Informe de Laboratorio 20

Tema: Clase Ejército - Soldado - Mapa. Herencia y Polimorfismo. Miembros de clase

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
20	Clase Ejército - Soldado - Mapa. Herencia y Polimorfismo. Miembros de clase	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 08 Enero 2024	Al 15 Enero 2024

1. Tarea

- **Item 1:**
  • Crear diagrama de clases UML y programa.
  • Crear los miembros de cada clase de la forma más adecuada: como miembros de clase o de instancia.
  • Crear la clase Mapa, que esté constituida por el tablero antes visto, que posicione soldados en ciertas posiciones aleatorias (entre 1 y 10 soldados por cada ejército, sólo 1 ejército por reino). Se deben generar ejércitos de 2 reinos. No se admite guerra civil. El Mapa tiene como atributo el tipo de territorio que es (bosque, campo abierto, montaña, desierto, playa). La cantidad de soldados, así como todos sus atributos se deben generar aleatoriamente.
  • Dibujar el Mapa con las restricciones que sólo 1 soldado como máximo en cada cuadrado.
  • El mapa tiene un solo tipo de territorio.
  • Considerar que el territorio influye en los resultados de las batallas, así cada reino tiene bonus según el territorio: Inglaterra-¿bosque, Francia-¿campo abierto, Castilla-Aragón-¿montaña, Moros-¿desierto, Sacro Imperio Romano- Germánico-¿bosque, playa, campo abierto. En dichos casos, se aumenta el nivel de vida en 1 a todos los soldados del reino beneficiado.
  • En la historia, los ejércitos estaban conformados por diferentes tipos de soldados, que tenían similitudes, pero también particularidades.
  • Basándose en la clase Soldado crear las clases Espadachín, Arquero, Caballero y Lancero. Las

cuatro clases heredan de la superclase Soldado pero aumentan atributos y métodos, o sobrescriben métodos heredados.
• Los espadachines tienen como atributo particular "longitud de espadaz como acción çrear un muro de escudos"que es un tipo de defensa en particular.
• Los caballeros pueden alternar sus armas entre espada y lanza, además de desmontar (sólo se realiza cuando está montando e implica defender y cambiar de arma a espada), montar (sólo se realiza cuando está desmontado e implica montar, cambiar de arma a lanza y envestir). El caballero también puede envestir, ya sea montando o desmontando, cuando es desmontado equivale a atacar 2 veces pero cuando está montando implica a atacar 3 veces.
• Los arqueros tienen un número de flechas disponibles las cuales pueden dispararse y se gastan cuando se hace eso.
• Los lanceros tienen como atributo particular, "longitud de lanzaz como acción "schiltrom"(como una falange que es un tipo de defensa en particular y que aumenta su nivel de defensa en 1).
• Tendrá 2 Ejércitos que pueden ser constituidos sólo por espadachines, caballeros, arqueros y lanceros. No se acepta guerra civil. Crear una estructura de datos conveniente para el tablero. Los soldados del primer ejército se almacenarán en un arreglo estándar y los soldados del segundo ejército se almacenarán en un ArrayList. Cada soldado tendrá un nombre autogenerado: Espadachin0X1, Arquero1X1, Caballero2X2, etc., un valor de nivel de vida autogenerado aleatoriamente, la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado) y valores autogenerados para el resto de atributos.
• Todos los caballeros tendrán los siguientes valores: ataque 13, defensa 7, nivel de vida [10..12] (el nivel de vida actual empieza con el valor del nivel de vida).
• Todos los arqueros tendrán los siguientes valores: ataque 7, defensa 3, nivel de vida [3..5] (el nivel de vida actual empieza con el valor del nivel de vida).
• Todos los espadachines tendrán los siguientes valores: ataque 10, defensa 8, nivel de vida [8..10] (el nivel de vida actual empieza con el valor del nivel de vida).
• Todos los lanceros tendrán los siguientes valores: ataque 5, defensa 10, nivel de vida [5..8] (el nivel de vida actual empieza con el valor del nivel de vida).
• Mostrar el tablero, distinguiendo los ejércitos y los tipos de soldados creados. Además, se debe mostrar todos los datos de todos los soldados creados para ambos ejércitos. Además de los datos del soldado con mayor vida de cada ejército, el promedio de nivel de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando algún algoritmo de ordenamiento.
• Finalmente, que muestre el resumen los 2 ejércitos, indicando el reino, cantidad de unidades, distribución del ejército según las unidades, nivel de vida total del ejército y qué ejército ganó la batalla (usar la métrica de suma de niveles de vida y porcentajes de probabilidad de victoria basado en ella). Este porcentaje también debe mostrarse.
• Hacerlo programa 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

3. URL de Repositorio Github

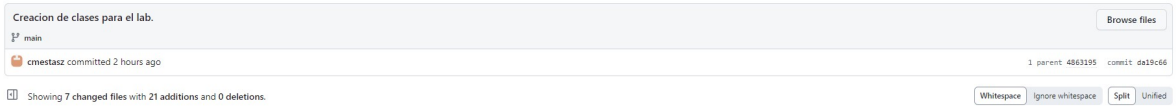
- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/cmestasz/fp2-23b.git`
- URL para el laboratorio 20 en el Repositorio GitHub.
- `https://github.com/cmestasz/fp2-23b/tree/main/fase03/lab20`

4. Actividades con el repositorio GitHub

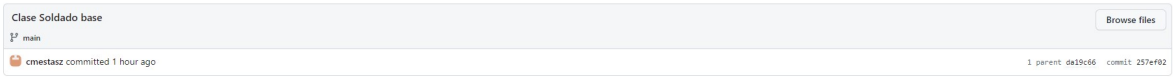
commits.bash

```
$ git add Soldado.java
$ git commit -m "Clase Soldado base"
[main 257ef02] Clase Soldado base
 1 file changed, 17 insertions(+), 1 deletion(-)

$ git add Mapa.java
$ git commit -m "Metodo inicializarSoldados"
[main 0731100] Metodo inicializarSoldados
 1 file changed, 65 insertions(+), 1 deletion(-)

$ git add Mapa.java
$ git commit -m "Metodo imprimirTablero y auxiliares"
[main 6969a01] Metodo imprimirTablero y auxiliares
 1 file changed, 51 insertions(+)

$ git add Mapa.java
$ git add Soldado.java
$ git commit -m "Metodo verificarVentaja y auxiliares"
[main c62d75b] Metodo verificarVentaja y auxiliares
 2 files changed, 88 insertions(+), 11 deletions(-)

$ git add Espadachin.java
$ git commit -m "Clase Espadachin"
[main 07daea2] Clase Espadachin
 1 file changed, 9 insertions(+)

$ git add Caballero.java
$ git add Soldado.java
$ git commit -m "Clase Caballero"
[main 6da2cef] Clase Caballero
 2 files changed, 41 insertions(+)

$ git add Arquero.java
$ git commit -m "Clase Arquero"
[main 333fdee] Clase Arquero
 1 file changed, 13 insertions(+), 1 deletion(-)

$ git add Lancero.java
$ git commit -m "Clase Lancero"
[main bdd4a61] Clase Lancero
 1 file changed, 9 insertions(+)

$ git add Mapa.java
$ git add Espadachin.java
$ git add Caballero.java
$ git add Arquero.java
$ git add Lancero.java
$ git add Soldado.java
$ git commit -m "Metodos y atributos estaticos para contar"
[main 82e7fea] Metodos y atributos estaticos para contar
 6 files changed, 82 insertions(+), 9 deletions(-)

```

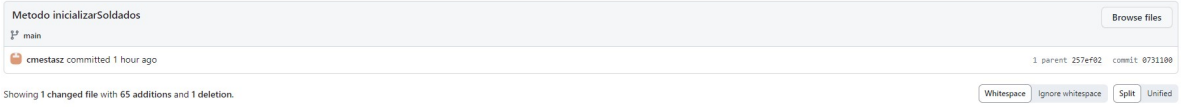
```
$ git add Mapa.java
$ git commit -m "Metodos imprimirEstado e imprimirGanador"
[main d34f34d] Metodos imprimirEstado e imprimirGanador
 1 file changed, 40 insertions(+), 7 deletions(-)

$ git push
Enumerating objects: 69, done.
Counting objects: 100% (69/69), done.
Delta compression using up to 4 threads
Compressing objects: 100% (50/50), done.
Writing objects: 100% (68/68), 7.92 KiB | 901.00 KiB/s, done.
Total 68 (delta 33), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (33/33), completed with 1 local object.
To https://github.com/cmestasz/fp2-23b.git
   4863195..d34f34d  main -> main

$ git add Videojuego.java
$ git commit -m "Juego principal con comportamiento iterativo"
[main da970e0] Juego principal con comportamiento iterativo
 1 file changed, 11 insertions(+), 1 deletion(-)

$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 608 bytes | 608.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/cmestasz/fp2-23b.git
   d34f34d..da970e0  main -> main
```



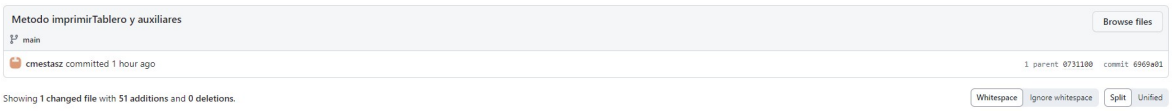
Primer Commit.



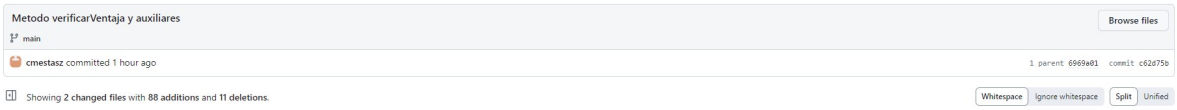
Segundo Commit.



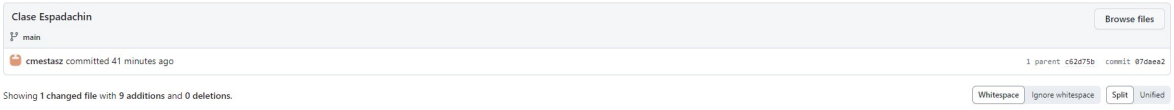
Tercer Commit.



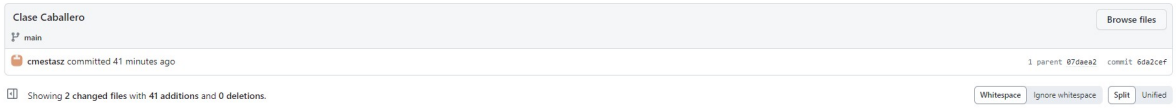
Cuarto Commit.



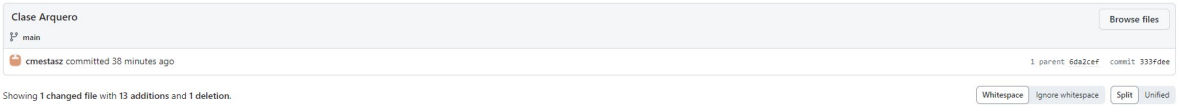
Quinto Commit.



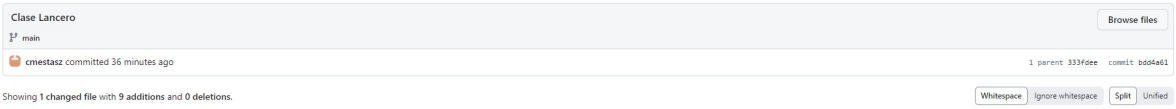
Sexto Commit.



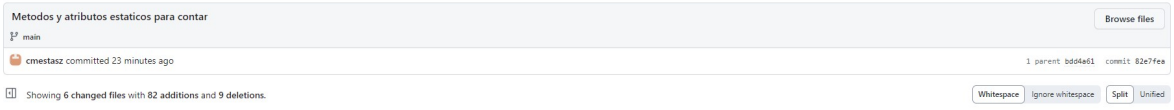
Septimo Commit.



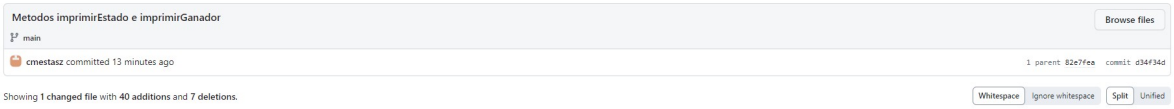
Octavo Commit.



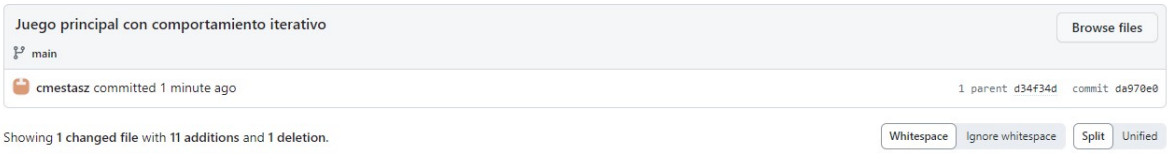
Noveno Commit.



Decimo Commit.



Decimo Primer Commit.



Decimo Segundo Commit.

5. Código desarrollado

Soldado.java

```java
package fase03.lab20;

public abstract class Soldado {
    private static int totalSoldados1;
    private static int totalSoldados2;
    private String nombre;
    private int equipo;
    private int vidaInicial;
    private int vidaActual;
    private int ataque;
    private int defensa;
    private String clase;

    public Soldado(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
         String clase) {
        this.nombre = nombre;
        this.equipo = equipo;
        this.vidaInicial = vidaInicial;
        this.vidaActual = vidaInicial;
        this.ataque = ataque;
        this.defensa = defensa;
        this.clase = clase;
        if (equipo == 1)
            totalSoldados1++;
        else
            totalSoldados2++;
    }

    public String getNombre() {
        return nombre;
    }

    public int getEquipo() {
        return equipo;
    }

    public int getVidaInicial() {
        return vidaInicial;
    }

    public int getVidaActual() {
        return vidaActual;
    }

    public int getAtaque() {
        return ataque;
    }

    public int getDefensa() {
        return defensa;
    }

```

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

- 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.

Caballero.java

```
1   package fase03.lab20;
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,
            String clase) {
10          super(nombre, equipo, vidaInicial, ataque, defensa, clase);
11          if (equipo == 1)
12              totalCaballeros1++;
13          else
14              totalCaballeros2++;
15      }
16
17      public void cambiarArma(String arma) {
18          this.arma = arma;
```

```
    }

    public void montar(Soldado otro) {
        montado = true;
        cambiarArma("LANZA");
        embestir(otro);
    }

    public void desmontar() {
        montado = false;
        cambiarArma("ESPADA");
        defender();
    }

    public void embestir(Soldado otro) {
        if (montado)
            atacar(otro, 3);
        else
            atacar(otro, 2);
    }

    public static int getTotalCaballeros1() {
        return totalCaballeros1;
    }

    public static int getTotalCaballeros2() {
        return totalCaballeros2;
    }
}
```

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

Arquero.java

```
package fase03.lab20;

public class Arquero extends Soldado {
    private static int totalArqueros1;
    private static int totalArqueros2;
    private int flechas = 10;

    public Arquero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
         String clase) {
        super(nombre, equipo, vidaInicial, ataque, defensa, clase);
        if (equipo == 1)
            totalArqueros1++;
        else
            totalArqueros2++;
    }

    public void disparar(Soldado otro) {
        if (flechas > 0) {
            atacar(otro, 1);
            flechas--;
```

```
        }
    }

    public static int getTotalArqueros1() {
        return totalArqueros1;
    }

    public static int getTotalArqueros2() {
        return totalArqueros2;
    }
}
```

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

Espadachin.java

```
package fase03.lab20;

public class Espadachin extends Soldado {
    private static int totalEspadachines1;
    private static int totalEspadachines2;
    private double longitudEspada;

    public Espadachin(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
         String clase) {
        super(nombre, equipo, vidaInicial, ataque, defensa, clase);
        if (equipo == 1)
            totalEspadachines1++;
        else
            totalEspadachines2++;
    }

    public void generarMuroEscudos() {
        System.out.println(getNombre() + " genera un muro de escudos!");
    }

    public static int getTotalEspadachines1() {
        return totalEspadachines1;
    }

    public static int getTotalEspadachines2() {
        return totalEspadachines2;
    }
}
```

- Clase que mantiene atributos y métodos independientes de un espadachin.
- Longitud de espada, generarMuroEscudos.

Lancero.java

```
package fase03.lab20;
```

```
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,
           String clase) {
9          super(nombre, equipo, vidaInicial, ataque, defensa, clase);
10         if (equipo == 1)
11             totalLanceros1++;
12         else
13             totalLanceros2++;
14     }
15
16     public void schiltrom() {
17         defender();
18     }
19
20     public static int getTotalLanceros1() {
21         return totalLanceros1;
22     }
23
24     public static int getTotalLanceros2() {
25         return totalLanceros2;
26     }
27 }
```

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

Mapa.java

```
1  package fase03.lab20;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.Random;
6
7  public class Mapa {
8      private HashMap<String, Soldado> soldados = new HashMap<String, Soldado>();
9      private ArrayList<Soldado> listaSoldados1 = new ArrayList<Soldado>();
10     private ArrayList<Soldado> listaSoldados2 = new ArrayList<Soldado>();
11     private String reino1;
12     private String reino2;
13     private String terreno;
14     private final String[] CLASES = { "CABALLERO", "ARQUERO", "ESPADACHIN", "LANCERO" };
15     private final String[] TIPOS = { "BOSQUE", "CAMPO ABIERTO", "MONTANA", "DESIERTO",
           "PLAYA" };
16     private final String[] REINOS = { "INGLATERRA", "FRANCIA", "CASTILLA-ARAGON", "MOROS",
           "SACRO IMPERIO" };
17     private final Random RANDOM = new Random();
18
19     public Mapa() {
20         inicializarSoldados(soldados, listaSoldados1, 1);
```

```
        inicializarSoldados(soldados, listaSoldados2, 2);
        imprimirTablero(soldados);
        terreno = TIPOS[RANDOM.nextInt(TIPOS.length)];
        reino1 = REINOS[RANDOM.nextInt(REINOS.length)];
        reino2 = REINOS[RANDOM.nextInt(REINOS.length)];
        System.out.println("El terreno elegido es: " + terreno);
        System.out.println("El reino 1 es: " + reino1);
        System.out.println("El reino 2 es: " + reino2);
        System.out.println();
        verificarVentaja(reino1);
        verificarVentaja(reino2);
        System.out.println();
        imprimirEstado();
        System.out.println();
        imprimirGanador();
    }

    private void inicializarSoldados(HashMap<String, Soldado> mapaSoldados,
        ArrayList<Soldado> listaSoldados,
            int equipo) {
        int cantidad = RANDOM.nextInt(10) + 1;
        for (int i = 0; i < cantidad; i++) {
            String clase = CLASES[RANDOM.nextInt(CLASES.length)];
            int vida, ataque, defensa, fila, columna;
            do {
                fila = RANDOM.nextInt(10);
                columna = RANDOM.nextInt(10);
            } while (mapaSoldados.containsKey(generarLlave(fila, columna)));
            String nombre = clase + i + "X" + equipo;
            Soldado soldado = null;
            switch (clase) {
                case "CABALLERO":
                    ataque = 13;
                    defensa = 7;
                    vida = RANDOM.nextInt(3) + 10;
                    soldado = new Caballero(nombre, equipo, vida, ataque, defensa, clase);
                    break;
                case "ARQUERO":
                    ataque = 7;
                    defensa = 3;
                    vida = RANDOM.nextInt(3) + 3;
                    soldado = new Arquero(nombre, equipo, vida, ataque, defensa, clase);
                    break;
                case "ESPADACHIN":
                    ataque = 10;
                    defensa = 8;
                    vida = RANDOM.nextInt(3) + 8;
                    soldado = new Espadachin(nombre, equipo, vida, ataque, defensa, clase);
                    break;
                case "LANCERO":
                    ataque = 5;
                    defensa = 10;
                    vida = RANDOM.nextInt(4) + 5;
                    soldado = new Lancero(nombre, equipo, vida, ataque, defensa, clase);
                    break;
            }
```

```
76          mapaSoldados.put(generarLlave(fila, columna), soldado);
77          listaSoldados.add(soldado);
78      }
79  }
80
81  private void imprimirTablero(HashMap<String, Soldado> mapaSoldados) {
82      System.out.print(generarEncabezado(mapaSoldados));
83      String separacion = generarSeparacion(mapaSoldados);
84      for (int i = 0; i < 10; i++) {
85          System.out.print(separacion);
86          System.out.print(generarFila(mapaSoldados, i));
87      }
88      System.out.print(separacion);
89  }
90
91  private String generarEncabezado(HashMap<String, Soldado> mapaSoldados) {
92      String encabezado = "\t";
93      for (int i = 0; i < 10; i++)
94          encabezado += (" " + intToChar(i + 1) + " ");
95      encabezado += " \n";
96      return encabezado;
97  }
98
99  private String generarSeparacion(HashMap<String, Soldado> mapaSoldados) {
100     String fila = "\t";
101     for (int i = 0; i < 10; i++)
102         fila += "-------";
103     fila += "-\n";
104     return fila;
105 }
106
107 private String generarFila(HashMap<String, Soldado> mapaSoldados, int f) {
108     String fila = (f + 1) + "\t";
109     for (int i = 0; i < 10; i++) {
110         fila += "| ";
111         Soldado soldado = mapaSoldados.get(generarLlave(f, i));
112         if (soldado != null) {
113             String nombre = soldado.getNombre();
114             fila += nombre.charAt(0) + nombre.substring(nombre.length() - 3);
115         } else {
116             fila += "   ";
117         }
118         fila += " ";
119     }
120     fila += "|\n";
121     return fila;
122 }
123
124 private void verificarVentaja(String reino) {
125     switch (reino) {
126         case "INGLATERRA":
127             if (terreno.equals("BOSQUE"))
128                 mejorarSoldados(reino);
129             break;
130         case "FRANCIA":
131             if (terreno.equals("CAMPO ABIERTO"))
```

```
132                 mejorarSoldados(reino);
133             break;
134         case "CASTILLA-ARAGON":
135             if (terreno.equals("MONTANA"))
136                 mejorarSoldados(reino);
137             break;
138         case "MOROS":
139             if (terreno.equals("DESIERTO"))
140                 mejorarSoldados(reino);
141             break;
142         case "SACRO IMPERIO":
143             if (terreno.equals("BOSQUE") || terreno.equals("PLAYA") ||
                    terreno.equals("CAMPO ABIERTO"))
144                 mejorarSoldados(reino);
145             break;
146     }
147 }
148
149 private void mejorarSoldados(String reino) {
150     System.out.println(reino + " tiene ventaja en el terreno!");
151     ArrayList<Soldado> soldados = reino.equals(reino1) ? listaSoldados1 : listaSoldados2;
152     for (Soldado soldado : soldados)
153         soldado.aumentarVida();
154 }
155
156 private void imprimirEstado() {
157     System.out.printf(
158             "Ejercito 1: %s%nCantidad total de soldados: %d%nEspadachines: %d%nArqueros:
                    %d%nCaballeros: %d%nLanceros: %d%n",
159             reino1, Soldado.getTotalSoldados1(), Espadachin.getTotalEspadachines1(),
                    Arquero.getTotalArqueros1(),
160             Caballero.getTotalCaballeros1(), Lancero.getTotalLanceros1());
161     System.out.println();
162     System.out.printf(
163             "Ejercito 2: %s%nCantidad total de soldados: %d%nEspadachines: %d%nArqueros:
                    %d%nCaballeros: %d%nLanceros: %d%n",
164             reino2, Soldado.getTotalSoldados2(), Espadachin.getTotalEspadachines2(),
                    Arquero.getTotalArqueros2(),
165             Caballero.getTotalCaballeros2(), Lancero.getTotalLanceros2());
166 }
167
168 private void imprimirGanador() {
169     int vida1 = vidaTotal(listaSoldados1);
170     int vida2 = vidaTotal(listaSoldados2);
171     double chance1 = 1.0 * vida1 / (vida1 + vida2);
172     double chance2 = 1 - chance1;
173     System.out.println(reino1 + ": " + vida1 + "\t\t" + chance1 * 100 + "% de
            probabilidad de victoria");
174     System.out.println(reino2 + ": " + vida2 + "\t\t" + chance2 * 100 + "% de
            probabilidad de victoria");
175     double aleatorio = RANDOM.nextDouble(1);
176     if (aleatorio < chance1)
177         System.out.println("Gana " + reino1);
178     else
179         System.out.println("Gana " + reino2);
180     System.out.println("Aleatorio generado: " + aleatorio);
```

```
    }

    private int vidaTotal(ArrayList<Soldado> soldados) {
        int suma = 0;
        for (Soldado soldado : soldados)
            suma += soldado.getVidaActual();
        return suma;
    }

    private String generarLlave(int fila, int columna) {
        return fila + "," + columna;
    }

    public static char intToChar(int n) {
        return (char) (n + 'A' - 1);
    }

    public static int charToInt(char c) {
        return (int) (c - 'A' + 1);
    }
}
```

- Ciclo de un juego contenido dentro del constructor.
- Método inicializarSoldados crea los soldados y los pone en el tablero.
- Método imprimirTablero imprime el tablero de juego.
- Método verificarVentaja otorga ventaja al reino aventajado por el terreno.
- Método imprimirEstado imprime el estado del tablero.
- Método imprimirGanador calcula el ganador de la batalla y lo imprime.

Videojuego.java

```
package fase03.lab20;

import java.util.Scanner;

public class Videojuego {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String respuesta = "S";
        while (respuesta.equalsIgnoreCase("S")) {
            new Mapa();
            System.out.print("Desea jugar de nuevo? (S/N): ");
            respuesta = sc.nextLine();
        }
    }
}
```

- Permite el comportamiento iterativo creando nuevos mapas cada iteración.

6. Ejecución del código

VideoJuego9.java

```
          A     B     C     D     E     F     G     H     I     J
        ------------------------------------------------------------------------
1       |     | COX2 |     |     | C4X1 |     |     |     |     |     |
        ------------------------------------------------------------------------
2       |     |     |     |     |     |     | L5X2 |     |     |     |
        ------------------------------------------------------------------------
3       |     |     |     |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
4       |     |     | L4X2 |     | L3X1 |     |     |     |     |     |
        ------------------------------------------------------------------------
5       |     |     |     |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
6       |     |     |     |     |     | A1X2 |     |     |     |     |
        ------------------------------------------------------------------------
7       |     |     | E2X1 | A1X1 |     | E3X2 |     |     |     |     |
        ------------------------------------------------------------------------
8       |     |     | L2X2 |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
9       | COX1 |     |     |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
10      |     |     |     |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
El terreno elegido es: DESIERTO
El reino 1 es: INGLATERRA
El reino 2 es: FRANCIA


Ejercito 1: INGLATERRA
Cantidad total de soldados: 5
Espadachines: 1
Arqueros: 1
Caballeros: 2
Lanceros: 1

Ejercito 2: FRANCIA
Cantidad total de soldados: 6
Espadachines: 1
Arqueros: 1
Caballeros: 1
Lanceros: 3

INGLATERRA: 38          44.70588235294118% de probabilidad de victoria
FRANCIA: 47             55.294117647058826% de probabilidad de victoria
Gana FRANCIA
Aleatorio generado: 0.7409087112038352
Desea jugar de nuevo? (S/N): S
          A     B     C     D     E     F     G     H     I     J
        ------------------------------------------------------------------------
1       |     |     | LOX1 |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
2       |     |     |     |     |     |     |     |     |     |     |
        ------------------------------------------------------------------------
```

```
3       |      |      |      |      |      |      |      |      |      |      |
        ------------------------------------------------------------------------
4       |      |      | A0X2 |      | C1X1 |      |      |      | C2X1 |      |
        ------------------------------------------------------------------------
5       | C3X1 |      |      |      |      |      |      |      | A7X1 |      |
        ------------------------------------------------------------------------
6       |      |      |      |      |      |      |      |      |      |      |
        ------------------------------------------------------------------------
7       | L1X2 |      |      |      |      |      |      |      |      |      |
        ------------------------------------------------------------------------
8       |      |      | E6X1 | L5X1 |      |      |      |      |      | L4X1 |
        ------------------------------------------------------------------------
9       |      |      |      |      |      |      |      |      |      |      |
        ------------------------------------------------------------------------
10      |      |      |      |      |      |      |      |      |      |      |
        ------------------------------------------------------------------------
El terreno elegido es: MONTANA
El reino 1 es: INGLATERRA
El reino 2 es: MOROS


Ejercito 1: INGLATERRA
Cantidad total de soldados: 13
Espadachines: 2
Arqueros: 2
Caballeros: 5
Lanceros: 4

Ejercito 2: MOROS
Cantidad total de soldados: 8
Espadachines: 1
Arqueros: 2
Caballeros: 1
Lanceros: 4

INGLATERRA: 70          87.5% de probabilidad de victoria
MOROS: 10               12.5% de probabilidad de victoria
Gana INGLATERRA
Aleatorio generado: 0.4358383932692418
Desea jugar de nuevo? (S/N): N
```

7. Diagrama UML

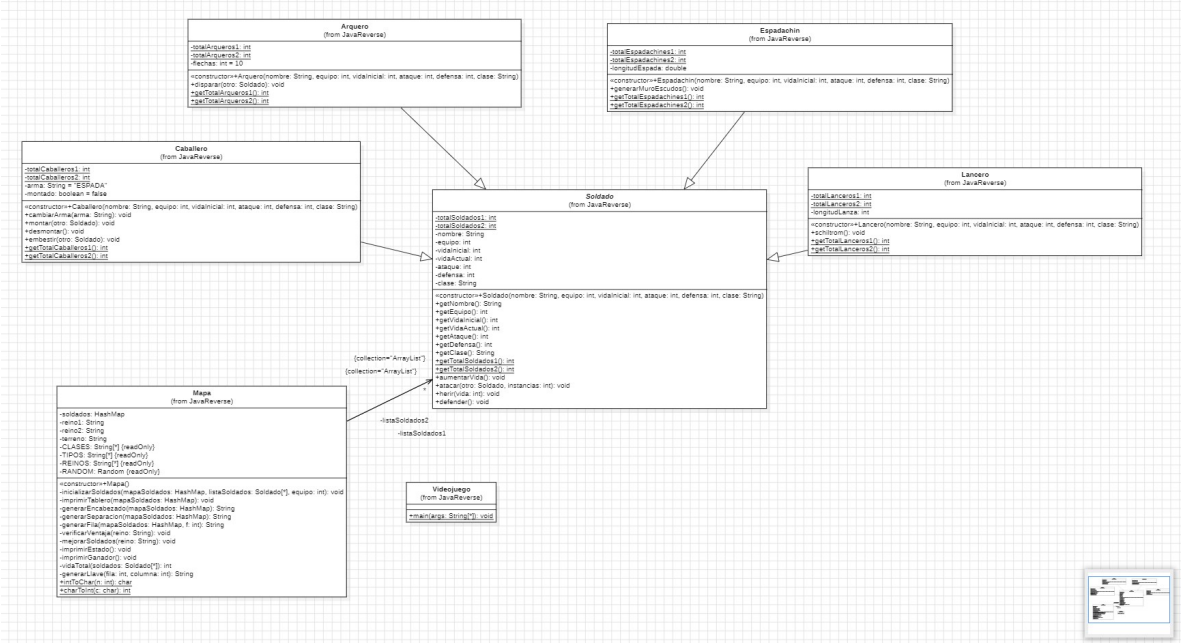


Diagrama UML.

8. Estructura de laboratorio 20

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

```
lab20/
|--- Soldado.java
|--- Caballero.java
|--- Arquero.java
|--- Espadachin.java
|--- Lancero.java
|--- Mapa.java
|--- Videojuego.java
|--- commits.bash
|--- ejec01.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
   |--- commit09.jpg
   |--- commit10.jpg
   |--- commit11.jpg
   |--- commit12.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 facil 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 cumplio 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 items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

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

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	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
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	

10. Referencias

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