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

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	<i>Fundamentos de la programación 2</i>				
<b>TÍTULO DE LA PRÁCTICA:</b>	<i>Definición de Clases de Usuario Estratégico y Táctico - Mecanismos de Agregación y Composición</i>				
<b>NÚMERO DE PRÁCTICA:</b>	<i>14</i>	<b>AÑO LECTIVO:</b>	<i>2023</i>	<b>NRO. SEMESTRE:</b>	<i>2do Semestre</i>
<b>FECHA DE PRESENTACIÓN</b>	<i>27/12/2023</i>	<b>HORA DE PRESENTACIÓN</b>	<i>16/20/00</i>		
<b>INTEGRANTE (s)</b> <i>Juan Diego Gutiérrez Ccama</i>				<b>NOTA (0-20)</b>	<i>Nota colocada por el docente</i>
<b>DOCENTE(s):</b> <i>Linno Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<b>I. EJERCICIOS RESUELTOS:</b>

Soldado.java × VideoJuego.java Soldado.java × VideoJuego.java

```
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 atacar() {
14        avanzar();
15        actitud = "ofensiva";
16    }
17    public void defender() {
18        velocidad = 0;
19        actitud = "defensiva";
20    }
21    public void avanzar() {
22        velocidad++;
23    }
24    public void retroceder() {
25        if (velocidad > 0) {
26            velocidad = 0;
27            actitud = "defensiva";
28        } else {
29            velocidad--;
30        }
31    }
32    public void serAtacado(int puntosDaño) {
33        vidaActual -= puntosDaño;
34        if (vidaActual <= 0) {
35            morir();
36        }
37    }
38    public void huir() {
39        velocidad += 2;
40        actitud = "fuga";
41    }
42    public void morir() {
43        vive = false;
```

```

44 }
45 //Zona de sets
46 public void setNombre(String n){
47     nombre = n;
48 }
49 public void setNivelAtaque(){
50     int ataque = (int) (Math.random() * 5 + 1 );
51     nivelAtaque = ataque;
52 }
53 public void setNivelDefensa(){
54     int defensa = (int) (Math.random() * 5 + 1 );
55     nivelDefensa = defensa;
56 }
57 public void setVidaActual(int vidaAc) {
58     vidaActual = vidaAc;
59 }
60 public void setFila(int f){
61     fila = f;
62 }
63 public void setColumna(int c){
64     columna = c;
65 }
66 public void setNivelVida(int vida){
67     nivelVida = vida;
68 }
69 public void setVelocidad(){
70     int veloci = (int) (Math.random() * 5 + 1 );
71     velocidad = veloci;
72 }
73 //Zona de gets
74 public String getNombre(){
75     return nombre;
76 }
77 public int getFila(){
78     return fila;
79 }
80 public int getColumna(){
81     return columna;
82 }
83 public int getNivelVida(){
84     return nivelVida;
85 }
86 public int getNivelAtaque(){
87     return nivelAtaque;
88 }
89 public int getNivelDefensa(){
90     return nivelDefensa;
91 }
92 public int getVelocidad(){
93     return velocidad;
94 }
95 public int getVidaActual() {
96     return vidaActual;
97 }
98 }

```

```

1 import java.util.*;
2 import java.util.ArrayList;
3 public class VideoJuego {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         String[][] tablero = new String[10][10];
7         ArrayList<ArrayList<Soldado>> reinos = new ArrayList<>();
8         String[] nombresReinos = {"Inglaterra", "Francia", "Sacro", "Castilla", "Italia", "Rusia"};
9         String reino1 = nombresReinos[(int)(Math.random() * nombresReinos.length)];
10        String reino2;
11        do {
12            reino2 = nombresReinos[(int)(Math.random() * nombresReinos.length)];
13        } while (reino2.equals(reino1));
14        crearEjercito(reinos, reino1, tablero);
15        crearEjercito(reinos, reino2, tablero);
16        int turno = (int)(Math.random() * 2) + 1;
17        boolean juegoActivo = true;
18        while (juegoActivo) {
19            imprimirTablero(tablero, reinos);
20            String jugadorActual = (turno == 1) ? reino1 : reino2;
21            System.out.println("Turno del Jugador " + turno + " (" + jugadorActual + ")");
22            moverEjercito(sc, tablero, reinos);
23            if (reinos.get(0).isEmpty() || reinos.get(1).isEmpty()) {
24                juegoActivo = false;
25                String ganador = (reinos.get(0).isEmpty()) ? reino2 : reino1;
26                System.out.println(ganador + " ha ganado el juego.");
27            }
28            turno = (turno == 1) ? 2 : 1;
29        }
30    }
31}
32 public static void crearEjercito(ArrayList<ArrayList<Soldado>> reinos, String nombreReino, String[][] tablero) {
33     int numEjercitos = (int)(Math.random() * 10) + 1;
34     for (int ejercitoId = 1; ejercitoId <= numEjercitos; ejercitoId++) {
35         ArrayList<Soldado> ejercito = new ArrayList<>();
36         for (int i = 0; i < (int)(Math.random() * 10) + 1; i++) {
37             int f = (int)(Math.random() * 10) + 1;
38             int c = (int)(Math.random() * 10) + 1;
39             int x = (int)(Math.random() * 5) + 1;
40             String nombreSoldado = "soldado" + i + "X" + nombreReino;
41             Soldado soldado = new Soldado();
42             soldado.setNombre(nombreSoldado);
43             soldado.setNivelAtaque(f);
44             soldado.setFila(f);
45             soldado.setColumna(c);
46             soldado.setNivelDefensa(x);
47             soldado.setNivelVida(x);
48             soldado.setVelocidad(x);
49             ejercito.add(soldado);
50         }
51         reinos.add(ejercito);
52         int fila;
53         int columna;
54         do {
55             fila = (int)(Math.random() * tablero.length);
56             columna = (int)(Math.random() * tablero[0].length);
57         } while (tablero[fila][columna] != null);
58         tablero[fila][columna] = nombreReino + ejercitoId;
59     }
60 }
61 public static void imprimirTablero(String[][] tablero, ArrayList<ArrayList<Soldado>> reinos) {
62     System.out.print(" ");
63     for (char columna = 'A'; columna <= 'J'; columna++) {
64         System.out.print(" " + columna);
65     }
66     System.out.println();
67     System.out.println(" _ _ _ _ _ ");
68     for (int i = 0; i < tablero.length; i++) {
69         System.out.print((i + 1 < 10 ? " " : "") + (i + 1));
70         for (int j = 0; j < tablero[i].length; j++) {
71             if (tablero[i][j] != null) {
72                 String reinoSoldado = tablero[i][j];
73                 char letra = reinoSoldado.charAt(0);
74                 System.out.print("|" + letra);
75             } else {
76                 System.out.print("|_");
77             }
78         }
79         System.out.println("|");
80     }
81 }
82 public static void moverSoldado(Scanner sc, Soldado[][] tablero, ArrayList<Soldado> ejercitoMov, ArrayList<Soldado> ejercitoOtro) {
83     System.out.print("Ingrese la coordenada del soldado a mover (por ejemplo, d3): ");
84     String coordenada = sc.next();
85     int filaMov = Integer.parseInt(coordenada.substring(1)) - 1;
86     char columnaChar = coordenada.charAt(0);

```

```

87     int columnaMov = columnaChar - 'a';
88     if (filaMov < 0 || filaMov >= tablero.length || columnaMov < 0 || columnaMov >= tablero[0].length) {
89         System.out.println("Movimiento inválido: coordenada fuera del tablero.");
90         return;
91     }
92     Soldado soldado = tablero[filaMov][columnaMov];
93     if (soldado == null || !ejercitoMov.contains(soldado)) {
94         System.out.println("Movimiento inválido: no hay un soldado del ejército correspondiente en esa posición.");
95         return;
96     }
97     System.out.print("Ingrese la dirección del movimiento (w: arriba, a: izquierda, s: abajo, d: derecha): ");
98     char direccion = sc.next().charAt(0);
99     int nuevaFila = filaMov;
100    int nuevaColumna = columnaMov;
101    if (direccion == 'w') {
102        nuevaFila--;
103    } else if (direccion == 'a') {
104        nuevaColumna--;
105    } else if (direccion == 's') {
106        nuevaFila++;
107    } else if (direccion == 'd') {
108        nuevaColumna++;
109    } else {
110        System.out.println("Movimiento inválido: dirección no válida.");
111        return;
112    }
113    if (nuevaFila < 0 || nuevaFila >= tablero.length || nuevaColumna < 0 || nuevaColumna >= tablero[0].length) {
114        System.out.println("Movimiento inválido: movimiento fuera del tablero.");
115        return;
116    }
117    if (tablero[nuevaFila][nuevaColumna] != null) {
118        Soldado soldadoEnNuevaPos = tablero[nuevaFila][nuevaColumna];
119        if (ejercitoOtro.contains(soldadoEnNuevaPos)) {
120            double probabilidadSoldado = soldado.getNivelVida() * 100.0 / (soldado.getNivelVida() + soldadoEnNuevaPos.getNivelVida());
121            double probabilidadEnemigo = soldadoEnNuevaPos.getNivelVida() * 100.0 / (soldado.getNivelVida() + soldadoEnNuevaPos.getNivelVida());
122            Soldado ganadorBatalla = batalla(soldado, soldadoEnNuevaPos);
123            System.out.println("¡Batalla!");
124            System.out.println(soldado.getNombre() + " - Vida: " + soldado.getNivelVida() + " - Probabilidad de victoria: " + probabilidadSoldado + "%");
125            System.out.println(soldadoEnNuevaPos.getNombre() + " - Vida: " + soldadoEnNuevaPos.getNivelVida() + " - Probabilidad de victoria: " + probabilidadEnemigo + "%");
126
127            if (ganadorBatalla == soldado) {
128                tablero[nuevaFila][nuevaColumna] = soldado;
129                tablero[filaMov][columnaMov] = null;
130                ejercitoOtro.remove(soldadoEnNuevaPos);
131                soldado.setNivelVida(soldado.getNivelVida() + 1);
132            } else {
133                tablero[filaMov][columnaMov] = null;
134                ejercitoMov.remove(soldado);
135                soldadoEnNuevaPos.setNivelVida(soldadoEnNuevaPos.getNivelVida() + 1);
136            }
137            System.out.println("Ganador: " + ganadorBatalla.getNombre() + " - Vida: " + ganadorBatalla.getNivelVida());
138            if (ejercitoOtro.isEmpty()) {
139                System.out.println("¡El ejército " + ejercitoMov.get(0).getNombre().substring(0, ejercitoMov.get(0).getNombre().length() - 2) + " ha ganado!");
140            }
141            return;
142        } else {
143            System.out.println("Movimiento inválido: ya hay un soldado en la nueva posición.");
144            return;
145        }
146    }
147    tablero[filaMov][columnaMov] = null;
148    tablero[nuevaFila][nuevaColumna] = soldado;
149 }
150 public static Soldado batalla(Soldado soldado1, Soldado soldado2) {
151     int totalVida = soldado1.getNivelVida() + soldado2.getNivelVida();
152     double porcentaje1 = (soldado1.getNivelVida() * 100.0) / totalVida;
153     double porcentaje2 = (soldado2.getNivelVida() * 100.0) / totalVida;
154     int numeroAleatorio = (int) (Math.random() * 100) + 1;
155     if (numeroAleatorio <= porcentaje1) {
156         return soldado1;
157     } else {
158         return soldado2;
159     }
160 }
161 public static int calcularSumaDeVidas(ArrayList<Soldado> ejercito) {
162     int sumaVidas = 0;
163     for (Soldado soldado : ejercito) {
164         sumaVidas += soldado.getNivelVida();
165     }
166     return sumaVidas;
167 }
168 public static String batallaEjercito(ArrayList<Soldado> ejercito1, ArrayList<Soldado> ejercito2) {
169     int sumaVidaEjercito1 = calcularSumaDeVidas(ejercito1);
170     int sumaVidaEjercito2 = calcularSumaDeVidas(ejercito2);
171     if (sumaVidaEjercito1 > sumaVidaEjercito2) {
172         return ejercito1.get(0).getNombre();
173     } else if (sumaVidaEjercito2 > sumaVidaEjercito1) {
174         return ejercito2.get(0).getNombre();
175     } else {
176         return "Empate";
177     }
178 }

```

```

173     } else if (sumaVidaEjercito1 < sumaVidaEjercito2) {
174         return ejercito2.get(0).getNombre();
175     } else {
176         return "Empate";
177     }
178 }
179 public static void moverEjercito(Scanner sc, String[][] tablero, ArrayList<ArrayList<Soldado>> reinos) {
180     System.out.print("Ingrese el nombre del reino del cual desea mover ejércitos: ");
181     String reino = sc.next();
182     int reinoIndex = -1;
183     for (int i = 0; i < reinos.size(); i++) {
184         ArrayList<Soldado> ejercitoReino = reinos.get(i);
185         if (!ejercitoReino.isEmpty() && ejercitoReino.get(0).getNombre().contains(reino)) {
186             reinoIndex = i;
187             break;
188         }
189     }
190     if (reinoIndex == -1) {
191         System.out.println("No se encontró el reino " + reino + ".");
192         return;
193     }
194     ArrayList<Soldado> ejercitoReino = reinos.get(reinoIndex);
195     System.out.print("Ingrese las coordenadas del ejército que desea mover (por ejemplo, d3): ");
196     String coordenada = sc.next().toUpperCase();
197     if (coordenada.length() != 2) {
198         System.out.println("Formato de coordenada inválido.");
199         return;
200     }
201     char columnaChar = coordenada.charAt(0);
202     int filaMov = Integer.parseInt(coordenada.substring(1)) - 1;
203     int columnaMov = columnaChar - 'A';
204     if (filaMov < 0 || filaMov >= tablero.length || columnaMov < 0 || columnaMov >= tablero[0].length) {
205         System.out.println("Movimiento inválido: coordenada fuera del tablero.");
206         return;
207     }
208     String coordenadaReino = tablero[filaMov][columnaMov];
209     if (coordenadaReino == null || !coordenadaReino.contains(reino)) {
210         System.out.println("Movimiento inválido: no hay un ejército de " + reino + " en la coordenada seleccionada.");
211         return;
212     }
213     System.out.print("Ingrese la dirección del movimiento (w: arriba, a: izquierda, s: abajo, d: derecha): ");
214     char direccion = sc.next().charAt(0);
215     int nuevaFila = filaMov;
216     int nuevaColumna = columnaMov;
217     if (direccion == 'w') {
218         nuevaFila--;
219     } else if (direccion == 'a') {
220         nuevaColumna--;
221     } else if (direccion == 's') {
222         nuevaFila++;
223     } else if (direccion == 'd') {
224         nuevaColumna++;
225     } else {
226         System.out.println("Movimiento inválido: dirección no válida.");
227         return;
228     }
229     if (nuevaFila < 0 || nuevaFila >= tablero.length || nuevaColumna < 0 || nuevaColumna >= tablero[0].length) {
230         System.out.println("Movimiento inválido: movimiento fuera del tablero.");
231         return;
232     }
233     String coordenadaNueva = tablero[nuevaFila][nuevaColumna];
234     if (coordenadaNueva != null) {
235         ArrayList<Soldado> ejercitoEnemigo = null;
236
237         for (ArrayList<Soldado> ejercito : reinos) {
238             if (ejercito != ejercitoReino && coordenadaNueva.contains(ejercito.get(0).getNombre())) {
239                 ejercitoEnemigo = ejercito;
240                 break;
241             }
242         }
243         if (ejercitoEnemigo != null) {
244             int sumaVidaEjercitoReino = calcularSumaDeVidas(ejercitoReino);
245             int sumaVidaEjercitoEnemigo = calcularSumaDeVidas(ejercitoEnemigo);
246             if (sumaVidaEjercitoReino > sumaVidaEjercitoEnemigo) {
247                 ejercitoEnemigo.clear();
248             } else if (sumaVidaEjercitoEnemigo > sumaVidaEjercitoReino) {
249                 ejercitoReino.clear();
250             } else {
251                 System.out.println("¡Empate! Ambos ejércitos tienen la misma suma de vida.");
252             }
253         }
254     }
255     tablero[filaMov][columnaMov] = null;
256     tablero[nuevaFila][nuevaColumna] = coordenadaReino;
257     ejercitoReino.get(0).setFila(nuevaFila);
258     ejercitoReino.get(0).setColumna((char) (nuevaColumna + 'A'));
259 }
260 }

```

## II. PRUEBA

Pruebas (7) para aplicación (prle-001)

A B C D E F G H I J

```

1 | I | _ | I | _ | _ | I | _ | _ | _ |
2 | _ | _ | I | _ | _ | _ | I | _ | _ |
3 | _ | _ | _ | I | _ | R | _ | _ | _ |
4 | R | _ | _ | _ | _ | _ | _ | _ | _ |
5 | _ | _ | R | _ | _ | _ | _ | _ | _ |
6 | _ | _ | _ | _ | _ | _ | R | _ | _ |
7 | R | _ | _ | _ | _ | _ | _ | _ | I |
8 | _ | _ | _ | I | _ | _ | _ | _ | _ |
9 | _ | _ | _ | _ | _ | _ | _ | _ | _ |
10| _ | _ | _ | _ | _ | I | _ | _ | R |

```

Turno del Jugador 2 (Rusia)

Ingrese el nombre del reino del cual desea mover ejércitos: **I**

Ingrese las coordenadas del ejército que desea mover (por ejemplo, d3): **C5**

Movimiento inválido: no hay un ejército de I en la coordenada seleccionada.

A B C D E F G H I J


```

1 | I | _ | I | _ | _ | I | _ | _ | _ |
2 | _ | _ | I | _ | _ | _ | I | _ | _ |
3 | _ | _ | _ | I | _ | R | _ | _ | _ |
4 | R | _ | _ | _ | _ | _ | _ | _ | _ |
5 | _ | _ | R | _ | _ | _ | _ | _ | _ |
6 | _ | _ | _ | _ | _ | _ | R | _ | _ |
7 | R | _ | _ | _ | _ | _ | _ | _ | I |
8 | _ | _ | _ | I | _ | _ | _ | _ | _ |
9 | _ | _ | _ | _ | _ | _ | _ | _ | _ |
10| _ | _ | _ | _ | _ | I | _ | _ | R |

```

Turno del Jugador 1 (Inglaterra)

Ingrese el nombre del reino del cual desea mover ejércitos:

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

**III. CUESTIONARIO:**

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

CONCLUSIONES
<p>La expansión del programa para enfrentamientos entre jugadores humanos ha mejorado la experiencia del usuario mediante la introducción de métodos estratégicos. La selección de soldados, la planificación de movimientos y la gestión de turnos se han optimizado, resaltando la importancia de los atributos de la clase, como nivelAtaque y vive. La interfaz de usuario se ha mejorado, permitiendo una presentación clara de información. La gestión de eventos y resultados se ha afinado, utilizando atributos como vidaActual. La evaluación de fortalezas y debilidades se ha intensificado, destacando la relevancia de los atributos para estrategias tácticas, enriqueciendo así la experiencia de juego. En resumen, la sólida estructura de clase ha sido crucial para estas mejoras y para permitir una experiencia de juego más interactiva y estratégica.</p>

METODOLOGÍA DE TRABAJO
<ol style="list-style-type: none"> <li>1.- Leer los enunciados cuidadosamente para saber que requiere el problema.</li> <li>2.- Realizar el diagra uml</li> <li>3.-Empezar a hacer las clase soldado y la clase VideoJuego</li> <li>4.- Realizar el Videojuego con los métodos apropiados.</li> </ol>

REFERENCIAS Y BIBLIOGRAFÍA
----------------------------



	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: 9

[1] M. A. Lopez, E. Castro Gutierrez, *Fundamentos de la Programación 2 Topicos 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 los 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 formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	0	

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

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		14	

Tabla 2: Rúbrica para contenido del Informe y demostración