
	<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>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO


### No 06

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	<i>Fundamentos de la Programación 2</i>				
<b>TÍTULO DE LA PRÁCTICA:</b>	<i>ArrayList y Objetos Estándar</i>				
<b>NÚMERO DE PRÁCTICA:</b>	<i>06</i>	<b>AÑO LECTIVO:</b>	<i>2024-B</i>	<b>NRO. SEMESTRE:</b>	<i>II</i>
<b>FECHA DE PRESENTACIÓN</b>	<i>25/10/2024</i>	<b>HORA DE PRESENTACIÓN</b>	<i>17:00:00</i>		
<b>INTEGRANTE (s)</b> <i>Subia Huaicane Edson Fabricio</i>				<b>NOTA (0-20)</b>	<i>Nota colocada por el docente</i>
<b>DOCENTE(s):</b> <i>Lino José Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<ul style="list-style-type: none"> <li><b>EJERCICIOS RESUELTOS:</b></li> </ul> <p>Actividad VIDEOJUEGO de SOLDADOS:</p> <ol style="list-style-type: none"> <li>Cree un Proyecto llamado Laboratorio6</li> <li>Usted deberá crear las dos clases Soldado.java y VideoJuego3.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.</li> <li>Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).</li> <li>El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un ArrayList bidimensional.</li> <li>Tendrá 2 Ejércitos. Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos 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 2 diferentes métodos de ordenamiento.</li> </ol> <p>En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: <a href="https://github.com/Q3son/Videojuego_Soldados.git">https://github.com/Q3son/Videojuego_Soldados.git</a></p>

### Mis COMMITS:

- Este es el primer commit destacable que realicé, acomodé el método para Inicializar el juego y que este funcione correctamente con ArrayList:

Primer versión Videojuego3 hecha con ArrayList: 

- En esta primera versión reemplacé los arreglos privados anteriormente creados (Videojuego2) y los configuré para que funcionen como ArrayList Bidimensionales y Unidimensionales.  
 - Se configuró el constructor del método principal, para inicializar correctamente el ArrayList Bidimensional del tablero, asimismo, el ArrayList de Soldados unidimensional.


Q3son\_04 - 02ff896 +168 -0

1 changed file | Videojuego\_Soldados:Videojuego3.java

```

@@ -0,0 +1,168 @@
1 + import java.util.*;
2 + // BY: SUBIA_EDSON_FP2
3 +
4 + public class Videojuego3 {
5 +
6 +     private static final int TAMAÑO_TABLERO = 10; // Tablero de 10x10
7 +     private ArrayList<ArrayList<Soldado>> tablero;
8 +     private ArrayList<Soldado> soldados;
9 +     private int cantidadSoldados = 0;
10 +
11 +     // 1. Constructor para inicializar el juego con un número determinado de soldados
12 +     public Videojuego3(int cantidad) {
13 +         tablero = new ArrayList<>(TAMAÑO_TABLERO);
14 +         for (int i = 0; i < TAMAÑO_TABLERO; i++){
15 +             ArrayList<Soldado> fila = new ArrayList<>();
16 +             for (int j = 0; j < TAMAÑO_TABLERO; j++){
17 +                 fila.add(null);
18 +             }
19 +             tablero.add(fila);
20 +         }
21 +         soldados = new ArrayList<>(cantidad);
22 +         this.cantidadSoldados = cantidad;
23 +         inicializarSoldados(); // Inicializa los soldados en el tablero
24 +     }
  
```

- Para esta versión acomodé el método para InicializarSoldados() y funcione correctamente con ArrayList.

Segunda versión (full ArrayList): 



- Para esta segunda versión he implementado el método para crear de manera aleatoria a los soldados (InicializarSoldados()) con ArrayList, tiene la misma funcionalidad que la del método con arreglos tradicionales.

Q3son - ebe62f3 +6 -4

1 changed file | Videojuego\_Soldados:Videojuego3.java

```

...↑... @@ -26,16 +26,18 @@ public class Videojuego3 {
26 private void inicializarSoldados() {
27     Random random = new Random();
28     int soldadosCreados = 0;
29 -
30 +
31     while (soldadosCreados < cantidadSoldados) {
32         int fila = random.nextInt(TAMAÑO_TABLERO);
33         int columna = random.nextInt(TAMAÑO_TABLERO);
34 -         if (tablero[fila][columna] == null) { // Verifica que el espacio esté vacío
35 +
36 +         if (tablero.get(fila).get(columna) == null) { // Verifica que el espacio esté vacío
37             String nombre = "Soldado" + soldadosCreados; // Nombre del soldado
38             int nivelVida = random.nextInt(5) + 1; // Nivel de vida entre 1 y 5
39             Soldado soldado = new Soldado(nombre, nivelVida, fila, columna);
40             tablero[fila][columna] = soldado; // Asignar el soldado al tablero
41             soldados[soldadosCreados] = soldado; // Guardar el soldado en el array
42 +
43 +             tablero.get(fila).set(columna, soldado); // Asignar el soldado al tablero
44             soldados.add(soldado); // Guardar el soldado en la lista
45             soldadosCreados++; // Incrementar el contador de soldados creados
46         }
47     }
  
```

	<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> 3</p>

- Se actualizaron los métodos de ordenamiento y puntajes para funcionar con ArrayList

## Cuarta versión con full ArrayList: ↕

### Implementación de métodos de Videojuego3:

- Actualización del método para encontrar el soldado con mayor nivel de vida
- Cálculo del promedio de nivel de vida
- Suma total del nivel de vida de los soldados

VideoJuego\_Soldados\Videojuego3.java

```

71 71 // 4. Método para encontrar el soldado con mayor nivel de vida
72 72 public Soldado soldadoConMayorVida() {
73 - Soldado maxSoldado = soldados[0]; // Asume que hay al menos un soldado
73 + Soldado maxSoldado = soldados.get(0); // Asume que hay al menos un soldado
74 74 for (int i = 1; i < cantidadSoldados; i++) {
75 - if (soldados[i].getNivelVida() > maxSoldado.getNivelVida()) {
76 - maxSoldado = soldados[i];
75 + if (soldados.get(i).getNivelVida() > maxSoldado.getNivelVida()) {
76 + maxSoldado = soldados.get(i);
77 77 }
78 78 }
79 79 return maxSoldado;
↕
83 83 @@ -83,7 +83,7 @@ public class Videojuego3 {
84 84 public double promedioNivelVida() {
85 85 int suma = 0;
86 - suma += soldados[i].getNivelVida();
86 + suma += soldados.get(i).getNivelVida();
87 87 }
88 88 return (double) suma / cantidadSoldados;
89 89 }
↕
92 92 @@ -92,7 +92,7 @@ public class Videojuego3 {
93 93 public int nivelVidaTotal() {
93 93 int total = 0;

```

- Para esta versión actualicé los métodos para mostrar a los soldados según su Ranking de vida y Nombre...

### Actualización de métodos en Videojuego3: - Muestra de datos de l... ↕

...los soldados en orden de creación - Clasificación de soldados por nivel de vida (ranking de poder) - Clasificación de soldados por nombre

Cambios realizados:

#### 1. Método `mostrarDatosSoldados()`:

Se implementó una iteración para imprimir los datos de cada soldado en el orden de su creación en el `ArrayList`.

#### 2. Método `rankingDePoder()`:

Se implementó el ordenamiento manual por selección para ordenar los soldados según su nivel de vida sin utilizar `Collections`. Se intercambian los soldados en el `ArrayList` para lograr el orden adecuado.

#### 3. Método `rankingPorNombre()`:

Se utilizó un ordenamiento burbuja mejorado para ordenar los soldados alfabéticamente por nombre. Se añadió un control de intercambio para optimizar el algoritmo y evitar iteraciones adicionales si la lista ya está ordenada.

VideoJuego\_Soldados\Videojuego3.java



```

114 -         for (int j = i + 1; j < cantidadSoldados; j++) {
115 -             if (soldados[j].getNivelVida() > soldados[indexMax].getNivelVida()) {
116 -                 indexMax = j;
117 +
118 +         // Ordenamiento manual usando método burbuja
119 +         for (int i = 0; i < soldados.size() - 1; i++) {
120 +             for (int j = 0; j < soldados.size() - 1 - i; j++) {
121 +                 if (soldados.get(j).getNivelVida() < soldados.get(j + 1).getNivelVida()) {
122 +                     // Intercambiar soldados
123 +                     Soldado temp = soldados.get(j);
124 +                     soldados.set(j, soldados.get(j + 1));
125 +                     soldados.set(j + 1, temp);

```

```

126 +         // Ordenamiento manual usando método burbuja
127 +         for (int i = 0; i < soldados.size() - 1; i++) {
128 +             for (int j = 0; j < soldados.size() - 1 - i; j++) {
129 +                 if (soldados.get(j).getNombre().compareTo(soldados.get(j + 1).getNombre()) > 0) {
130 +                     // Intercambiar soldados
131 +                     Soldado temp = soldados.get(j);
132 +                     soldados.set(j, soldados.get(j + 1));
133 +                     soldados.set(j + 1, temp);
134 +
135 +             }
136 +         }
137 +
138 +         if (!Cambio) break; // Si no hubo intercambios, el array ya está ordenado
139 +
140 +     }
141 +
142 +     System.out.println("Ranking de soldados por nombre:");
143 +     for (int i = 0; i < cantidadSoldados; i++) {
144 +         System.out.println(soldados[i]);
145 +     }
146 +     for (Soldado soldado : soldados) {
147 +         System.out.println(soldado);
148 +     }
149 +     System.out.println();
150 + }

```

- **Para la versión final definitiva, agregué a los métodos para agregar 2 ejércitos y decidir un ganador, entre otras como la de solicitar estadísticas...**

**Versión final definitiva:**
**Inicialización de dos ejércitos:**

Añadidos ejercito1 y ejercito2 como ArrayLists de soldados, cada uno inicializado con soldados con posiciones y atributos aleatorios.

Actualización del método inicializarSoldados() para permitir la creación de soldados en posiciones únicas para ambos ejércitos, generando nombres en formato SoldadoXEjY (ej. Soldado0X1).

Actualización de mostrarTablero():

Modificado para representar visualmente en el tablero la diferencia entre soldados de ambos ejércitos.

**Estadísticas por ejército:**

Implementación de métodos para obtener estadísticas por ejército:

soldadoConMayorVida() encuentra el soldado con mayor vida en cada ejército.

promedioNivelVida() calcula el promedio de nivel de vida para cada ejército.

nivelVidaTotal() calcula el total de vida por ejército.

**Funcionalidad de Ranking de Poder:**

rankingDePoder() ordena soldados en cada ejército por nivel de vida usando el algoritmo de selección.



**Determinación del Ejército Ganador:**

Añadido el método determinarGanador() para comparar el total de puntos de vida de ambos ejércitos y determinar el ganador.

```
14 +         for (int i = 0; i < TAMAÑO_TABLERO; i++){
15 +             tablero.add(new ArrayList<>(TAMAÑO_TABLERO));
16 +             for (int j = 0; j < TAMAÑO_TABLERO; j++){
17 +                 tablero.get(i).add(null);
18 +             }
19 -         tablero.add(fila);
20 -     }
21 -     soldados = new ArrayList<>(cantidad);
22 -     this.cantidadSoldados = cantidad;
23 -     inicializarSoldados(); // Inicializa los soldados en el tablero
24 +
25 +     ejercito1 = new ArrayList<>(cantidadSoldadosEjercito1);
26 +     ejercito2 = new ArrayList<>(cantidadSoldadosEjercito2);
27 +
28 +     inicializarEjercito(ejercito1, cantidadSoldadosEjercito1, "X1");
29 +     inicializarEjercito(ejercito2, cantidadSoldadosEjercito2, "X2");
```

```
110 + // Método para ordenar soldados de un ejército por poder (nivel de vida) - Algoritmo de selección
111 + public void rankingDePoder(ArrayList<Soldado> ejercito) {
112 +     for (int i = 0; i < ejercito.size() - 1; i++) {
113 +         int indexMax = i;
114 +         for (int j = i + 1; j < ejercito.size(); j++) {
115 +             if (ejercito.get(j).getNivelVida() > ejercito.get(indexMax).getNivelVida()) {
116 +                 indexMax = j;
117 +             }
118 +         }
119 +         if (indexMax != i) {
120 +             Soldado temp = ejercito.get(indexMax);
121 +             ejercito.set(indexMax, ejercito.get(i));
122 +             ejercito.set(i, temp);
123 +         }
124 +     }
```

```
154 +
155 +     System.out.println("Soldado con mayor nivel de vida en Ejército 1: " + juego.soldadoConMayorVida(juego.ejercito1));
156 +     System.out.println("Soldado con mayor nivel de vida en Ejército 2: " + juego.soldadoConMayorVida(juego.ejercito2));
157 +
158 +     System.out.println("Promedio de nivel de vida en Ejército 1: " + juego.promedioNivelVida(juego.ejercito1));
159 +     System.out.println("Promedio de nivel de vida en Ejército 2: " + juego.promedioNivelVida(juego.ejercito2));
160 +
161 +     juego.mostrarDatosEjercito(juego.ejercito1, "Ejército 1");
162 +     juego.mostrarDatosEjercito(juego.ejercito2, "Ejército 2");
163 +
164 +     juego.rankingDePoder(juego.ejercito1);
165 +     juego.rankingDePoder(juego.ejercito2);
166 +
167 +     juego.determinarGanador();
```

	<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: 6</p>

**En la siguiente sección mostraré el código fuente y ejecución de la versión final de mi código fuente del programa, trabajado en Visual Studio, en cada captura de pantalla se visualizará el buen funcionamiento de los nuevos métodos adicionados y fundamentados correctamente. (El código fuente se visualiza mucho mejor en mi repositorio)**

```

You, 29 seconds ago | 2 authors (You and one other)
1  import java.util.*;
You, 29 seconds ago | 2 authors (You and one other)
2  // BY: SUBIA_EDSON_FP2
3
4  public class Videojuego3 {
5
6      private static final int TAMAÑO_TABLERO = 10; // Tablero de 10x10
7      private ArrayList<ArrayList<Soldado>> tablero;
8      private ArrayList<Soldado> ejercito1;
9      private ArrayList<Soldado> ejercito2;
10
11     // 1. Constructor para inicializar el juego con soldados en ambos ejércitos
12     public Videojuego3(int cantidadSoldadosEjercito1, int cantidadSoldadosEjercito2) {
13         tablero = new ArrayList<>(TAMAÑO_TABLERO);
14         for (int i = 0; i < TAMAÑO_TABLERO; i++) {
15             tablero.add(new ArrayList<>(TAMAÑO_TABLERO));
16             for (int j = 0; j < TAMAÑO_TABLERO; j++) {
17                 tablero.get(i).add(e:null);
18             }
19         }
20
21         ejercito1 = new ArrayList<>(cantidadSoldadosEjercito1);
22         ejercito2 = new ArrayList<>(cantidadSoldadosEjercito2);
23
24         inicializarEjercito(ejercito1, cantidadSoldadosEjercito1, etiquetaEjercito:"X1");
25         inicializarEjercito(ejercito2, cantidadSoldadosEjercito2, etiquetaEjercito:"X2");
26     }
27
28     // 2. Método para inicializar un ejército con soldados aleatorios
29     private void inicializarEjercito(ArrayList<Soldado> ejercito, int cantidad, String etiquetaEjercito) {
30         Random random = new Random();
31         int soldadosCreados = 0;
32
33         while (soldadosCreados < cantidad) {
34             int fila = random.nextInt(TAMAÑO_TABLERO);
35             int columna = random.nextInt(TAMAÑO_TABLERO);
36             if (tablero.get(fila).get(columna) == null) { // Verifica que el espacio esté vacío
37                 String nombre = "Soldado" + soldadosCreados + etiquetaEjercito;
38                 int nivelVida = random.nextInt(bound:5) + 1;
39                 Soldado soldado = new Soldado(nombre, nivelVida, fila, columna);
40
41                 tablero.get(fila).set(columna, soldado); // Asignar el soldado al tablero
42                 ejercito.add(soldado); // Guardar el soldado en el ejército
43                 soldadosCreados++;
44             }
45         }
46     }
47

```



```
48 // 3. Método para mostrar el tablero en la consola
49 public void mostrarTablero() {
50     System.out.println(x:"Tablero de Soldados:");
51     String lineaDivisoria = new String(new char[100]).replace(target:"\0", replacement:"-");
52
53     for (int fila = 0; fila < TAMAÑO_TABLERO; fila++) {
54         System.out.println(lineaDivisoria);
55         System.out.print(s:"|");
56
57         for (int columna = 0; columna < TAMAÑO_TABLERO; columna++) {
58             String celda = " "; // Espacio vacío
59             Soldado soldado = tablero.get(fila).get(columna);
60
61             if (soldado != null) {
62                 celda = String.format(format:"%-9s", soldado.getNombre()); // Nombre del soldado
63             }
64
65             System.out.print(celda + "|");
66         }
67         System.out.println();
68     }
69     System.out.println(lineaDivisoria); // Línea divisoria final
70 }
71
72 // 4. Método para mostrar datos del soldado con mayor nivel de vida de un ejército
73 public Soldado soldadoConMayorVida(ArrayList<Soldado> ejercito) {
74     Soldado maxSoldado = ejercito.get(index:0);
75     for (Soldado soldado : ejercito) {
76         if (soldado.getNivelVida() > maxSoldado.getNivelVida()) {
77             maxSoldado = soldado;
78         }
79     }
80     return maxSoldado;
81 }
82
83 // 5. Método para calcular el promedio de nivel de vida de un ejército
84 public double promedioNivelVida(ArrayList<Soldado> ejercito) {
85     int suma = 0;
86     for (Soldado soldado : ejercito) {
87         suma += soldado.getNivelVida();
88     }
89     return (double) suma / ejercito.size();
90 }
91
92 // 6. Método para calcular el nivel de vida total de un ejército
93 public int nivelVidaTotal(ArrayList<Soldado> ejercito) {
94     int total = 0;
95     for (Soldado soldado : ejercito) {
96         total += soldado.getNivelVida();
97     }
98     return total;
99 }
100
101 // 7. Método para mostrar datos de los soldados de un ejército
102 public void mostrarDatosEjercito(ArrayList<Soldado> ejercito, String nombreEjercito) {
103     System.out.println("Datos de los soldados en " + nombreEjercito + " en orden de creación:");
104     for (Soldado soldado : ejercito) {
105         System.out.println(soldado);
106     }
107     System.out.println();
108 }
```

```
110 // 8. Método para ordenar soldados de un ejército por poder (nivel de vida) - Algoritmo de selección
111 public void rankingDePoder(ArrayList<Soldado> ejercito) {
112     for (int i = 0; i < ejercito.size() - 1; i++) {
113         int indexMax = i;
114         for (int j = i + 1; j < ejercito.size(); j++) {
115             if (ejercito.get(j).getNivelVida() > ejercito.get(indexMax).getNivelVida()) {
116                 indexMax = j;
117             }
118         }
119         if (indexMax != i) {
120             Soldado temp = ejercito.get(indexMax);
121             ejercito.set(indexMax, ejercito.get(i));
122             ejercito.set(i, temp);
123         }
124     }
125
126     System.out.println("Ranking de poder de soldados en " + (ejercito == ejercito1 ? "Ejército 1" : "Ejército 2") + " (ordenados por nivel de vida):");
127     for (Soldado soldado : ejercito) {
128         System.out.println(soldado);
129     }
130     System.out.println();
131 }
132
133 // 9. Método para decidir cuál ejército gana la batalla en función del total de vida
134 public void determinarGanador() {
135     int vidaTotalEjercito1 = nivelVidaTotal(ejercito1);
136     int vidaTotalEjercito2 = nivelVidaTotal(ejercito2);
137
138     System.out.println("Vida total del Ejército 1: " + vidaTotalEjercito1);
139     System.out.println("Vida total del Ejército 2: " + vidaTotalEjercito2);
140
141     if (vidaTotalEjercito1 > vidaTotalEjercito2) {
142         System.out.println(x: "¡Ejército 1 gana la batalla!");
143     } else if (vidaTotalEjercito2 > vidaTotalEjercito1) {
144         System.out.println(x: "¡Ejército 2 gana la batalla!");
145     } else {
146         System.out.println(x: "La batalla termina en empate.");
147     }
148 }
149
```

```
150 // 10. Método principal para ejecutar el juego You, 1 second ago • Uncommitted changes
Run | Debug
151 public static void main(String[] args) {
152     Videojuego3 juego = new Videojuego3(cantidadSoldadosEjercito1:10, cantidadSoldadosEjercito2:10); // Inicia el juego con 10 soldados en cada ejército
153     juego.mostrarTablero();
154
155     System.out.println("Soldado con mayor nivel de vida en Ejército 1: " + juego.soldadoConMayorVida(juego.ejercito1));
156     System.out.println("Soldado con mayor nivel de vida en Ejército 2: " + juego.soldadoConMayorVida(juego.ejercito2));
157
158     System.out.println("Promedio de nivel de vida en Ejército 1: " + juego.promedioNivelVida(juego.ejercito1));
159     System.out.println("Promedio de nivel de vida en Ejército 2: " + juego.promedioNivelVida(juego.ejercito2));
160
161     juego.mostrarDatosEjercito(juego.ejercito1, nombreEjercito:"Ejército 1");
162     juego.mostrarDatosEjercito(juego.ejercito2, nombreEjercito:"Ejército 2");
163
164     juego.rankingDePoder(juego.ejercito1);
165     juego.rankingDePoder(juego.ejercito2);
166
167     juego.determinarGanador();
168 }
169 }
```



¡Ejército 2 gana la batalla!

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>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>

### ¿Con qué valores comprobaste que tu práctica estuviera correcta?



Comprobé la práctica utilizando valores aleatorios al generar soldados para dos ejércitos. Asigné nombres autogenerados como "Soldado0X1", "Soldado1X1", variando aleatoriamente la fila y columna de cada soldado y asegurando que no se repitieran las posiciones en el tablero. También generé valores de nivel de vida aleatorios entre 1 y 5 para verificar que el tablero se mostrara correctamente y que cada ejército mantuviera sus soldados en posiciones distintas. Para probar el ranking, utilicé los métodos de ordenamiento de selección y burbuja en soldados de cada ejército.

### ¿Qué resultado esperabas obtener para cada valor de entrada? Esperaba que el programa tuviera el siguiente comportamiento:

- Al ordenar soldados por nivel de vida (método de selección y burbuja): Los soldados de cada ejército se ordenarían correctamente de acuerdo con sus niveles de vida, mostrando el orden descendente esperado.
- Al mostrar el soldado con mayor nivel de vida por ejército: Se esperaba que se identificara correctamente el soldado con el mayor nivel de vida en cada ejército.
- Al calcular el promedio y el total de puntos de vida por ejército: El programa mostraría los valores adecuados para el promedio de vida y el total de vida de cada ejército.
- Al determinar el ejército ganador: Se esperaba que el ejército con mayor total de puntos de vida fuera declarado como ganador.

### ¿Qué valor o comportamiento obtuviste para cada valor de entrada? Los resultados obtenidos fueron los esperados:

- Al ordenar soldados por nivel de vida: Los soldados de ambos ejércitos se mostraron ordenados correctamente en función de sus niveles de vida, validando la eficacia del algoritmo de selección y burbuja para el ranking.
- Al mostrar el soldado con mayor nivel de vida: El programa identificó correctamente el soldado con el mayor nivel de vida en cada ejército.
- Al calcular el promedio y total de puntos de vida por ejército: Los resultados mostraron los valores esperados, confirmando que los métodos de cálculo se implementaron correctamente.
- Al determinar el ejército ganador: El programa identificó correctamente al ejército con el mayor total de puntos de vida, cumpliendo con las métricas definidas para decidir el ganador de la batalla.

	<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> 11</p>



## 1. CUESTIONARIO:

Los datos más importantes que consideré para realizar el código del juego de Soldados, especialmente en relación con las nuevas funcionalidades y requisitos, fueron los siguientes:

2. **Atributos del Soldado:** Cada soldado cuenta con atributos clave como nombre, fila, columna y nivel de vida. Estos atributos son fundamentales para definir las características de cada soldado en el tablero. El nivel de vida es especialmente crítico, ya que determina la resistencia de cada soldado en la batalla y, por ende, influye en la estrategia del juego.
3. **Arreglo bidimensional:** Utilizar un arreglo bidimensional para almacenar los objetos de tipo Soldado es esencial. Este arreglo simula el tablero de juego, permitiendo que los soldados se ubiquen en posiciones específicas y facilitando la gestión de sus interacciones durante la partida. Además, esta estructura es crucial para aplicar los métodos de ordenamiento que permiten organizar a los soldados de acuerdo a su nivel de vida o posición.
4. **Generación aleatoria de soldados:** La creación de soldados con valores aleatorios, como el nivel de vida y las posiciones, asegura un entorno dinámico y desafiante para el juego. Esto incluye garantizar que no haya dos soldados en la misma posición del tablero, lo que agrega un nivel de estrategia adicional al juego.
5. **Métodos de ordenamiento:** La implementación de métodos de ordenamiento es crucial para organizar a los soldados según diferentes criterios:
  - **Ordenamiento por selección:** Se utilizó para ordenar a los soldados por nivel de vida, permitiendo identificar rápidamente cuáles son los más débiles y cuáles están en mejor estado. Esto es útil para tomar decisiones estratégicas durante la partida.
  - **Ordenamiento por burbuja:** Este método se utilizó para organizar a los soldados por nombre, facilitando la búsqueda y gestión del ejército, lo que mejora la experiencia del usuario.
6. **Cálculo de promedios y totales:** Los métodos que calculan el promedio y total de puntos de vida por ejército son esenciales para evaluar la fuerza general de cada ejército. Esto ayuda a determinar quién ganará la batalla, basándose en el total de vida de los soldados, y aporta a la lógica del juego.
7. **Determinación del ejército ganador:** Definir qué ejército ganará la batalla es fundamental para la dinámica del juego. Se basa en la métrica del total de puntos de vida, permitiendo a los jugadores evaluar el rendimiento de sus ejércitos de manera clara y concisa.
8. **Interacción con el usuario:** Todos estos elementos mejoran la experiencia del usuario, al permitirle ver los soldados organizados y calcular los resultados de manera eficiente. Los jugadores pueden evaluar fácilmente la capacidad de su ejército y tomar decisiones informadas, enriqueciendo así la jugabilidad.

## CONCLUSIONES

*Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.*

	<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> 12</p>

La implementación del juego de Soldados ha sido exitosa tanto en su estructura como en la interacción que ofrece al usuario. La incorporación de algoritmos de ordenamiento ha permitido gestionar los atributos de los soldados, como el nivel de vida y el nombre, de manera eficiente. Al ordenar por nivel de vida utilizando el método de selección, el jugador puede visualizar fácilmente qué soldados tienen mayor resistencia, lo que facilita la toma de decisiones estratégicas. Por otro lado, el ordenamiento por nombre mediante el método de burbuja mejora la gestión alfabética de los soldados, optimizando la experiencia de búsqueda y análisis.

### METODOLOGÍA DE TRABAJO

*Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.*

- a) **Comprensión del problema:** En esta etapa, revisé cada una de las actividades propuestas, identificando cuidadosamente las restricciones y los objetivos a alcanzar.
- b) **Diseño del algoritmo:** Planifiqué la secuencia lógica necesaria para implementar la solución, aplicando los conocimientos adquiridos en Fundamentos de Programación I y II.
- c) **Codificación:** Procedí a implementar los programas solicitados, asegurándome de utilizar correctamente los arreglos y métodos.
- d) **Pruebas:** Realicé pruebas adicionales para verificar que el código funcionara de manera correcta con diferentes casos de prueba.



### REFERENCIAS Y BIBLIOGRAFÍA

*Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE*

M. W. Aedo López, *Fundamentos de programación I: Java Básico*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, jul. 2019. ISBN: 978-612-4337-55-0. 116 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 21 x 29.7 cm

[https://github.com/LINOPINTO2023/FundProq2/blob/main/entregaLaboratorio01/Hilacondo\\_Emanuel\\_LABORATORIO\\_01.pdf](https://github.com/LINOPINTO2023/FundProq2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf)

[https://github.com/Q3son/Videojuego\\_Soldados.git](https://github.com/Q3son/Videojuego_Soldados.git)

	<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 style="text-align: center;"><b>Código:</b> GUIA-PRLE-001</p>	<p style="text-align: right;"><b>Página:</b> 13</p>

### RÚBRICA 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 con 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	4	
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	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	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	8	19	