mostrarVentana ubica todos los elementos y los muestra
- Se posee la clase interna BotonListener para manejar el evento de acceder a una batalla.
- Cuando ya no quedan batallas, la guerra termina.

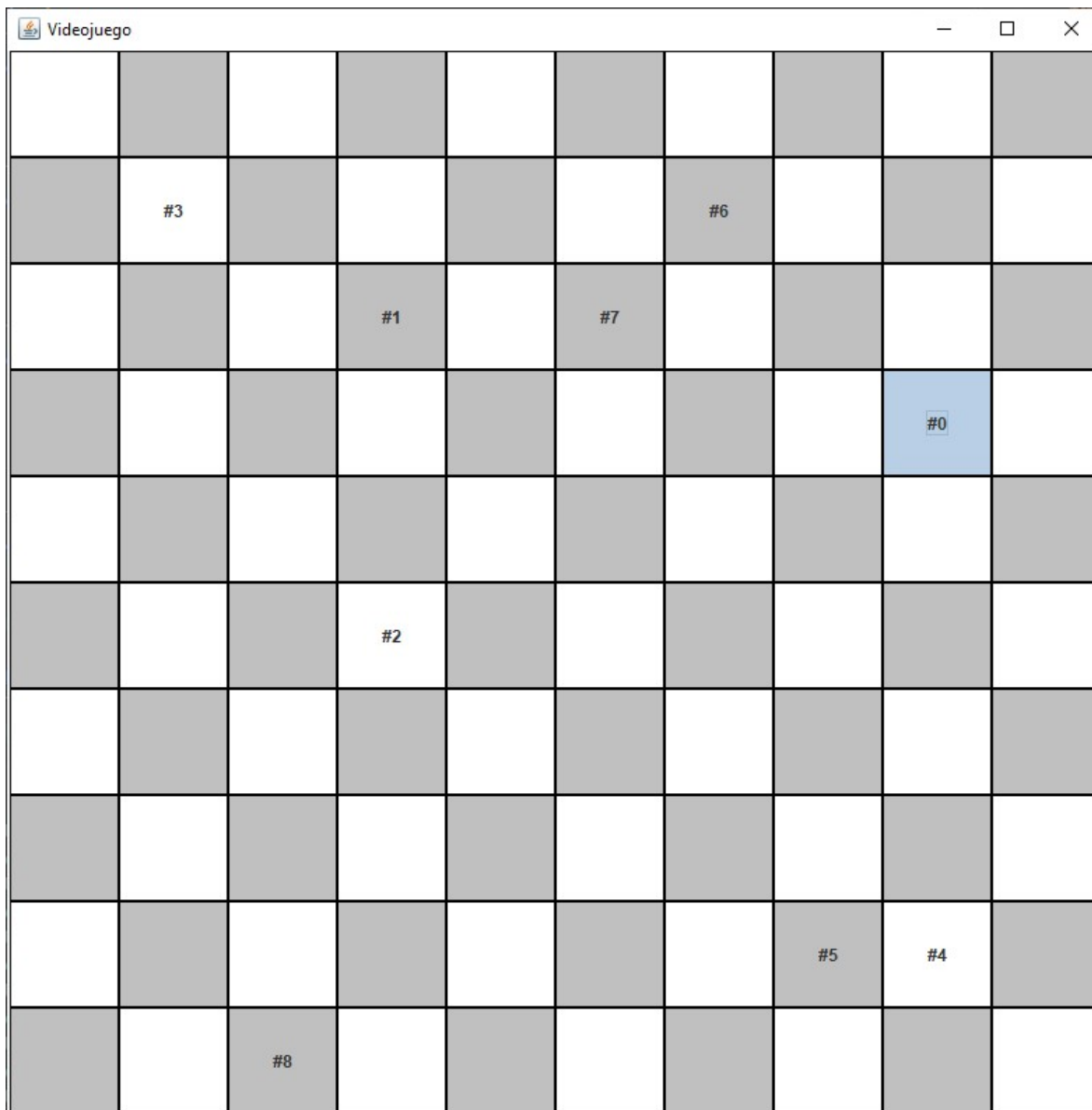
Videojuego.java

```
1 package fase03.lab22;
2
3 import javax.swing.JOptionPane;
4
5 public class Videojuego {
6     public static void main(String[] args) {
7         new Videojuego();
8     }
9
10    public Videojuego() {
11        new MapaSuperior(this);
12    }
13
14    public void continuar() {
15        int continuar = JOptionPane.showConfirmDialog(null, "Desea jugar de nuevo? (S/N): ");
```


```
16         if (continuar == JOptionPane.YES_OPTION)
17             new MapaSuperior(this);
18     }
19 }
```

- Permite el comportamiento iterativo creando nuevos mapas cada vez que acaba la guerra.

6. Ejecución del código



Ejecución.


Videojuego

					L1X2				
		E7X1		E4X1					
							E3X1		
L0X2					C0X1	C2X2			
E1X1				C9X1					
		E5X1			L6X1				
								A8X1	
C2X1									

El terreno elegido es: PLAYA

El reino 1 es: FRANCIA

El reino 2 es: CASTILLA-ARAGON

Ejercito 1: FRANCIA

Cantidad total de soldados: 10

Espadachines: 5

Arqueros: 1

Caballeros: 3

Lanceros: 1

FRANCIA: 89

0.78761% de probabilidad de victoria

Ejercito 2: CASTILLA-ARAGON

Cantidad total de soldados: 3

Espadachines: 0

Arqueros: 0

Caballeros: 1

Lanceros: 2

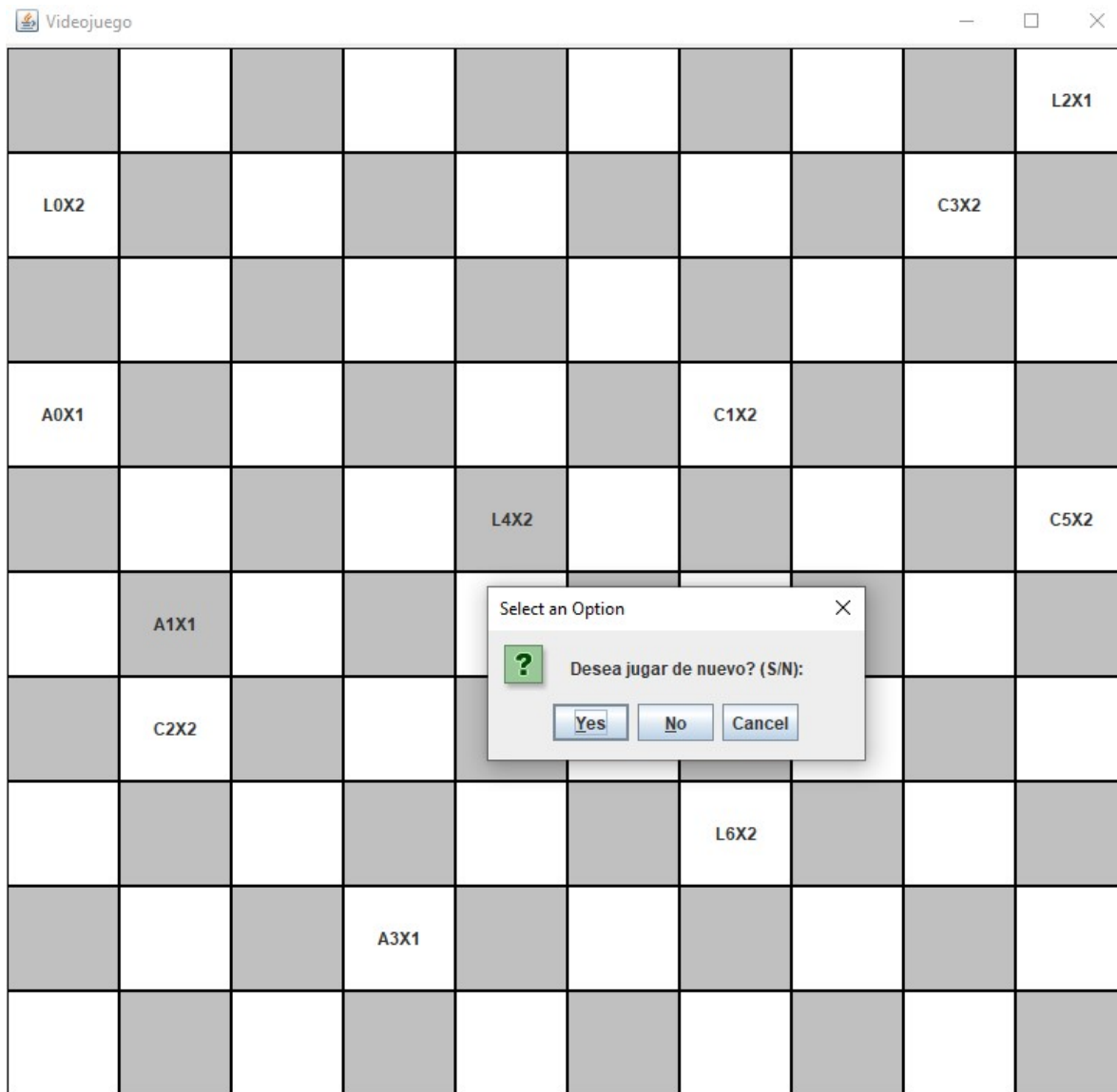
CASTILLA-ARAGON: 24

0.21239% de probabilidad de victoria

Gana FRANCIA

Aleatorio generado: 0.11639522343535058

Ejecución.



El terreno elegido es: PLAYA
El reino 1 es: FRANCIA
El reino 2 es: CASTILLA-ARAGON

Ejercito 1: FRANCIA
Cantidad total de soldados: 61
Espadachines: 16
Arqueros: 16
Caballeros: 15
Lanceros: 14

Ejercito 2: CASTILLA-ARAGON
Cantidad total de soldados: 55
Espadachines: 5
Arqueros: 14
Caballeros: 18
Lanceros: 18

FRANCIA: 17
0.21250% de probabilidad de victoria

CASTILLA-ARAGON: 63
0.78750% de probabilidad de victoria

Gana CASTILLA-ARAGON
Aleatorio generado: 0.3810299287147737

Ejecución.

7. Diagrama UML

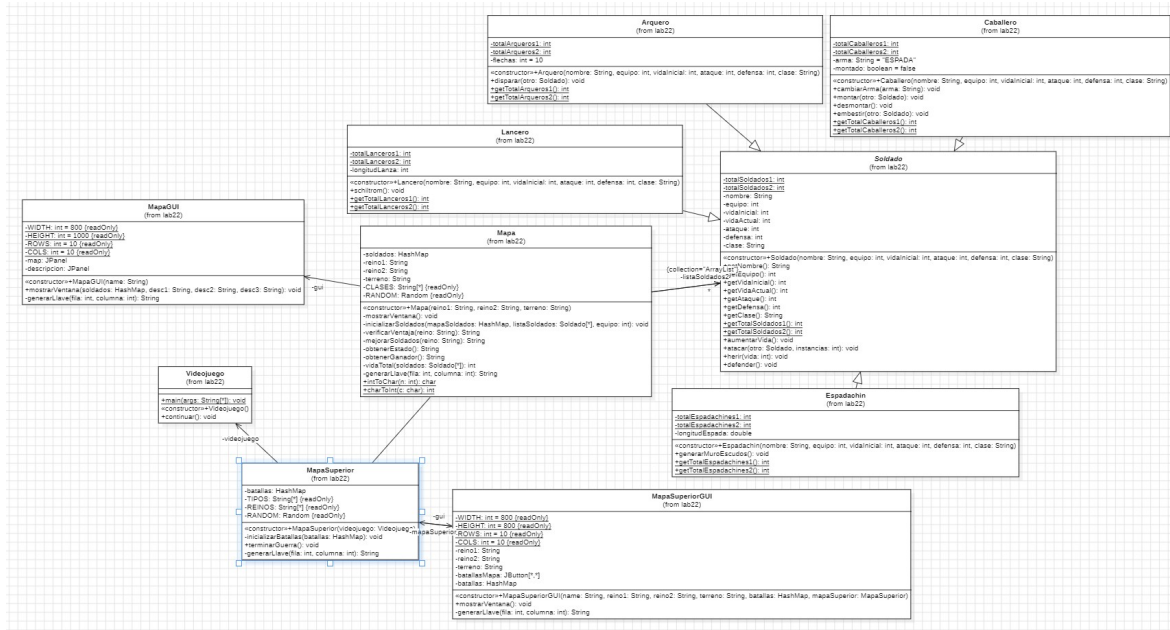


Diagrama UML.

8. Estructura de laboratorio 22

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

```
lab22/  
|--- Soldado.java  
|--- Caballero.java  
|--- Arquero.java  
|--- Espadachin.java  
|--- Lancero.java  
|--- Mapa.java  
|--- MapaGUI.java  
|--- MapaSuperior.java  
|--- MapaSuperiorGUI.java  
|--- Videojuego.java  
|--- commits.bash  
|--- Informe.tex  
|--- Informe.pdf  
|--- img  
|   |--- logo_abet.png  
|   |--- logo_episunsa.png  
|   |--- logo_unsa.jpg  
|   |--- commit01.jpg  
|   |--- commit02.jpg  
|   |--- commit03.jpg  
|   |--- commit04.jpg  
|   |--- commit05.jpg  
|   |--- commit06.jpg  
|   |--- commit07.jpg  
|   |--- commit08.jpg  
|   |--- ejec01.jpg  
|   |--- ejec02.jpg  
|   |---uml.jpg
```

9. Rúbricas

9.1. Entregable Informe

Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

9.2. 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 aprobatoria, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Niveles de desempeño

Puntos	Nivel			
	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

Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1.5	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1.5	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
8. Madurez	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	
Total		20		18.5	

10. Referencias

- Aedo, M. y Castro, E. (2021). FUNDAMENTOS DE PROGRAMACIÓN 2 - Tópicos de Programación Orientada a Objetos. Editorial UNSA.