	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la programación 2</i>				
TÍTULO DE LA PRÁCTICA:	<i>Definición de Clases de Usuario</i> <i>Clase Ejército – Soldado – Mapa</i>				
NÚMERO DE PRÁCTICA:	<i>16</i>	AÑO LECTIVO:	<i>2023</i>	NRO. SEMESTRE:	<i>2do Semestre</i>
FECHA DE PRESENTACIÓN	<i>27/12/2023</i>	HORA DE PRESENTACIÓN	<i>19/20/00</i>		
INTEGRANTE (s) <i>Juan Diego Gutiérrez Ccama</i>				NOTA (0-20)	<i>Nota colocada por el docente</i>
DOCENTE(s): <i>Linno Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
I. EJERCICIOS RESUELTOS:

```

1 public class Soldado {
2     private String nombre;
3     private int nivelAtaque;
4     private int nivelDefensa;
5     private int vidaActual;
6     private int velocidad;
7     private String actitud;
8     private boolean vive;
9     private int fila;
10    private int columna;
11    private int nivelVida;
12
13    public void setNombre(String n){
14        nombre = n;
15    }
16    public void setNivelAtaque(int ataque){
17        nivelAtaque = ataque;
18    }
19    public void setNivelDefensa(int defensa){
20        nivelDefensa = defensa;
21    }
22    public void setVidaActual(int vidaAc) {
23        vidaActual = vidaAc;
24    }
25    public void setFila(int f){
26        fila = f;
27    }
28    public void setColumna(int c){
29        columna = c;
30    }
31    public void setNivelVida(int vida){
32        nivelVida = vida;
33    }
34    public void setVelocidad(int veloci){
35        velocidad = veloci;
36    }
37    public void setActitud(String ac) {
38        actitud = ac;
39    }
40    public String getNombre(){
41        return nombre;
42    }
43    public int getFila(){
44        return fila;
45    }
46    public int getColumna(){
47        return columna;
48    }
49    public int getNivelVida(){
50        return nivelVida;
51    }
52    public int getNivelAtaque(){
53        return nivelAtaque;
54    }
55    public int getNivelDefensa(){
56        return nivelDefensa;
57    }
58    public int getVelocidad(){
59        return velocidad;
60    }
61    public int getVidaActual() {
62        return vidaActual;
63    }
64    public void atacar() {
65        avanzar();
66        actitud = "ofensiva";
67    }
68    public void defender() {
69        velocidad = 0;
70        actitud = "defensiva";
71    }
72    public void avanzar() {
73        velocidad++;
74    }
75    public void retroceder() {
76        if (velocidad > 0) {
77            velocidad = 0;
78            actitud = "defensiva";
79        } else {
80            velocidad--;
81        }
82    }
83    public void serAtacado(int puntosDaño) {
84        vidaActual -= puntosDaño;
85        if (vidaActual <= 0) {

```

```

86         morir();
87     }
88 }
89 public void huir() {
90     velocidad += 2;
91     actitud = "fuga";
92 }
93 public void morir() {
94     vive = false;
95 }
96 }

1 import java.util.ArrayList;
2 public class Ejercito {
3     private String nombreReino;
4     private ArrayList<Soldado> misSoldados;
5     private int fila;
6     private int columna;
7     private int numeroSoldados;
8     private int sumaVidaTotal;
9
10 public Ejercito() {
11     misSoldados = new ArrayList<>();
12     numeroSoldados = 0;
13     sumaVidaTotal = 0;
14 }
15 public void crearEjercito() {
16     misSoldados = new ArrayList<>();
17     numeroSoldados = 0;
18     sumaVidaTotal = 0;
19     int numSoldados = (int)(Math.random() * 10) + 1;
20     for (int i = 0; i < numSoldados; i++) {
21         Soldado soldado = new Soldado();
22         soldado.setNombre("Soldado " + (i + 1));
23         soldado.setNivelAtaque((int)(Math.random() * 5) + 1);
24         soldado.setNivelDefensa((int)(Math.random() * 5) + 1);
25         soldado.setVidaActual((int)(Math.random() * 5) + 1);
26         soldado.setVelocidad((int)(Math.random() * 5) + 1);
27         soldado.setActitud("ofensiva");
28         soldado.setFila((int)(Math.random() * 10) + 1);
29         soldado.setColumna((int)(Math.random() * 10) + 1);
30         soldado.setNivelVida((int)(Math.random() * 5) + 1);
31         misSoldados.add(soldado);
32         numeroSoldados++;
33         sumaVidaTotal += soldado.getNivelVida();
34     }
35 }
36 public void agregarSoldado(Soldado soldado) {
37     if (misSoldados.size() < 10) {
38         misSoldados.add(soldado);
39     } else {
40         System.out.println("El ejército está completo. No se pueden agregar más soldados.");
41     }
42 }
43 public int vidaTotalEjercito() {

```

```

44     int vidaTotal = 0;
45     for (Soldado soldado : misSoldados) {
46         vidaTotal += soldado.getNivelVida();
47     }
48     return vidaTotal;
49 }
50 public ArrayList<Soldado> getMisSoldados() {
51     return misSoldados;
52 }
53 public void setFila(int f) {
54     fila = f;
55 }
56
57 public void setColumna(int c) {
58     columna = c;
59 }
60
61 public int getFila() {
62     return fila;
63 }
64
65 public int getColumna() {
66     return columna;
67 }
68 public void setNombreReino(String nombreReino) {
69     this.nombreReino = nombreReino;
70 }
71 public String getNombreReino() {
72     return nombreReino;
73 }
74 public int getNumeroSoldados() {
75     return numeroSoldados;
76 }
77
78 public int getSumaVidaTotal() {
79     return sumaVidaTotal;
80 }
81 }

1 import java.util.ArrayList;
2 public class Mapa {
3     private String tipoTerritorio;
4     private String[][] tablero;
5     private ArrayList<Ejercito> reinoA;
6     private ArrayList<Ejercito> reinoB;
7     private String[] nombresReinos = {"Inglaterra", "Francia", "Sacro", "Castilla", "Aragon", "Moros"};
8     private String[] tiposTerritorio = {"bosque", "campo abierto", "montaña", "desierto", "playa"};
9
10    public Mapa() {
11        this.tablero = new String[10][10];
12        reinoA = new ArrayList<>();
13        reinoB = new ArrayList<>();
14    }
15    public void iniciarJuego() {
16        tipoTerritorio = tiposTerritorio[(int) (Math.random() * tiposTerritorio.length)];
17        System.out.println("El tipo de territorio es: " + this.tipoTerritorio);
18        System.out.println();
19        String reino1 = nombresReinos[(int) (Math.random() * nombresReinos.length)];
20        String reino2;
21        do {
22            reino2 = nombresReinos[(int) (Math.random() * nombresReinos.length)];
23        } while (reino2.equals(reino1));
24
25        crearEjercito(reinoA, reino1, tablero);
26        crearEjercito(reinoB, reino2, tablero);
27        bonificacion(reinoA);
28        bonificacion(reinoB);
29        imprimirTablero(tablero, reinoA, reinoB);
30        Ejercito ganadorSoldados = determinarGanadorPorCantidadSoldados();
31        if (ganadorSoldados != null) {
32            System.out.println("El ganador por cantidad de soldados es: " + ganadorSoldados.getNombreReino());
33        } else {
34            System.out.println("Empate en la cantidad de soldados.");
35        }
36        Ejercito ganadorVida = determinarGanadorPorVidaSoldados();
37        if (ganadorVida != null) {
38            System.out.println("El ganador por la vida de los soldados es: " + ganadorVida.getNombreReino());
39        } else {
40            System.out.println("Empate en la vida de los soldados.");
41        }
42        Ejercito ganadorEjercitos = determinarGanadorPorNumeroEjercitos();
43        if (ganadorEjercitos != null) {

```

```

44         System.out.println("El ganador por número de ejércitos es: " + ganadorEjercitos.getNombreReino());
45     } else {
46         System.out.println("Empate en el número de ejércitos.");
47     }
48 }
49 public void bonificacion(ArrayList<Ejercito> ejercitos) {
50     for (Ejercito ejercito : ejercitos) {
51         for (Soldado soldado : ejercito.getMisSoldados()) {
52             String nombreReino = ejercito.getNombreReino();
53             if (tipoTerritorio.equals("bosque") && (nombreReino.equals("Inglaterra") || nombreReino.equals("Sacro"))) {
54                 soldado.setNivelVida(soldado.getNivelVida() + 1);
55             } else if (tipoTerritorio.equals("campo abierto") && (nombreReino.equals("Francia") || nombreReino.equals("Sacro"))) {
56                 soldado.setNivelVida(soldado.getNivelVida() + 1);
57             } else if (tipoTerritorio.equals("montaña") && nombreReino.equals("Castilla") || nombreReino.equals("Aragon")) {
58                 soldado.setNivelVida(soldado.getNivelVida() + 1);
59             } else if (tipoTerritorio.equals("desierto") && nombreReino.equals("Moros")) {
60                 soldado.setNivelVida(soldado.getNivelVida() + 1);
61             } else if (tipoTerritorio.equals("playa") && nombreReino.equals("Sacro")) {
62                 soldado.setNivelVida(soldado.getNivelVida() + 1);
63             }
64         }
65     }
66 }
67 public static void crearEjercito(ArrayList<Ejercito> reinos, String nombreReino, String[][] tablero) {
68     int numEjercitos = (int) (Math.random() * 10) + 1;
69     for (int ejercitoId = 1; ejercitoId <= numEjercitos; ejercitoId++) {
70         Ejercito ejercito = new Ejercito();
71         ejercito.crearEjercito();
72         ejercito.setNombreReino(nombreReino);
73         reinos.add(ejercito);
74         int fila;
75         int columna;
76         boolean posicionValida = false;
77         while (!posicionValida) {
78             fila = (int) (Math.random() * tablero.length);
79             columna = (int) (Math.random() * tablero[0].length);
80             if (tablero[fila][columna] == null) {
81                 tablero[fila][columna] = nombreReino + " " + ejercitoId;
82                 posicionValida = true;
83             }
84         }
85     }
86 }
87 public void imprimirTablero(String[][] tablero, ArrayList<Ejercito> reinoA, ArrayList<Ejercito> reinoB) {
88     for (char columna = 'A'; columna <= 'J'; columna++) {
89         System.out.print("    " + columna);
90     }
91     System.out.println();
92     System.out.println("    _____");
93     for (int i = 0; i < tablero.length; i++) {
94         System.out.print((i + 1 < 10 ? " " : "") + (i + 1));
95         for (int j = 0; j < tablero[i].length; j++) {
96             if (tablero[i][j] != null) {
97                 String reinoEjercito = tablero[i][j];
98                 String[] partes = reinoEjercito.split(" ");
99                 String nombreReino = partes[0];
100                 int indiceEjercito = Integer.parseInt(partes[1]);
101                 Ejercito ejercitoEncontrado = null;
102                 for (Ejercito ejercito : reinoA) {
103                     if (ejercito.getNombreReino().equals(nombreReino) && reinoA.indexOf(ejercito) == indiceEjercito) {
104                         ejercitoEncontrado = ejercito;
105                         break;
106                     }
107                 }
108                 if (ejercitoEncontrado == null) {
109                     for (Ejercito ejercito : reinoB) {
110                         if (ejercito.getNombreReino().equals(nombreReino) && reinoB.indexOf(ejercito) == indiceEjercito) {
111                             ejercitoEncontrado = ejercito;
112                             break;
113                         }
114                     }
115                 }
116                 if (ejercitoEncontrado != null) {
117                     int totalSoldados = ejercitoEncontrado.getNumeroSoldados();
118                     int vidaTotal = ejercitoEncontrado.getSumaVidaTotal();
119                     String info = totalSoldados + "-" + vidaTotal + "-" + nombreReino.charAt(0);
120                     System.out.printf("%1$-7s", info);
121                 } else {
122                     System.out.print("|_____");
123                 }
124             } else {
125                 System.out.print("|_____");
126             }
127         }
128         System.out.println("|");

```

```

129     }
130 }
131 public Ejercito determinarGanadorPorNumeroEjercitos() {
132     if (reinoA.size() > reinoB.size()) {
133         return reinoA.get(0);
134     } else if (reinoB.size() > reinoA.size()) {
135         return reinoB.get(0);
136     } else {
137         return null;
138     }
139 }
140
141 public Ejercito determinarGanadorPorCantidadSoldados() {
142     int soldadosReinoA = contarSoldados(reinoA);
143     int soldadosReinoB = contarSoldados(reinoB);
144
145     if (soldadosReinoA > soldadosReinoB) {
146         return reinoA.get(0);
147     } else if (soldadosReinoB > soldadosReinoA) {
148         return reinoB.get(0);
149     } else {
150         return null;
151     }
152 }
153
154 public Ejercito determinarGanadorPorVidaSoldados() {
155     int vidaReinoA = sumarVidaSoldados(reinoA);
156     int vidaReinoB = sumarVidaSoldados(reinoB);
157
158     if (vidaReinoA > vidaReinoB) {
159         return reinoA.get(0);
160     } else if (vidaReinoB > vidaReinoA) {
161         return reinoB.get(0);
162     } else {
163         return null;
164     }
165 }
166 public int contarSoldados(ArrayList<Ejercito> ejercitos) {
167     int totalSoldados = 0;
168     for (Ejercito ejercito : ejercitos) {
169         totalSoldados += ejercito.getNumeroSoldados();
170     }
171     return totalSoldados;
172 }
173
174 public int sumarVidaSoldados(ArrayList<Ejercito> ejercitos) {
175     int vidaTotal = 0;
176     for (Ejercito ejercito : ejercitos) {
177         vidaTotal += ejercito.getSumaVidaTotal();
178     }
179     return vidaTotal;
180 }
181 }

```

Ejercito.java × Mapa.java Soldado.java VideoJuego.java ×

```

1 public class VideoJuego {
2     public static void main(String[] args) {
3         Mapa mapa = new Mapa();
4         mapa.iniciarJuego();
5     }
6 }

```

II. PRUEBA

Console x

<terminated> VideoJuego (2) [Java Application] C:\Users\UJKMjuandi\Downloads\eclipse-java-2023-03-R-win32-x86_64\eclipse\plugins\org.ec

El tipo de territorio es: campo abierto

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										10-30-M
5						9-32-C			5-15-C	
6										
7										
8							4-9-M			
9										
10		6-13-C		5-16-M	1-3-C		10-28-C		4-7-M	

El ganador por cantidad de soldados es: Castilla
 El ganador por la vida de los soldados es: Castilla
 El ganador por número de ejércitos es: Castilla

III. CUESTIONARIO:

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

REPOSITORIO: <https://github.com/UJKMjuandi/FundamentosP2>

CONCLUSIONES

La realización de este proyecto de programación basado en los laboratorios anteriores ha sido una experiencia enriquecedora que ha permitido aplicar diversos conceptos y habilidades adquiridas a lo largo del curso. La creación de la clase Mapa representa un paso más allá en la complejidad del sistema, ya que ahora no solo se manejan soldados individuales, sino que se posicionan ejércitos enteros en un tablero con ciertas restricciones.

La generación aleatoria de la cantidad de ejércitos y soldados, así como la asignación de territorios específicos a cada reino, ha añadido un elemento de realismo y variabilidad al juego. La inclusión de bonificaciones según el tipo de territorio introduce una capa estratégica adicional, ya que las decisiones tácticas ahora no solo se basan en la cantidad de soldados y su nivel de vida, sino también en la ubicación geográfica de los ejércitos.

El diagrama de clases UML completo proporciona una visión clara y estructurada de la relación entre las diversas clases del sistema, facilitando la comprensión y el mantenimiento del código. La modularidad de las clases como Soldado y Ejercito contribuye a un diseño limpio y fácil de mantener.

El dibujo del mapa, que representa gráficamente la disposición de los ejércitos en el tablero, agrega una dimensión visual al juego, permitiendo una mejor comprensión de la situación estratégica. La limitación de un solo ejército por cuadrado en el tablero añade un desafío táctico al juego, ya que los jugadores deben tomar decisiones cuidadosas sobre la ubicación de sus fuerzas.


En cuanto a la iteración del juego, la capacidad de repetir la creación del mapa y realizar múltiples simulaciones de guerra proporciona una forma de analizar resultados variados. La introducción de métricas como la cantidad total de soldados, la vida total de los soldados y el número de ejércitos permite evaluar diferentes aspectos de la guerra y determinar un ganador.

Al evaluar quién ganaría la guerra, se observa que la decisión no es trivial y depende de las métricas elegidas. Si se considera la cantidad total de soldados, el reino con más soldados puede tener ventaja. Si se evalúa la vida total de los soldados, la estrategia de bonificación por territorio puede influir significativamente en el resultado. Además, si se tiene en cuenta el número de ejércitos, la distribución táctica en el mapa podría ser determinante.

METODOLOGÍA DE TRABAJO

1.- Leer los enunciados cuidadosamente para saber que requiere el problema.

2.- Seguir el diagrama uml como base que se da en la practica

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 9

3.-Empezar a hacer la ejercito,la clase soldado ,la clase VideoJuego y la clase Mapa

4.- Realizar el Videojuego con los métodos apropiados.

REFERENCIAS Y BIBLIOGRAFÍA

[1] M. A. Lopez, E. Castro Gutierrez, *Fundamentos de la Programación 2 Topics de Programación Orientada a Objetos*. Arequipa: UNSA, 2021

RUBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.


Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Tabla 1: 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

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	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta for-	2	X	2	

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

	mulada en la tarea. (El profesor puede preguntar para refrendar calificación).				
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
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	3	
TOTAL		20		16	

Tabla 2: Rúbrica para contenido del Informe y demostración