

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 espada" como acción crear 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 investir). El caballero también puede investir, 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 lanzas 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

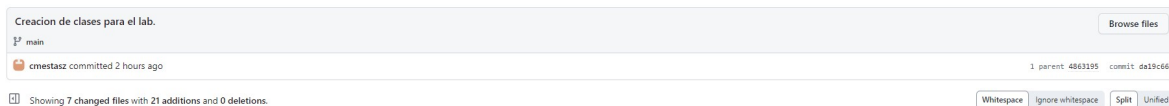
- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/cmestasz/fp2-23b.git`
- URL para el laboratorio 20 en el Repositorio GitHub.
- `https://github.com/cmestasz/fp2-23b/tree/main/fase03/lab20`

4. Actividades con el repositorio GitHub

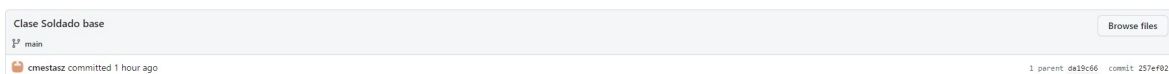
commits.bash

```
1 $ git add Soldado.java
2 $ git commit -m "Clase Soldado base"
3 [main 257ef02] Clase Soldado base
4 1 file changed, 17 insertions(+), 1 deletion(-)
5
6 $ git add Mapa.java
7 $ git commit -m "Metodo inicializarSoldados"
8 [main 0731100] Metodo inicializarSoldados
9 1 file changed, 65 insertions(+), 1 deletion(-)
10
11 $ git add Mapa.java
12 $ git commit -m "Metodo imprimirTablero y auxiliares"
13 [main 6969a01] Metodo imprimirTablero y auxiliares
14 1 file changed, 51 insertions(+)
15
16 $ git add Mapa.java
17 $ git add Soldado.java
18 $ git commit -m "Metodo verificarVentaja y auxiliares"
19 [main c62d75b] Metodo verificarVentaja y auxiliares
20 2 files changed, 88 insertions(+), 11 deletions(-)
21
22 $ git add Espadachin.java
23 $ git commit -m "Clase Espadachin"
24 [main 07daea2] Clase Espadachin
25 1 file changed, 9 insertions(+)
26
27 $ git add Caballero.java
28 $ git add Soldado.java
29 $ git commit -m "Clase Caballero"
30 [main 6da2cef] Clase Caballero
31 2 files changed, 41 insertions(+)
32
33 $ git add Arquero.java
34 $ git commit -m "Clase Arquero"
35 [main 333fdee] Clase Arquero
36 1 file changed, 13 insertions(+), 1 deletion(-)
37
38 $ git add Lancero.java
39 $ git commit -m "Clase Lancero"
40 [main bdd4a61] Clase Lancero
41 1 file changed, 9 insertions(+)
42
43 $ git add Mapa.java
44 $ git add Espadachin.java
45 $ git add Caballero.java
46 $ git add Arquero.java
47 $ git add Lancero.java
48 $ git add Soldado.java
49 $ git commit -m "Metodos y atributos estaticos para contar"
50 [main 82e7fea] Metodos y atributos estaticos para contar
51 6 files changed, 82 insertions(+), 9 deletions(-)
52
```

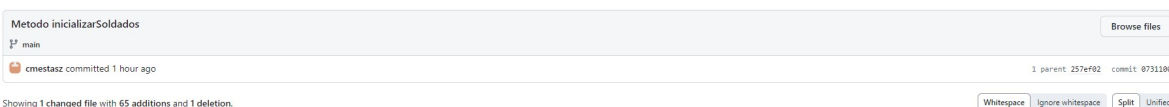
```
53 $ git add Mapa.java
54 $ git commit -m "Metodos imprimirEstado e imprimirGanador"
55 [main d34f34d] Metodos imprimirEstado e imprimirGanador
56 1 file changed, 40 insertions(+), 7 deletions(-)
57
58 $ git push
59 Enumerating objects: 69, done.
60 Counting objects: 100% (69/69), done.
61 Delta compression using up to 4 threads
62 Compressing objects: 100% (50/50), done.
63 Writing objects: 100% (68/68), 7.92 KiB | 901.00 KiB/s, done.
64 Total 68 (delta 33), reused 0 (delta 0), pack-reused 0
65 remote: Resolving deltas: 100% (33/33), completed with 1 local object.
66 To https://github.com/cmestasz/fp2-23b.git
67 4863195..d34f34d main -> main
68
69 $ git add Videojuego.java
70 $ git commit -m "Juego principal con comportamiento iterativo"
71 [main da970e0] Juego principal con comportamiento iterativo
72 1 file changed, 11 insertions(+), 1 deletion(-)
73
74 $ git push
75 Enumerating objects: 9, done.
76 Counting objects: 100% (9/9), done.
77 Delta compression using up to 4 threads
78 Compressing objects: 100% (4/4), done.
79 Writing objects: 100% (5/5), 608 bytes | 608.00 KiB/s, done.
80 Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
81 remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
82 To https://github.com/cmestasz/fp2-23b.git
83 d34f34d..da970e0 main -> main
```



Primer Commit.



Segundo Commit.



Tercer Commit.

Metodo imprimirTablero y auxiliares
main
cmestasz committed 1 hour ago
Showing 1 changed file with 51 additions and 0 deletions.

[Browse files](#)
1 parent 0731180 commit 6969e01
Whitespace Ignore whitespace Split Unified

Cuarto Commit.

Metodo verificarVentaja y auxiliares
main
cmestasz committed 1 hour ago
Showing 2 changed files with 88 additions and 11 deletions.

[Browse files](#)
1 parent 6969e01 commit c62c75b
Whitespace Ignore whitespace Split Unified

Quinto Commit.

Clase Espadachin
main
cmestasz committed 41 minutes ago
Showing 1 changed file with 9 additions and 0 deletions.

[Browse files](#)
1 parent c62c75b commit 07dae2
Whitespace Ignore whitespace Split Unified

Sexto Commit.

Clase Caballero
main
cmestasz committed 41 minutes ago
Showing 2 changed files with 41 additions and 0 deletions.

[Browse files](#)
1 parent 07dae2 commit 6da2cef
Whitespace Ignore whitespace Split Unified

Septimo Commit.

Clase Arquero
main
cmestasz committed 38 minutes ago
Showing 1 changed file with 13 additions and 1 deletion.

[Browse files](#)
1 parent 6da2cef commit 333fdee
Whitespace Ignore whitespace Split Unified

Octavo Commit.

Clase Lancero
main
cmestasz committed 36 minutes ago
Showing 1 changed file with 9 additions and 0 deletions.

[Browse files](#)
1 parent 333fdee commit bdd4e01
Whitespace Ignore whitespace Split Unified

Noveno Commit.

Metodos y atributos estaticos para contar
main
cmestasz committed 23 minutes ago
Showing 6 changed files with 82 additions and 9 deletions.

[Browse files](#)
1 parent bdd4e01 commit 82e7fee
Whitespace Ignore whitespace Split Unified

Decimo Commit.

Metodos imprimirEstado e imprimirGanador

main

cmestasz committed 13 minutes ago

1 parent 82e7fee

commit d34f34d

Showing 1 changed file with 40 additions and 7 deletions.

Whitespace

Ignore whitespace

Split

Unified

Decimo Primer Commit.

Juego principal con comportamiento iterativo

main

cmestasz committed 1 minute ago

1 parent d34f34d

commit ds970e0

Showing 1 changed file with 11 additions and 1 deletion.

Whitespace

Ignore whitespace

Split

Unified

Decimo Segundo Commit.

5. Código desarrollado

Soldado.java

```
1 public abstract class Soldado {
2     private static int totalSoldados1;
3     private static int totalSoldados2;
4     private String nombre;
5     private int equipo;
6     private int vidaInicial;
7     private int vidaActual;
8     private int ataque;
9     private int defensa;
10    private String clase;
11
12    public Soldado(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
13        String clase) {
14        this.nombre = nombre;
15        this.equipo = equipo;
16        this.vidaInicial = vidaInicial;
17        this.vidaActual = vidaInicial;
18        this.ataque = ataque;
19        this.defensa = defensa;
20        this.clase = clase;
21        if (equipo == 1)
22            totalSoldados1++;
23        else
24            totalSoldados2++;
25    }
26
27    public String getNombre() {
28        return nombre;
29    }
30
31    public int getEquipo() {
32        return equipo;
33    }
34
35    public int getVidaInicial() {
36        return vidaInicial;
37    }
38
39    public int getVidaActual() {
40        return vidaActual;
41    }
42
43    public int getAtaque() {
44        return ataque;
45    }
46
47    public int getDefensa() {
48        return defensa;
49    }
50
51    public String getClase() {
52        return clase;
53    }
54 }
```



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

- 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 public class Caballero extends Soldado {
2     private static int totalCaballeros1;
3     private static int totalCaballeros2;
4     private String arma = "ESPADA";
5     private boolean montado = false;
6
7     public Caballero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
8         String clase) {
9         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
10        if (equipo == 1)
11            totalCaballeros1++;
12        else
13            totalCaballeros2++;
14    }
15
16    public void cambiarArma(String arma) {
17        this.arma = arma;
18    }
19
20    public void montar(Soldado otro) {
21        montado = true;
22    }
23 }
```

```
21     cambiarArma("LANZA");
22     embestir(otro);
23 }
24
25 public void desmontar() {
26     montado = false;
27     cambiarArma("ESPADA");
28     defender();
29 }
30
31 public void embestir(Soldado otro) {
32     if (montado)
33         atacar(otro, 3);
34     else
35         atacar(otro, 2);
36 }
37
38 public static int getTotalCaballeros1() {
39     return totalCaballeros1;
40 }
41
42 public static int getTotalCaballeros2() {
43     return totalCaballeros2;
44 }
45 }
```

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

Arquero.java

```
1 public class Arquero extends Soldado {
2     private static int totalArqueros1;
3     private static int totalArqueros2;
4     private int flechas = 10;
5
6     public Arquero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
7         String clase) {
8         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
9         if (equipo == 1)
10             totalArqueros1++;
11         else
12             totalArqueros2++;
13     }
14
15     public void disparar(Soldado otro) {
16         if (flechas > 0) {
17             atacar(otro, 1);
18             flechas--;
19         }
20     }
21
22     public static int getTotalArqueros1() {
23         return totalArqueros1;
24     }
25 }
```

```
24
25     public static int getTotalArqueros2() {
26         return totalArqueros2;
27     }
28 }
```

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

Espadachin.java

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

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

Lancero.java

```
1 public class Lancero extends Soldado {
2     private static int totalLanceros1;
3     private static int totalLanceros2;
4     private int longitudLanza;
5
6     public Lancero(String nombre, int equipo, int vidaInicial, int ataque, int defensa,
7         String clase) {
8         super(nombre, equipo, vidaInicial, ataque, defensa, clase);
9         if (equipo == 1)
```

```
9         totalLanceros1++;
10     else
11         totalLanceros2++;
12 }
13
14 public void schiltrom() {
15     defender();
16 }
17
18 public static int getTotalLanceros1() {
19     return totalLanceros1;
20 }
21
22 public static int getTotalLanceros2() {
23     return totalLanceros2;
24 }
25 }
```

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

Mapa.java

```
1 import java.util.ArrayList;
2 import java.util.HashMap;
3 import java.util.Random;
4
5 public class Mapa {
6     private HashMap<String, Soldado> soldados = new HashMap<String, Soldado>();
7     private ArrayList<Soldado> listaSoldados1 = new ArrayList<Soldado>();
8     private ArrayList<Soldado> listaSoldados2 = new ArrayList<Soldado>();
9     private String reino1;
10    private String reino2;
11    private String terreno;
12    private final String[] CLASES = { "CABALLERO", "ARQUERO", "ESPADACHIN", "LANCERO" };
13    private final String[] TIPOS = { "BOSQUE", "CAMPO ABIERTO", "MONTANA", "DESIERTO",
14        "PLAYA" };
15    private final String[] REINOS = { "INGLATERRA", "FRANCIA", "CASTILLA-ARAGON", "MOROS",
16        "SACRO IMPERIO" };
17    private final Random RANDOM = new Random();
18
19    public Mapa() {
20        inicializarSoldados(soldados, listaSoldados1, 1);
21        inicializarSoldados(soldados, listaSoldados2, 2);
22        imprimirTablero(soldados);
23        terreno = TIPOS[RANDOM.nextInt(TIPOS.length)];
24        reino1 = REINOS[RANDOM.nextInt(REINOS.length)];
25        reino2 = REINOS[RANDOM.nextInt(REINOS.length)];
26        System.out.println("El terreno elegido es: " + terreno);
27        System.out.println("El reino 1 es: " + reino1);
28        System.out.println("El reino 2 es: " + reino2);
29        System.out.println();
30        verificarVentaja(reino1);
31        verificarVentaja(reino2);
32        System.out.println();
33    }
```

```
31     imprimirEstado();
32     System.out.println();
33     imprimirGanador();
34 }
35
36 private void inicializarSoldados(HashMap<String, Soldado> mapaSoldados,
37     ArrayList<Soldado> listaSoldados,
38     int equipo) {
39     int cantidad = RANDOM.nextInt(10) + 1;
40     for (int i = 0; i < cantidad; i++) {
41         String clase = CLASES[RANDOM.nextInt(CLASES.length)];
42         int vida, ataque, defensa, fila, columna;
43         do {
44             fila = RANDOM.nextInt(10);
45             columna = RANDOM.nextInt(10);
46         } while (mapaSoldados.containsKey(generarLlave(fila, columna)));
47         String nombre = clase + i + "X" + equipo;
48         Soldado soldado = null;
49         switch (clase) {
50             case "CABALLERO":
51                 ataque = 13;
52                 defensa = 7;
53                 vida = RANDOM.nextInt(3) + 10;
54                 soldado = new Caballero(nombre, equipo, vida, ataque, defensa, clase);
55                 break;
56             case "ARQUERO":
57                 ataque = 7;
58                 defensa = 3;
59                 vida = RANDOM.nextInt(3) + 3;
60                 soldado = new Arquero(nombre, equipo, vida, ataque, defensa, clase);
61                 break;
62             case "ESPADACHIN":
63                 ataque = 10;
64                 defensa = 8;
65                 vida = RANDOM.nextInt(3) + 8;
66                 soldado = new Espadachin(nombre, equipo, vida, ataque, defensa, clase);
67                 break;
68             case "LANCERO":
69                 ataque = 5;
70                 defensa = 10;
71                 vida = RANDOM.nextInt(4) + 5;
72                 soldado = new Lancero(nombre, equipo, vida, ataque, defensa, clase);
73                 break;
74         }
75         mapaSoldados.put(generarLlave(fila, columna), soldado);
76         listaSoldados.add(soldado);
77     }
78 }
79
80 private void imprimirTablero(HashMap<String, Soldado> mapaSoldados) {
81     System.out.print(generarEncabezado(mapaSoldados));
82     String separacion = generarSeparacion(mapaSoldados);
83     for (int i = 0; i < 10; i++) {
84         System.out.print(separacion);
85         System.out.print(generarFila(mapaSoldados, i));
86     }
```

```
86     System.out.print(separacion);
87 }
88
89 private String generarEncabezado(HashMap<String, Soldado> mapaSoldados) {
90     String encabezado = "\t";
91     for (int i = 0; i < 10; i++)
92         encabezado += (" " + intToChar(i + 1) + " ");
93     encabezado += " \n";
94     return encabezado;
95 }
96
97 private String generarSeparacion(HashMap<String, Soldado> mapaSoldados) {
98     String fila = "\t";
99     for (int i = 0; i < 10; i++)
100         fila += "-----";
101     fila += "-\n";
102     return fila;
103 }
104
105 private String generarFila(HashMap<String, Soldado> mapaSoldados, int f) {
106     String fila = (f + 1) + "\t";
107     for (int i = 0; i < 10; i++) {
108         fila += "| ";
109         Soldado soldado = mapaSoldados.get(generarLlave(f, i));
110         if (soldado != null) {
111             String nombre = soldado.getNombre();
112             fila += nombre.charAt(0) + nombre.substring(nombre.length() - 3);
113         } else {
114             fila += "   ";
115         }
116         fila += " ";
117     }
118     fila += "|\n";
119     return fila;
120 }
121
122 private void verificarVentaja(String reino) {
123     switch (reino) {
124         case "INGLATERRA":
125             if (terreno.equals("BOSQUE"))
126                 mejorarSoldados(reino);
127             break;
128         case "FRANCIA":
129             if (terreno.equals("CAMPO ABIERTO"))
130                 mejorarSoldados(reino);
131             break;
132         case "CASTILLA-ARAGON":
133             if (terreno.equals("MONTANA"))
134                 mejorarSoldados(reino);
135             break;
136         case "MOROS":
137             if (terreno.equals("DESIERTO"))
138                 mejorarSoldados(reino);
139             break;
140         case "SACRO IMPERIO":
141             if (terreno.equals("BOSQUE") || terreno.equals("PLAYA") ||
```

```
142         terreno.equals("CAMPO ABIERTO"))
143         mejorarSoldados(reino);
144         break;
145     }
146 }
147
148 private void mejorarSoldados(String reino) {
149     System.out.println(reino + " tiene ventaja en el terreno!");
150     ArrayList<Soldado> soldados = reino.equals(reino1) ? listaSoldados1 : listaSoldados2;
151     for (Soldado soldado : soldados)
152         soldado.aumentarVida();
153 }
154
155 private void imprimirEstado() {
156     System.out.printf(
157         "Ejercito 1: %s\nCantidad total de soldados: %d\nEspadachines: %d\nArqueros: %d\nCaballeros: %d\nLanceros: %d\n",
158         reino1, Soldado.getTotalSoldados1(), Espadachin.getTotalEspadachines1(),
159         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(),
165         Arquero.getTotalArqueros2(),
166         Caballero.getTotalCaballeros2(), Lancero.getTotalLanceros2());
167 }
168
169 private void imprimirGanador() {
170     int vida1 = vidaTotal(listaSoldados1);
171     int vida2 = vidaTotal(listaSoldados2);
172     double chance1 = 1.0 * vida1 / (vida1 + vida2);
173     double chance2 = 1 - chance1;
174     System.out.println(reino1 + ": " + vida1 + "\t\t" + chance1 * 100 + "% de probabilidad de victoria");
175     System.out.println(reino2 + ": " + vida2 + "\t\t" + chance2 * 100 + "% de probabilidad de victoria");
176     double aleatorio = RANDOM.nextDouble(1);
177     if (aleatorio < chance1)
178         System.out.println("Gana " + reino1);
179     else
180         System.out.println("Gana " + reino2);
181     System.out.println("Aleatorio generado: " + aleatorio);
182 }
183
184 private int vidaTotal(ArrayList<Soldado> soldados) {
185     int suma = 0;
186     for (Soldado soldado : soldados)
187         suma += soldado.getVidaActual();
188     return suma;
189 }
190
191 private String generarLlave(int fila, int columna) {
192     return fila + "," + columna;
193 }
```

```
191
192 public static char intToChar(int n) {
193     return (char) (n + 'A' - 1);
194 }
195
196 public static int charToInt(char c) {
197     return (int) (c - 'A' + 1);
198 }
199 }
```

- 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

```
1 import java.util.Scanner;
2
3 public class Videojuego {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         String respuesta = "S";
7         while (respuesta.equalsIgnoreCase("S")) {
8             new Mapa();
9             System.out.print("Desea jugar de nuevo? (S/N): ");
10            respuesta = sc.nextLine();
11        }
12    }
13 }
```

- Permite el comportamiento iterativo creando nuevos mapas cada iteración.

6. Ejecución del código

```

                                VideoJuego9.java
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
                                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  |      |      |      |      |      |      |      |      |      |      |
-----

```

```

53 3 | | | | | | | | | | | |
54 -----
55 4 | | | AOX2 | | C1X1 | | | | C2X1 | |
56 -----
57 5 | C3X1 | | | | | | | | A7X1 | |
58 -----
59 6 | | | | | | | | | | | |
60 -----
61 7 | L1X2 | | | | | | | | | | |
62 -----
63 8 | | | E6X1 | L5X1 | | | | | | L4X1 |
64 -----
65 9 | | | | | | | | | | | |
66 -----
67 10 | | | | | | | | | | | |
68 -----
69 El terreno elegido es: MONTANA
70 El reino 1 es: INGLATERRA
71 El reino 2 es: MOROS
72
73
74 Ejercito 1: INGLATERRA
75 Cantidad total de soldados: 13
76 Espadachines: 2
77 Arqueros: 2
78 Caballeros: 5
79 Lanceros: 4
80
81 Ejercito 2: MOROS
82 Cantidad total de soldados: 8
83 Espadachines: 1
84 Arqueros: 2
85 Caballeros: 1
86 Lanceros: 4
87
88 INGLATERRA: 70 87.5% de probabilidad de victoria
89 MOROS: 10 12.5% de probabilidad de victoria
90 Gana INGLATERRA
91 Aleatorio generado: 0.4358383932692418
92 Desea jugar de nuevo? (S/N): N

```

7. Diagrama UML

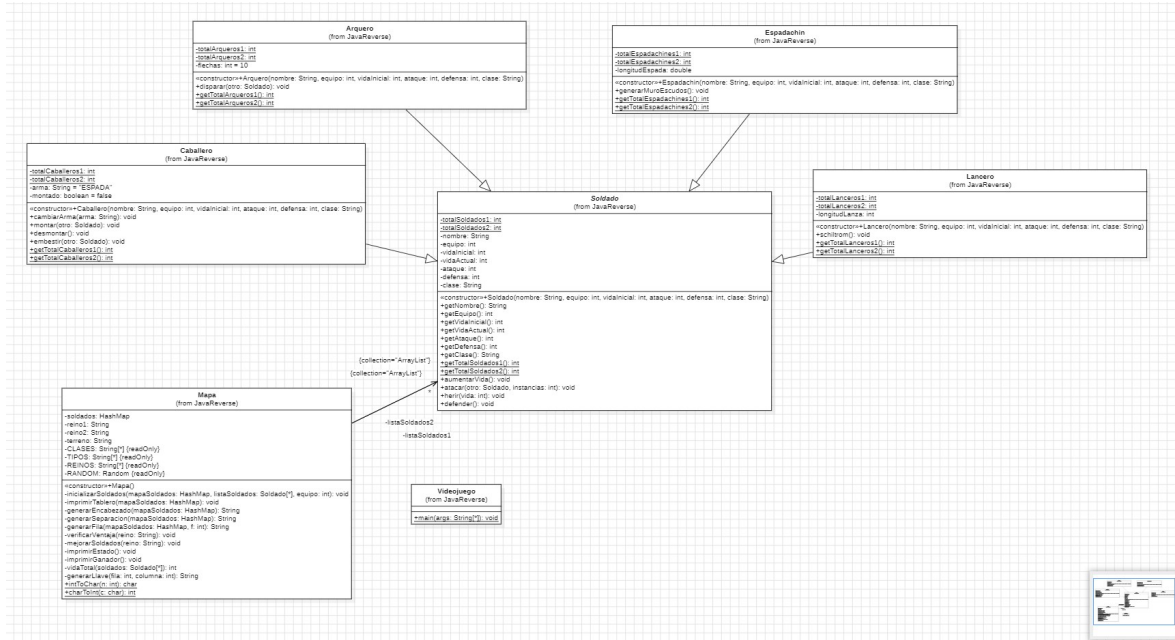


Diagrama UML.

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
```

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

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	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.